МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Методичні вказівки до виконання
практичних робіт з дисципліни

"ПРОГРАМУВАННЯ ДЛЯ МОБІЛЬНИХ ПЛАТФОРМ"

Тернопіль – 2017

Р.П. Шевчук, М.В. Сусла // Методичні вказівки до виконання практичних робіт з дисципліни „Програмування для мобільних платформ”, для студентів за спеціальністю 121 «Інженерія програмного забезпечення». – Тернопіль, 2017. – 110 с.

Укладачі: **Шевчук Руслан Петрович**, к.т.н., доцент кафедри комп’ютерних наук ТНЕУ
Сусла Михайло Володимирович, викладач кафедри комп’ютерних наук ТНЕУ

Відповідальний за випуск: **Пукас Андрій Васильович**, к.т.н., доцент., завідувач кафедри комп’ютерних наук ТНЕУ

Рецензенти: доцент кафедри інформаційно - обчислювальних систем і управління, д.т.н., доцент **Яцків В.В.**

доцент кафедри комп’ютерних технологій Тернопільського національного педагогічного університету ім. В. Гнатюка, к.т.н., доцент **Франко Ю.П.**

Затверджено на засіданні кафедри комп’ютерних наук ТНЕУ.
Протокол № 10 від «17» січня 2017 р.

Рекомендовано до друку науково-методичною радою факультету комп’ютерних інформаційних технологій ТНЕУ. Протокол № 5 від 28 лютого 2017 р.

ЗМІСТ

ВСТУП 4
ПРАКТИЧНА РОБОТА №1 Встановлення та налаштування компонентів середовища розробки мобільних програм для ОС Android. 5
ПРАКТИЧНА РОБОТА №2 Створення першої програми в Android Studio для ОС Android 18
ПРАКТИЧНА РОБОТА №3 Робота із активностями, кольором та локалізацією в ОС Android 26
ПРАКТИЧНА РОБОТА №4 Робота із анімацією та макетами в ОС Android 44
ПРАКТИЧНА РОБОТА №5 Робота із віджетами та керуючими елементами 55
ПРАКТИЧНА РОБОТА №6 Наміри в ОС Android 72
ПРАКТИЧНА РОБОТА №7 Робота з базами даних в ОС Android 87
ПРАКТИЧНА РОБОТА №8 Робота із контент-провайдерами 96
СПИСОК ЛІТЕРАТУРИ 109

ВСТУП

Розробка мобільних додатків є відносно молодою областю наукових досліджень. Незважаючи на це, кількість і складність завдань, які виконуються мобільними додатками незмінно зростає. Серед них можна виділити як загальні наукові завдання з розпізнавання зображень, звуку, створення штучного інтелекту або паралельного програмування, так і специфічні - дослідження різних інтерактивних способів взаємодії з людиною, способи побудови адаптивного інтерфейсу користувача та мобільних кібер-фізичних систем. Мобільні додатки можуть встановлюватись на пристрій в процесі виробництва, завантажуватись користувачами за допомогою різних платформ для поширення програмного забезпечення або реалізовуватись у вигляді клієнт-серверних додатків.

Усі найновіші технології відразу ж стають доступні на мобільних пристроях. Наприклад, голосове керування, віртуальний помічник і співрозмовник (Siri в ОС IOS і Google Now в Android), розпізнавання тексту і предметів і т.д.

Мобільні додатки тісно інтегруються з операційною системою - можуть передавати і отримувати через неї дані з інших програм, обробляти запити операційної системи і приймати сигнали про події, такі як зміна орієнтації пристрою, підключення або відключення бездротової мережі. Звернення до операційної системи виробляються через так званий програмний інтерфейс API (Application Programming Interface). Код, який реалізує API-функції, міститься у пам'яті тільки в одному екземплярі, що робить додатки значно компактнішими та зменшує кількість споживаних ресурсів персонального мобільного пристрою.

Найпопулярнішою на сьогоднішній день платформою для якої розробляють мобільні додатки є операційна система Android, яка базується на ядрі Linux. На даний час її підтримкою та розробкою займається консорціум Open Handset Alliance, ініційований компанією Google після покупки компанії Android Inc.

Операційна система Android забезпечує простоту та зручність використання і налаштування системи, захист даних від зараження вірусами завдяки - ізольованій роботі кожного додатку, високу функціональність в користуванні Інтернетом, зручну роботу з електронною поштою, підтримку додатків Adobe Flash, широкий спектр можливостей підключення - Wi-Fi, Bluetooth, GPRS, EDGE, 3G та багато іншого. Android додатки пишуться на мові програмування Java та виконуються на віртуальній машині - Dalvik Virtual Machine.

Даний курс практичних робіт з дисципліни «Програмування для мобільних платформ» побудований на вирішенні практичних задач із детальною візуалізацією процесів створення мобільних додатків для операційної системи Android із використанням середовища розробки Android Studio.

ПРАКТИЧНА РОБОТА №1

Тема роботи: Встановлення та налаштування компонентів середовища розробки мобільних програм для ОС Android.

Мета роботи: Отримати досвід встановлення та налаштування Java Development Kit та Android Studio.

Теоретичні відомості

Мобільні додатки для операційної системи Android як правило розробляються на універсальному ПК, де ресурси не обмежені (порівняно із мобільним пристроєм) і завантажуються на цільовий персональний мобільний пристрій для відлагодження, тестування та подальшого використання. Додатки можна відлагоджувати і тестувати на реальному пристрої під управлінням операційної системи Android або на емуляторі. Для попередньої розробки додатку та його відлагодження зручніше використовувати емулятор, а потім виконувати остаточне тестування на реальних пристроях.

Розробникам мобільних додатків надається можливість використовувати засоби розробки додатків для Android на ПК під управлінням будь-якої з поширених операційних систем сімейств Windows, Linux або Mac OS X. Нижче буде детально описаний процес встановлення компонентів середовища розробки мобільних додатків IDE Android Studio під операційну систему Windows 7.

IDE Android Studio - інтегроване середовище розробки додатків розроблене компанією Google для операційної системи Android. Даний продукт забезпечує розробників новими інструментами для створення мобільних додатків, а також являється альтернативою середовищу Eclipse, що є в даний час найбільш популярним.

IDE Android Studio базується на програмному забезпеченні IntelliJ IDEA від компанії JetBrains та є офіційним засобом розробки Android додатків.

Перед встановленням Android Studio необхідно встановити Java SE Development Kit (JDK) і прописати системну змінну JAVA_HOME.

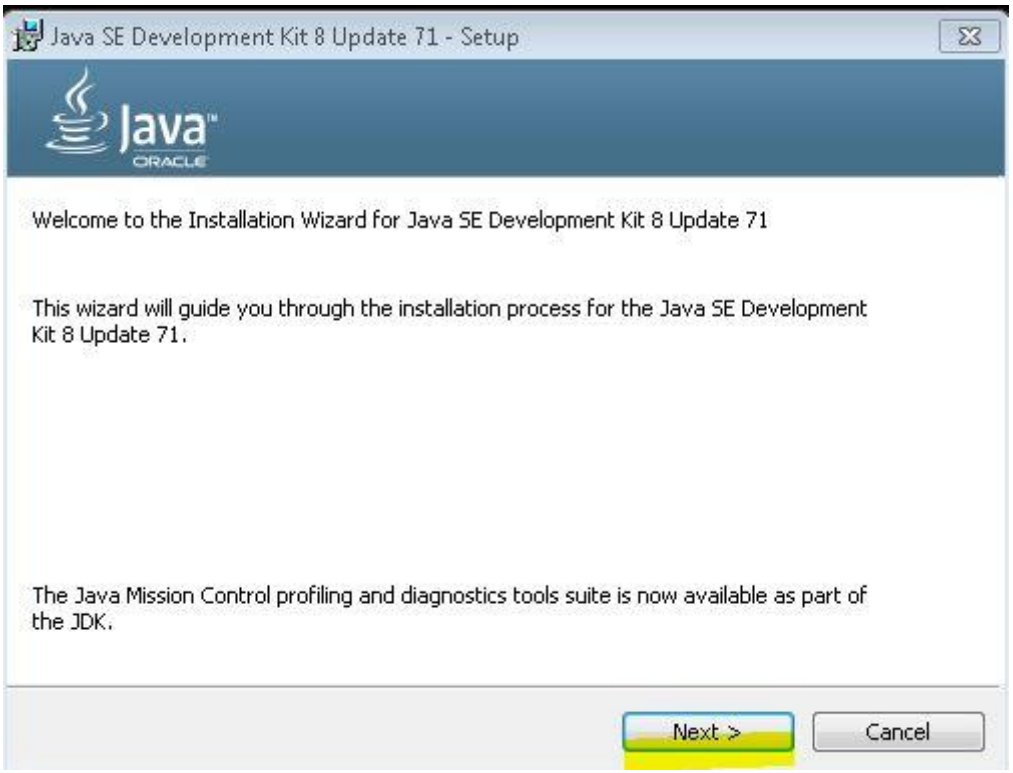
Завантаження та встановлення JDK

На сайті компанії Oracle вибираємо кнопку Download під написом JDK, приймаємо угоду і завантажуємо Java SE Development Kit 7 для Windows x64 (в разі 64-х розрядної архітектури) або Windows x86.

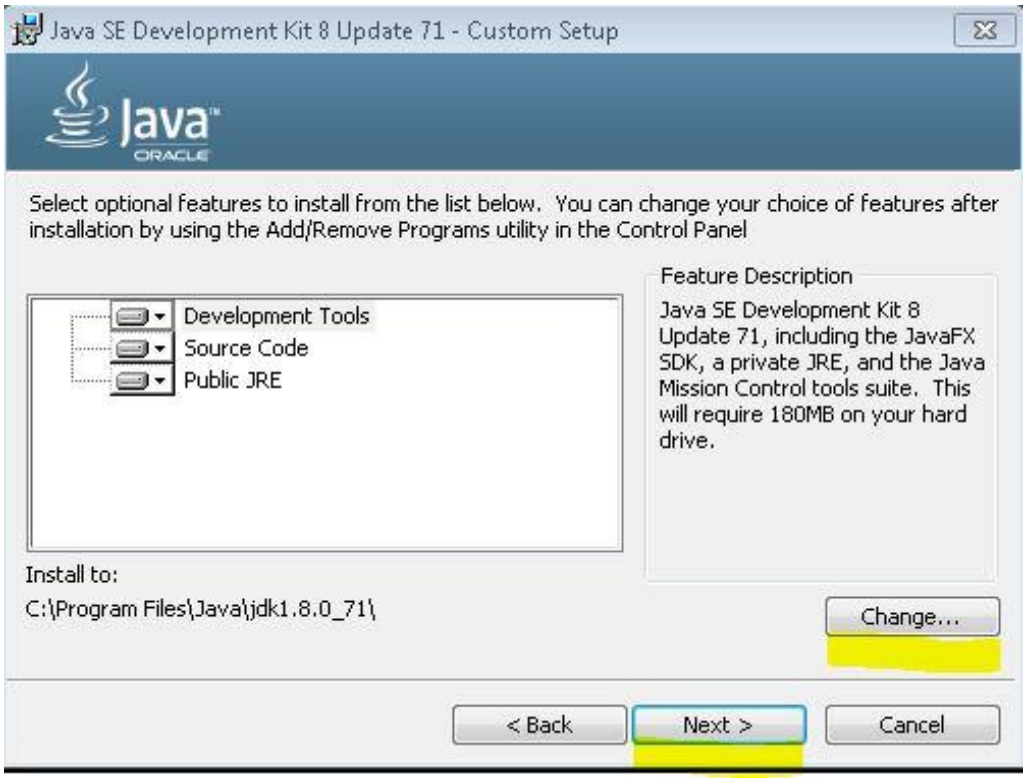
Сторінка завантаження JDK:
http://www.oracle.com/technetwork/java/javase/downloads/index.html

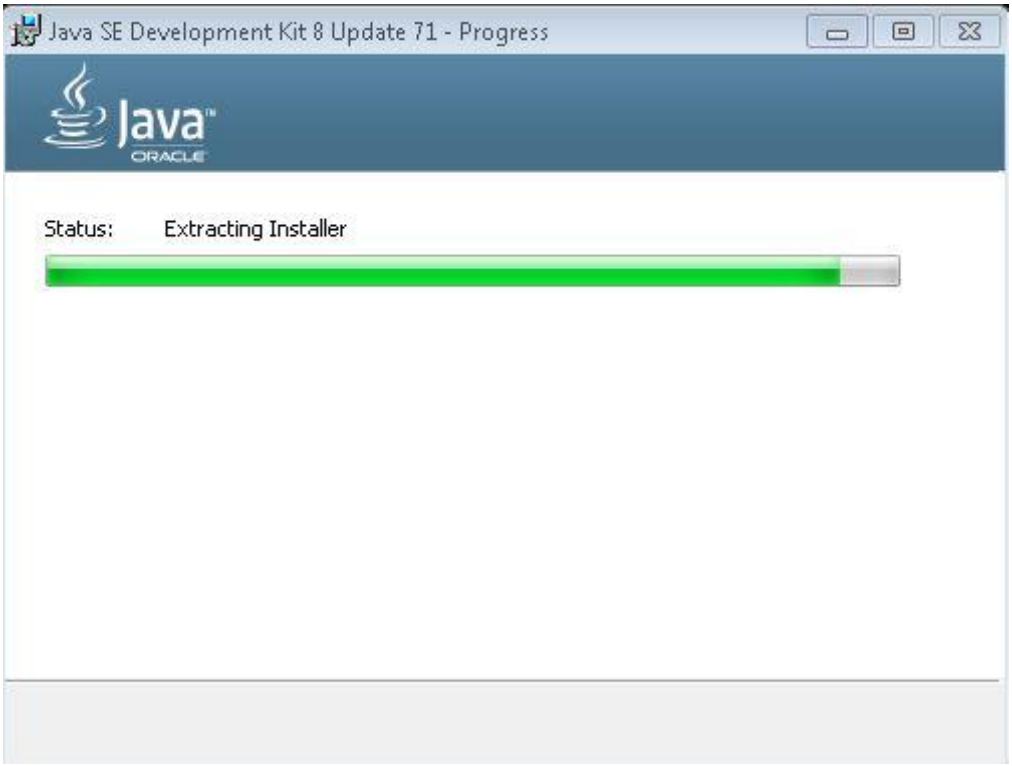
Для встановлення JDK необхідно мати права адміністратора ПК. Перевіряємо, чи встановлена на комп'ютері JDK (Пункт меню «Встановлення і видалення програм» в панелі управління ОС Windows 7) і, якщо так, то видаляємо його та запускаємо завантажений файл.

Екранні форми подані нижче ілюструють процес встановлення завантаженого JDK.

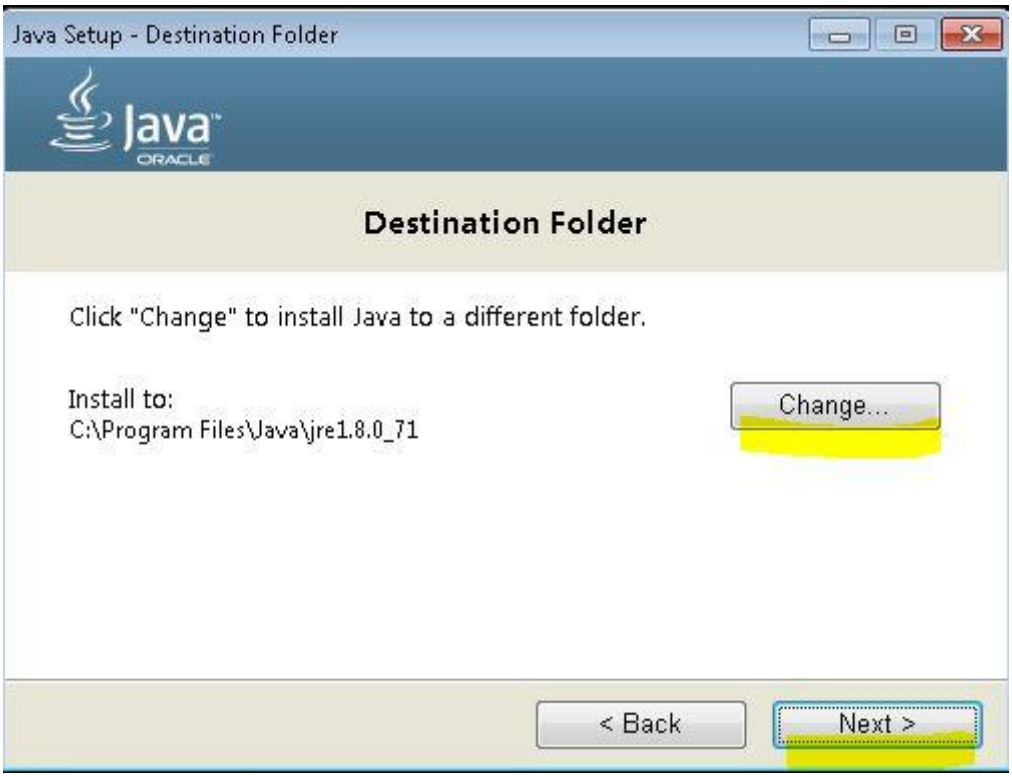


Якщо каталог для встановлення за замовчуванням не підходить, його можна змінити, а потім продовжити встановлення.





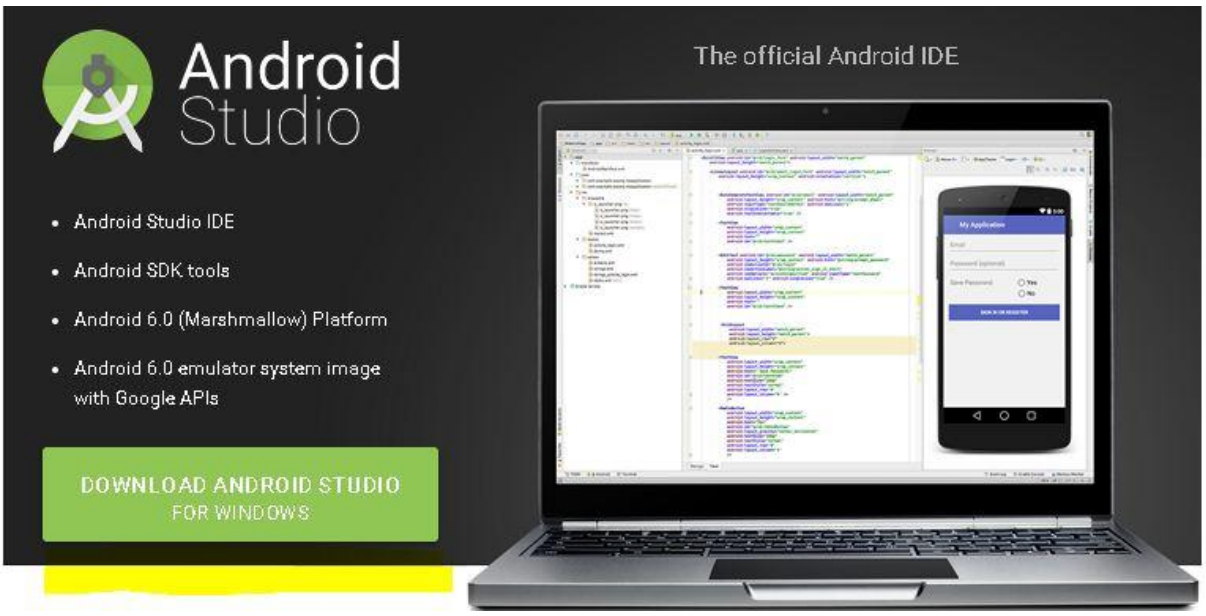
При встановленні Java Runtime так само можна змінити поточний каталог та продовжити встановлення.



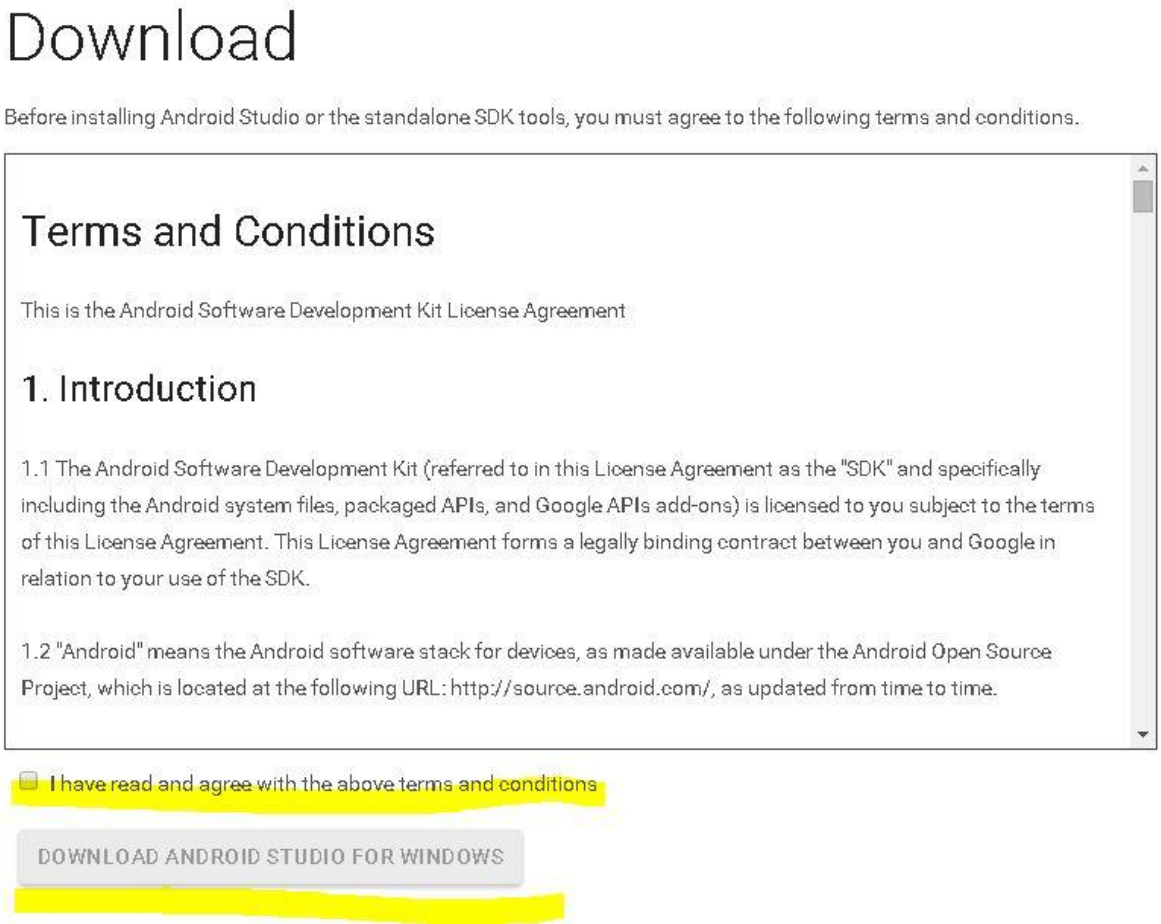
Подана нижче екранна форма ілюструє успішне закінчення процесу встановлення JDK.

Завантаження, встановлення та налаштування Android Studio

Android Studio необхідно завантажити із сайту:
http://developer.android.com/sdk/index.html#top

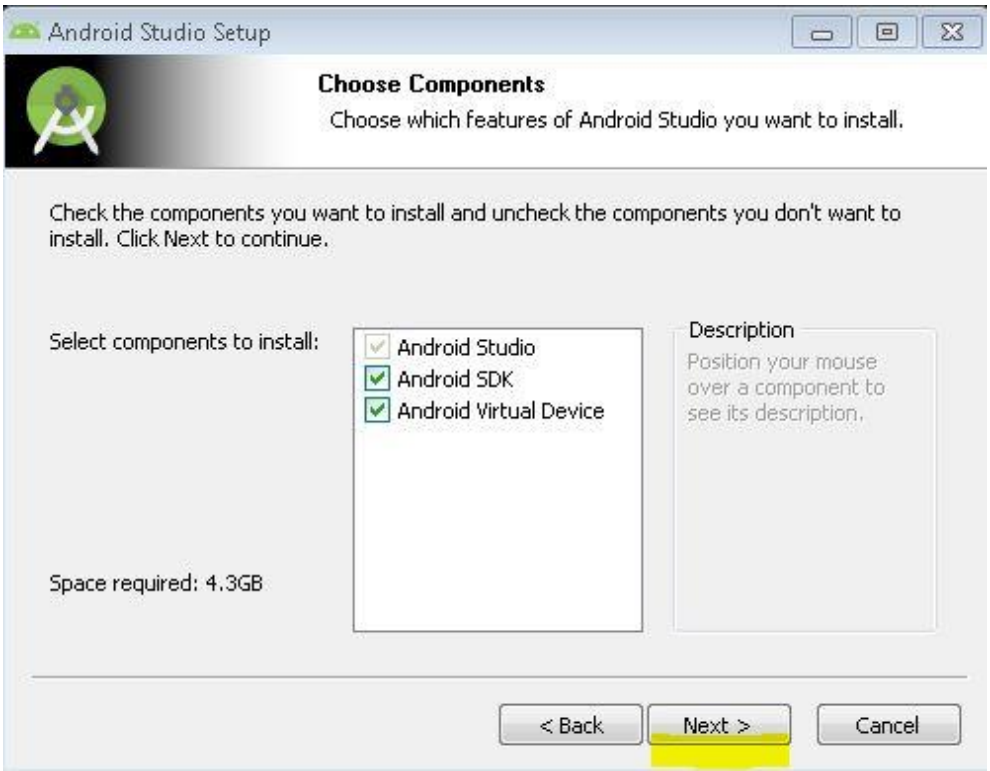


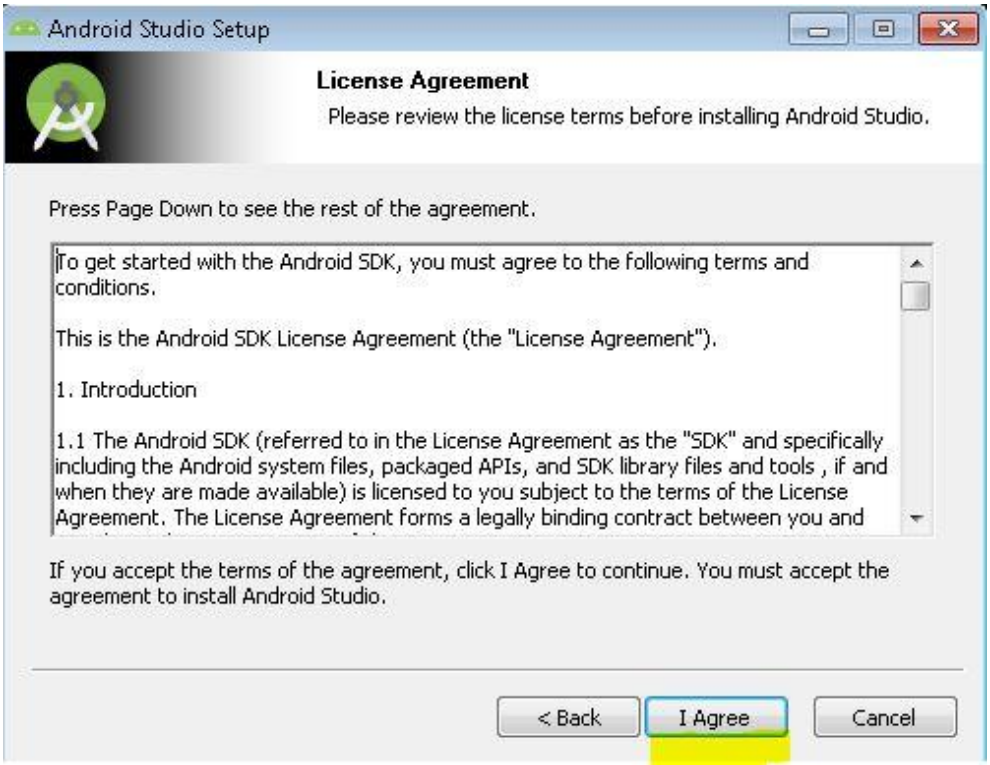
Натискаємо на виділену кнопку.

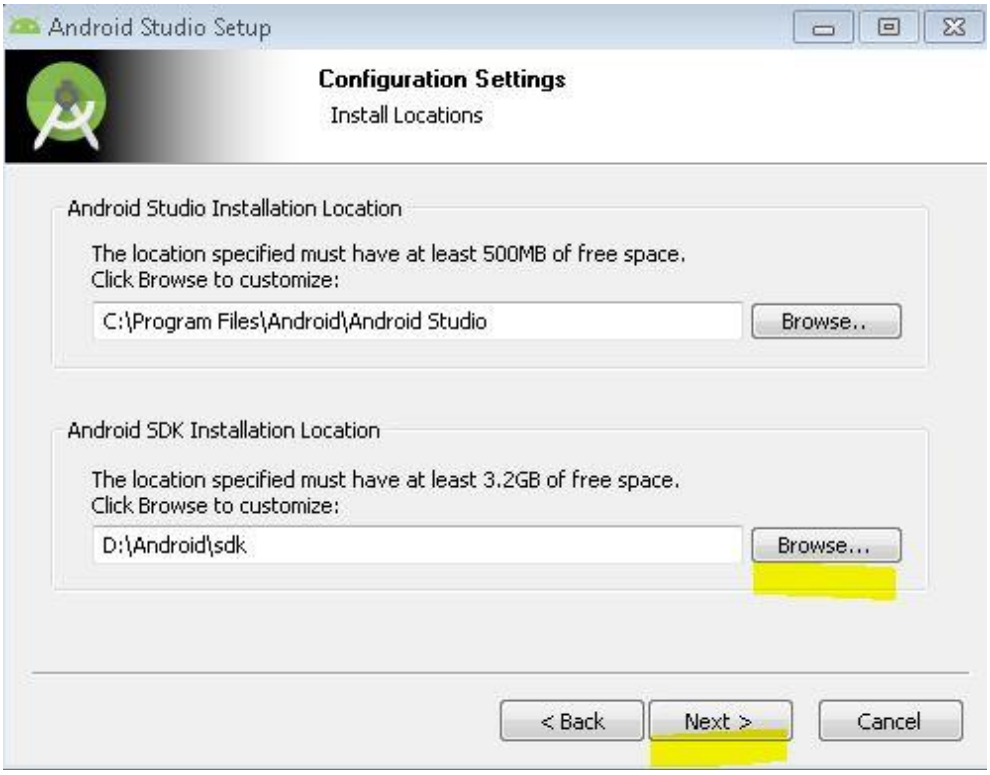


Ставимо відмітку на позначеному полі, і завантажуємо Android Studio. Наступним кроком є встановлення завантаженого файлу «android-studio…. exe».

Далі виконуємо все як на екранних формах. У Android Studio є можливість встановити SDK.

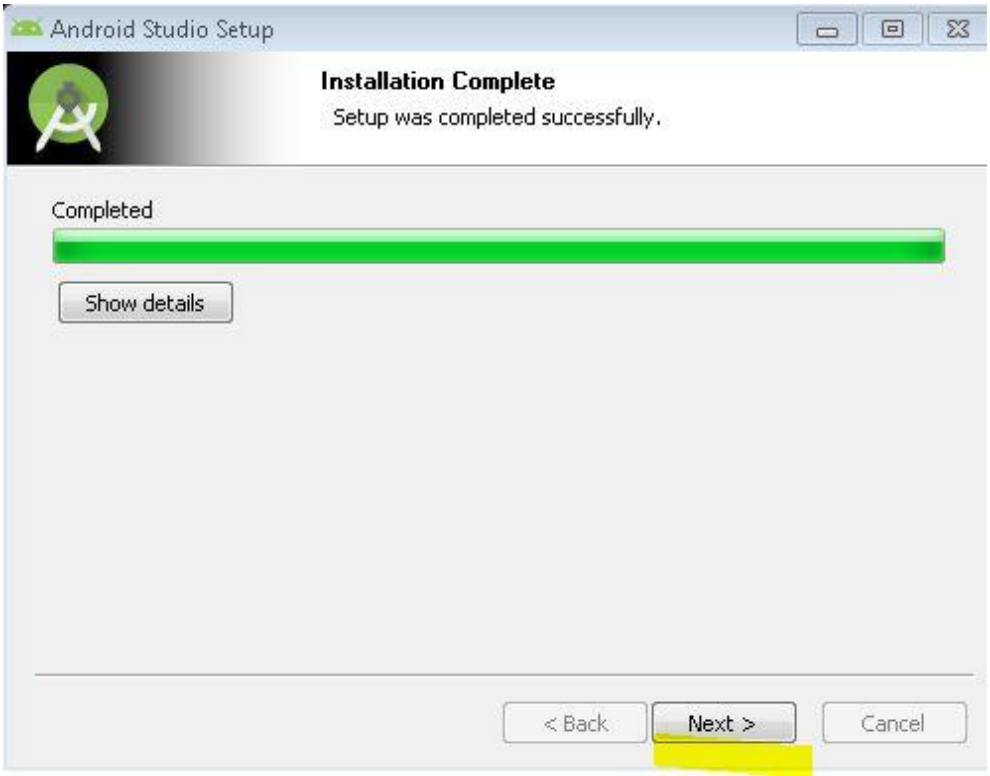



Android Studio Setup
License Agreement
Please review the license terms before installing Android Studio.
Press Page Down to see the rest of the agreement.
To get started with the Android SDK, you must agree to the following terms and conditions.
This is the Android SDK License Agreement (the "License Agreement").
1. Introduction
1.1 The Android SDK (referred to in the License Agreement as the "SDK" and specifically including the Android system files, packaged APIs, and SDK library files and tools , if and when they are made available) is licensed to you subject to the terms of the License Agreement. The License Agreement forms a legally binding contract between you and
If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install Android Studio.
< Back
I Agree
Cancel

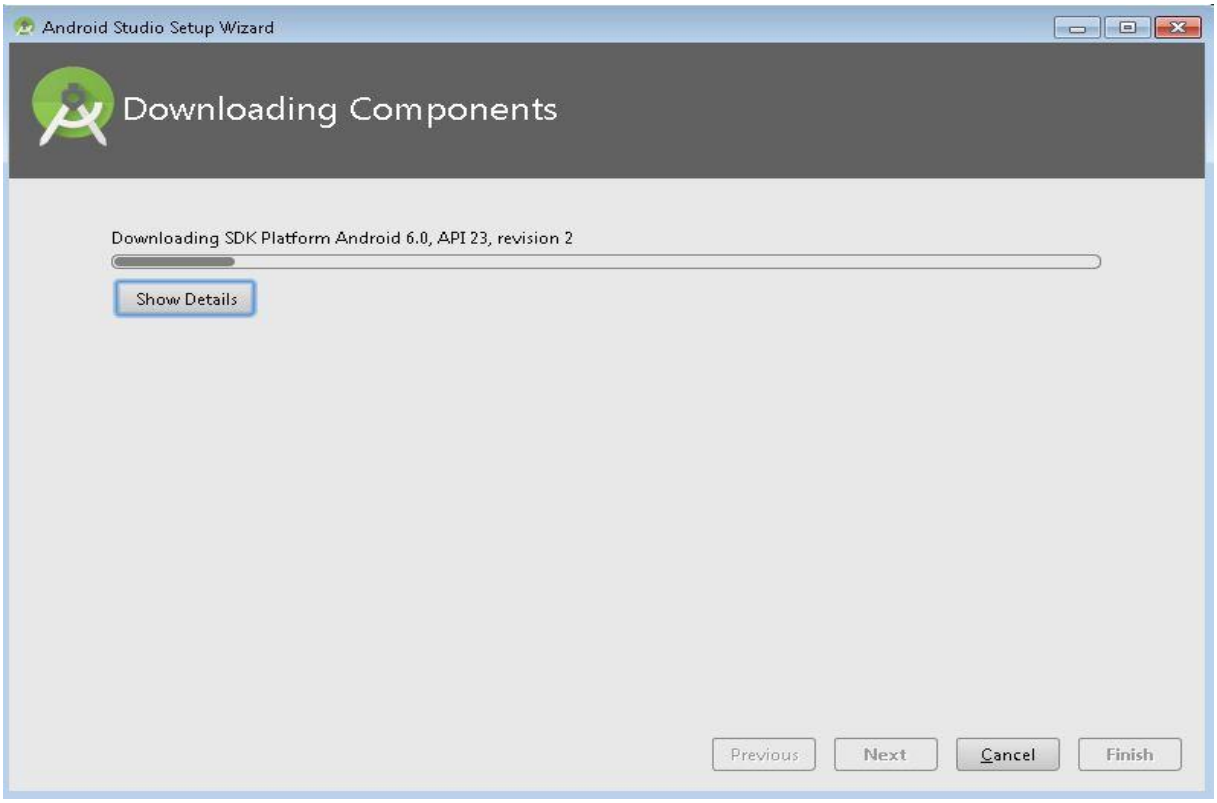

Android Studio Setup
Configuration Settings
Install Locations
Android Studio Installation Location
The location specified must have at least 500MB of free space.
Click Browse to customize:
C:\Program Files\Android\Android Studio
Browse..
Android SDK Installation Location
The location specified must have at least 3.2GB of free space.
Click Browse to customize:
D:\Android\sdk
Browse...
< Back
Next >
Cancel

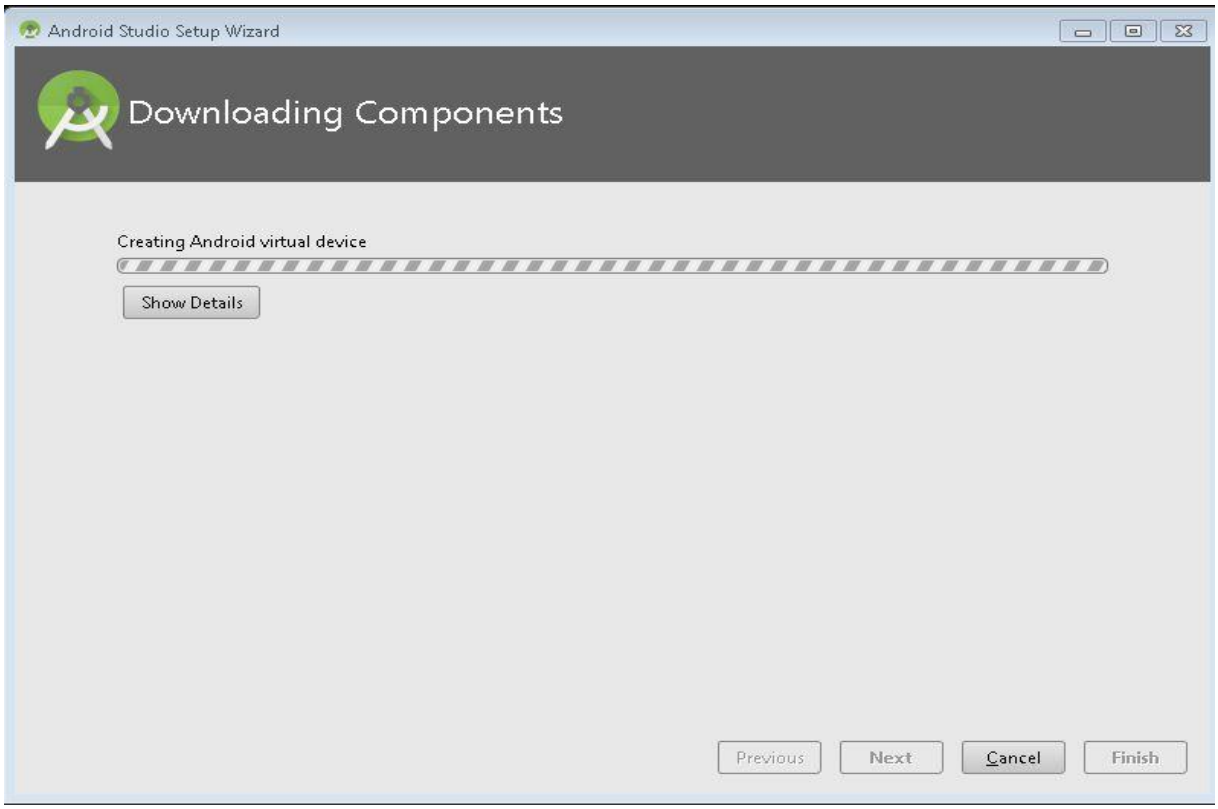

Android Studio Setup
Choose Start Menu Folder
Choose a Start Menu folder for the Android Studio shortcuts.
Select the Start Menu folder in which you would like to create the program's shortcuts. You can also enter a name to create a new folder.
Android Studio
Accessories
Administrative Tools
AMD Catalyst Control Center
DAEMON Tools Pro
ESET
Games
Google Chrome
Java
Java Development Kit
Logitech
Maintenance
Do not create shortcuts
< Back
Install
Cancel

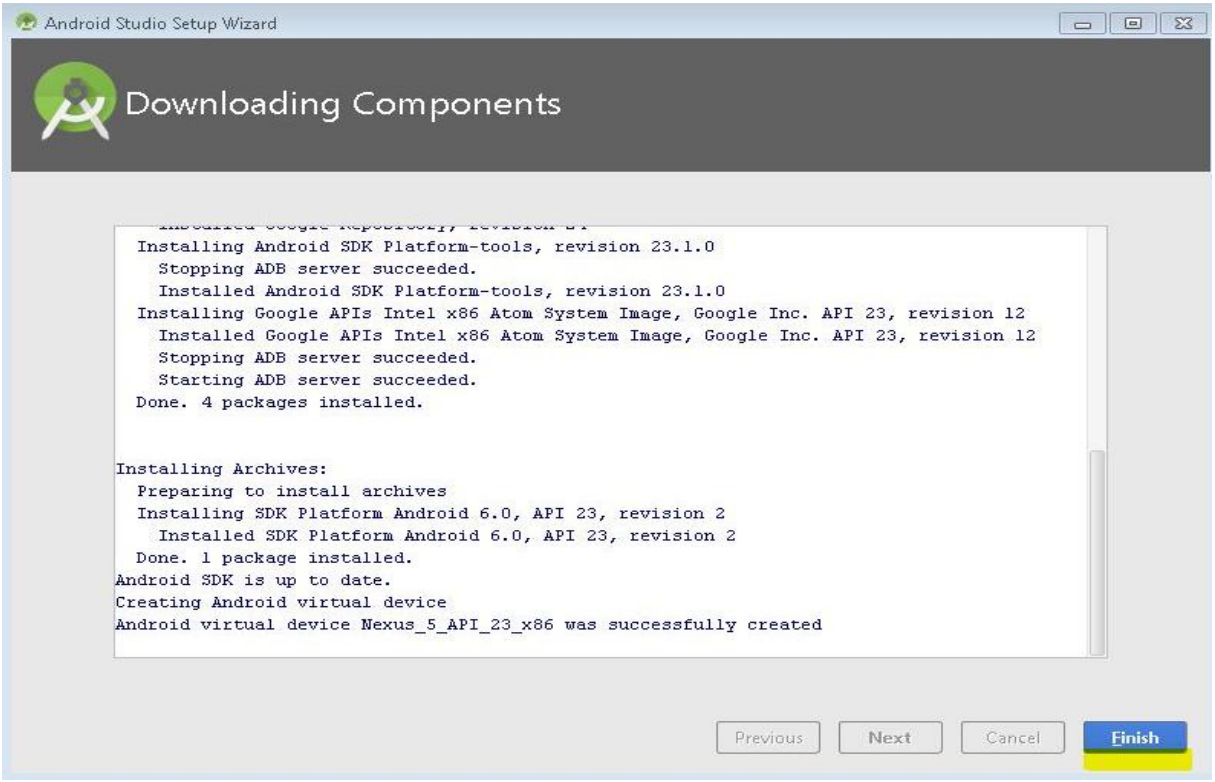

Android Studio Setup
Installing
Please wait while Android Studio is being installed.
Extract: lldb-server... 10%
Show details
< Back
Next >
Cancel


Android Studio Setup
Installation Complete
Setup was completed successfully.
Completed
Show details
< Back
Next >
Cancel

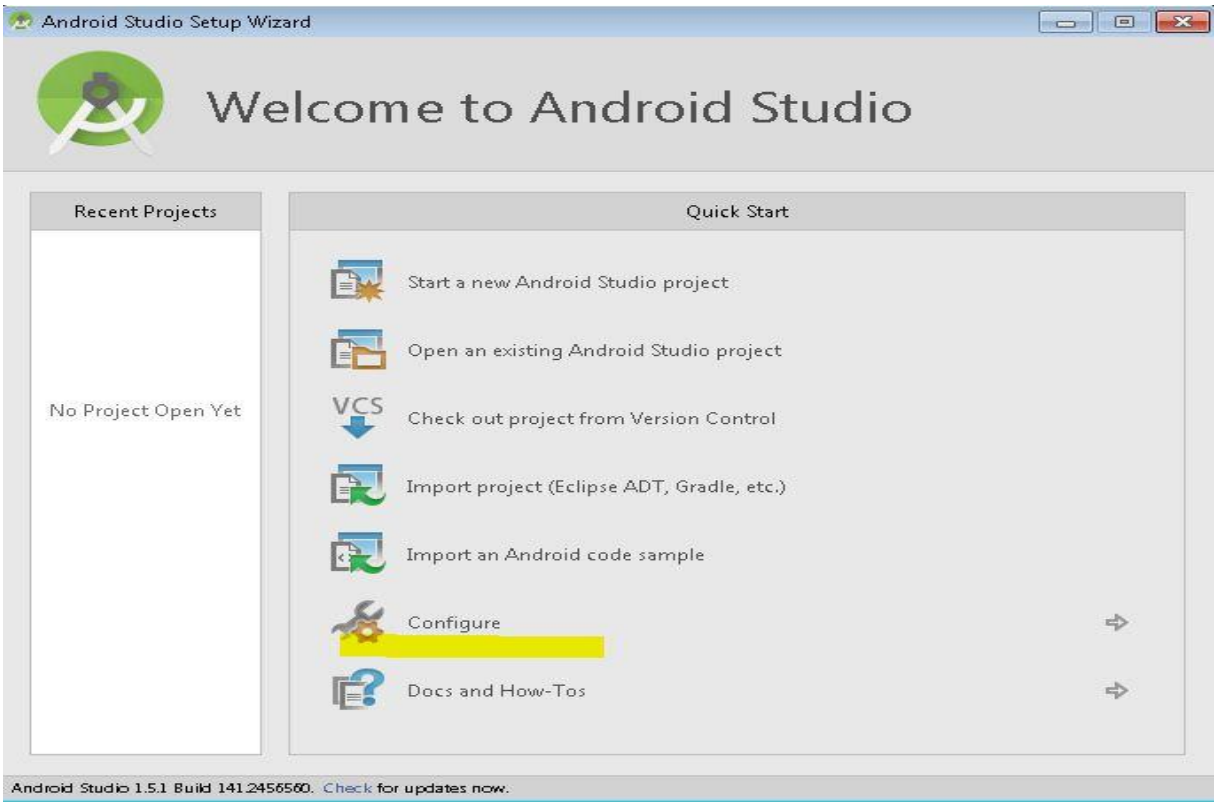

Android Studio Setup
Android
Studio
Completing Android Studio Setup
Android Studio has been installed on your computer.
Click Finish to close Setup.
Start Android Studio
< Back
Finish
Cancel

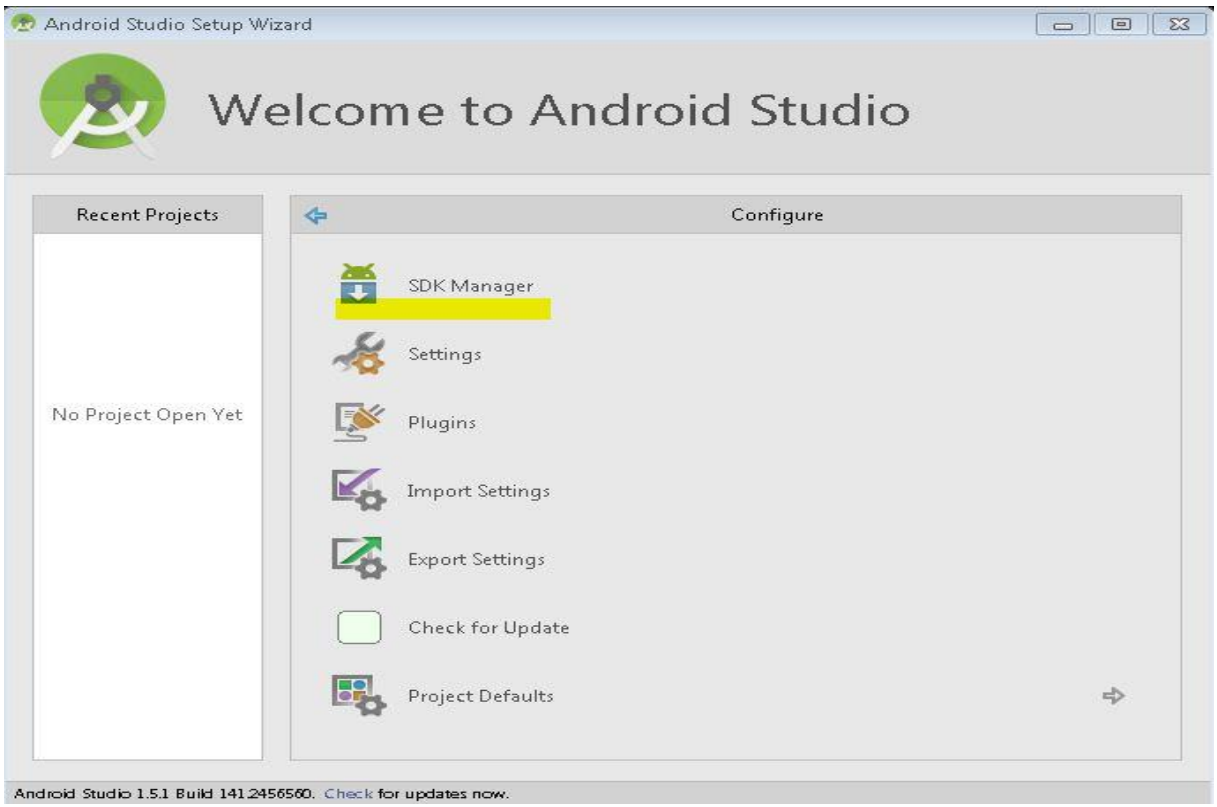

Android Studio Setup Wizard
Downloading Components
Downloading SDK Platform Android 6.0, API 23, revision 2
Show Details
Previous
Next
Cancel
Finish

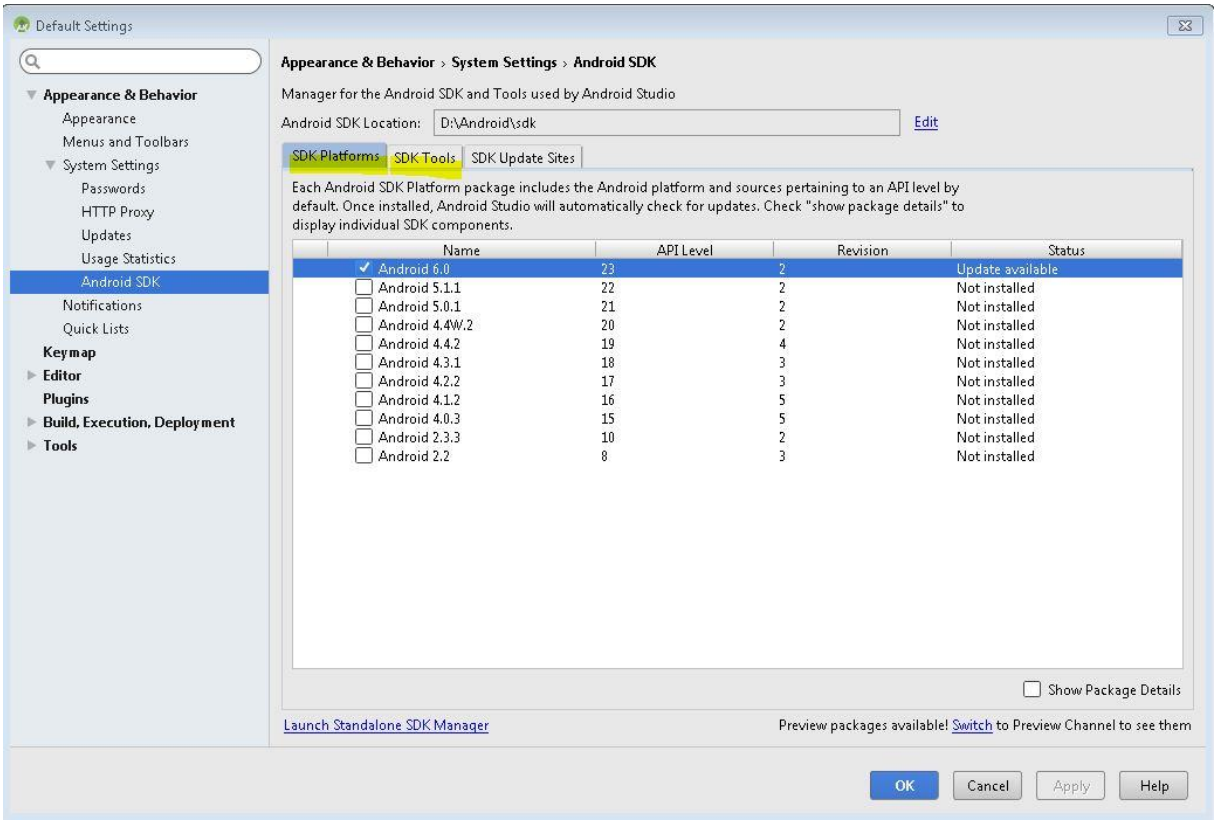

Android Studio Setup Wizard
Downloading Components
Creating Android virtual device
Show Details
Previous
Next
Cancel
Finish



Заходимо в Configure-> SDK Manager (SDK Manager, - це не частина Android Studio, а утиліта з Android SDK) і встановлюємо необхідні пакети.







Емулятор у Android Studio є досить повільний, тому в якості альтернативного варіанту можна використовувати Genymotion (https://www.genymotion.com/).

Завдання для виконання

Відповідно до поданого в теоретичних відомостях матеріалу встановіть та налаштуйте Java Development Kit та Android Studio.

Вимоги до звіту

В звіт по роботі включіть екранні форми, що відображають завершення процесу встановлення Java Development Kit та Android Studio.

Контрольні запитання

1. Для чого використовується емулятор ОС Android?
2. Які середовища та плагіни необхідно встановити для програмування додатків під ОС Android?
3. Для чого використовують Android Studio?
4. Опишіть процес встановлення JDK. Які функції виконує JDK?
5. Як налаштувати Android Studio ?

ПРАКТИЧНА РОБОТА №2

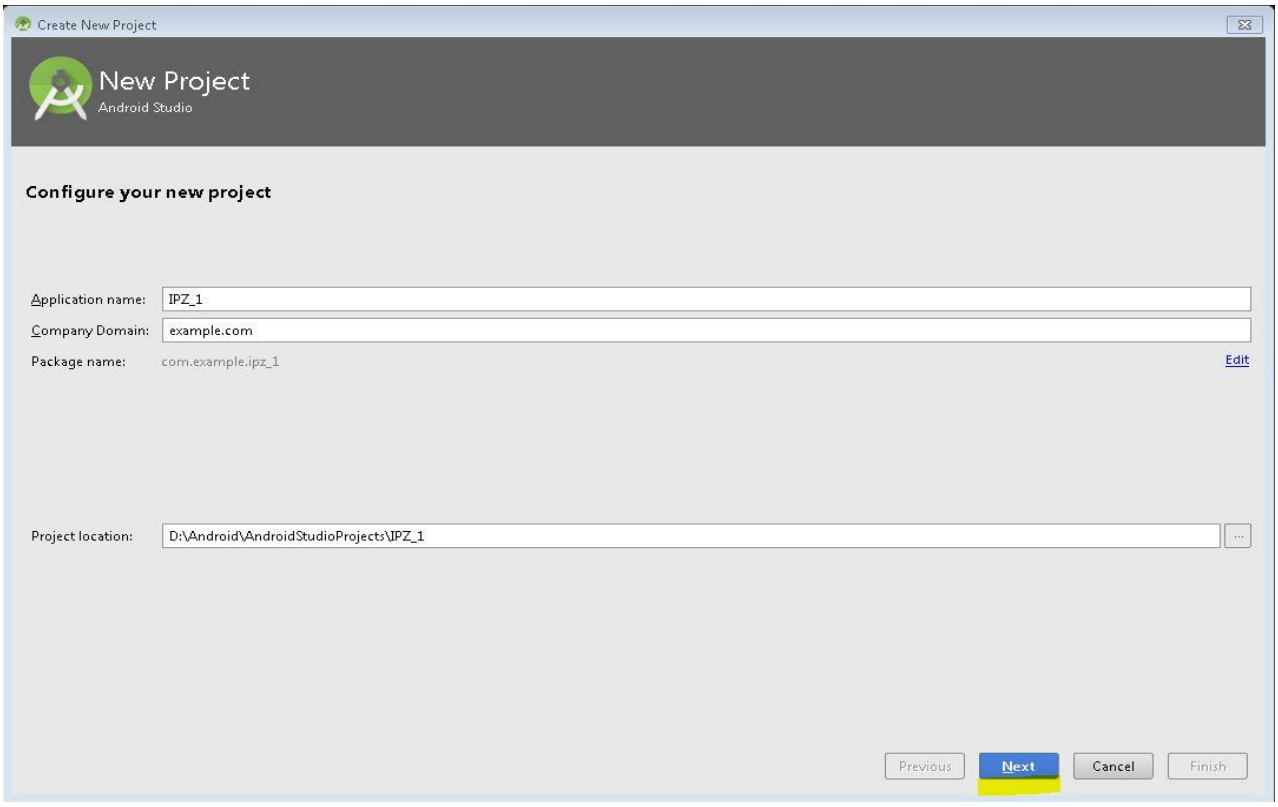
Тема роботи: Створення першої програми в Android Studio для ОС Android.

Мета роботи: Ознайомитися з процесом написання програмних додатків для мобільних пристроїв, що працюють на базі ОС Android. Освоїти принципи додавання в програму елементів користувацького інтерфейсу та особливості обробки їх подій.

Теоретичні відомості

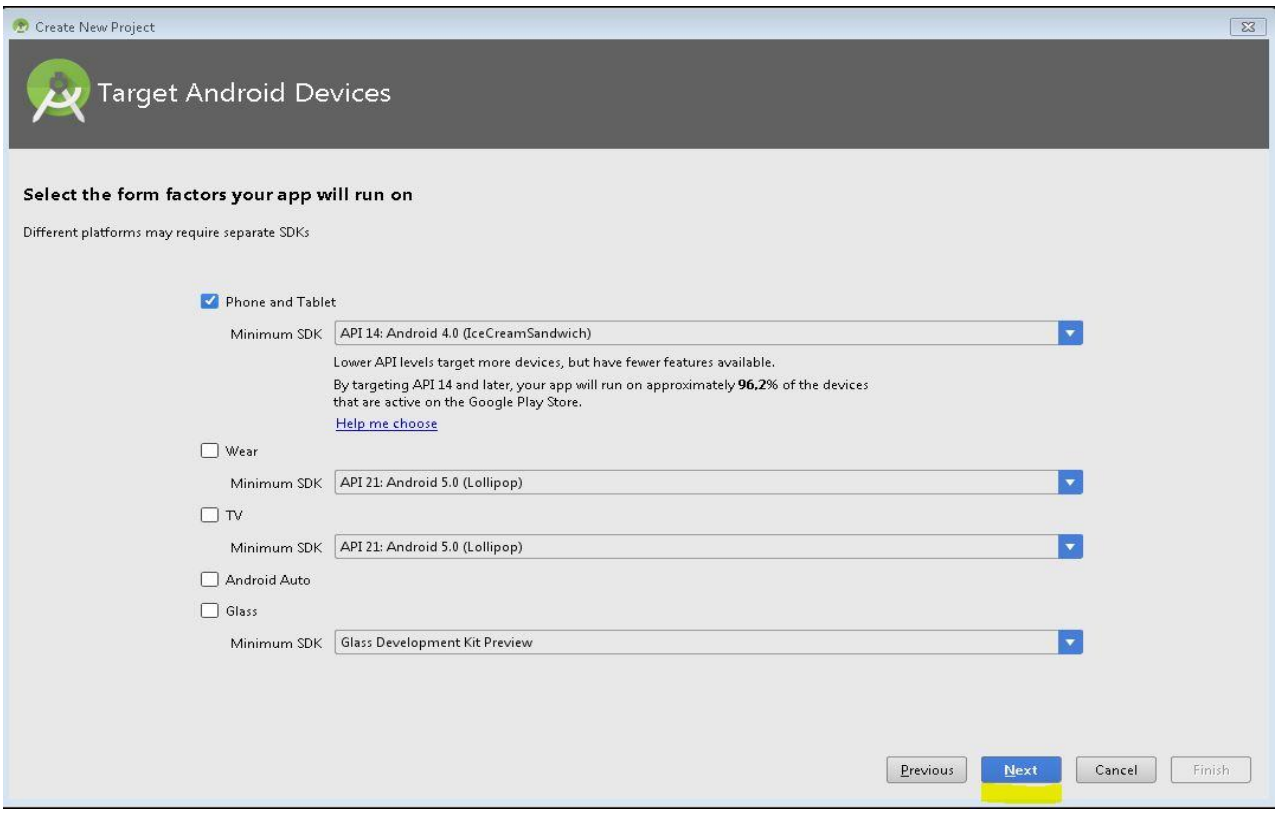
Створення проекту в Android Studio

Для початку роботи із Android Studio необхідно створити новий проект (Start a new Android Studio project). На екранній формі з'явиться вікно вибору параметрів проекту, які необхідно заповнити відповідно до рисунку поданого нижче.

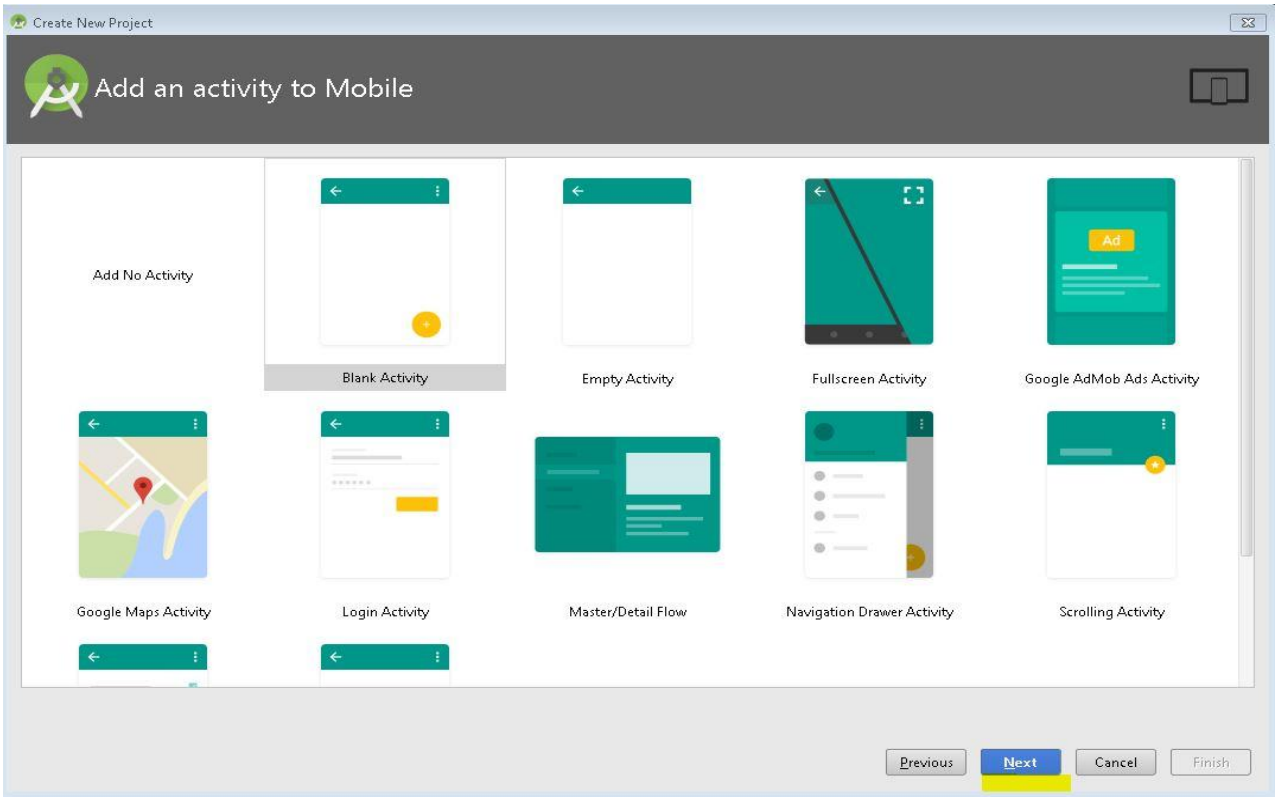


Вводимо назву проекту, назву компанії та шлях розміщення вихідних файлів. В якості "Application Name" задаємо своє прізвище латинськими буквами.

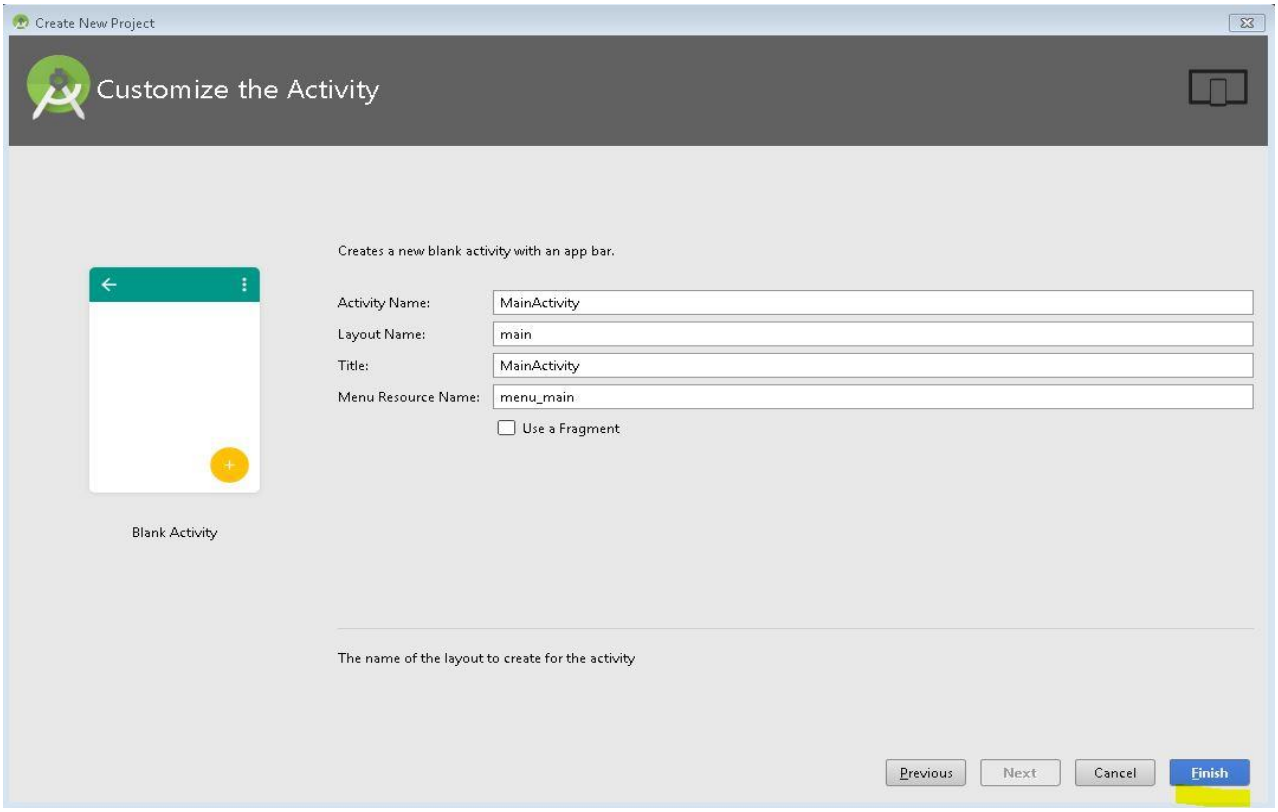
На наступному рисунку в полі Minimum SDK вибираємо API, яке охоплює велику кількість пристроїв (вказано у відсотках).



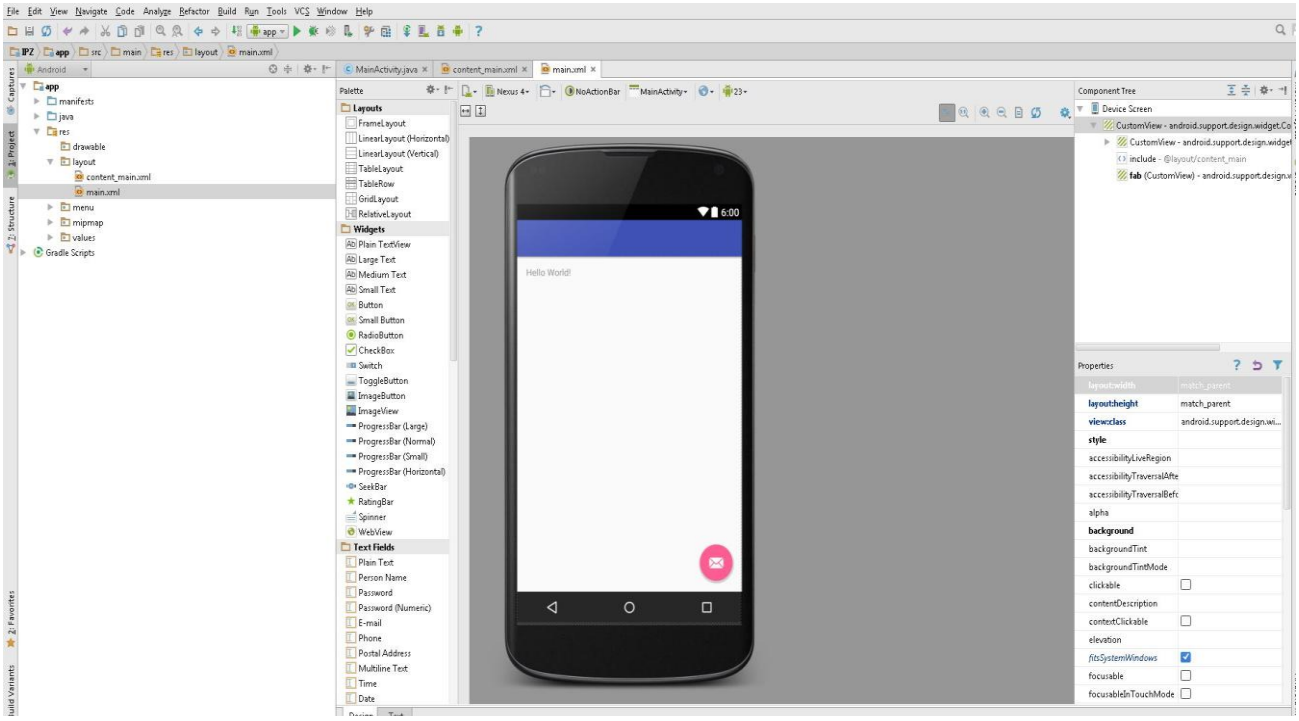
На наступному етапі вибираємо тип екрану.



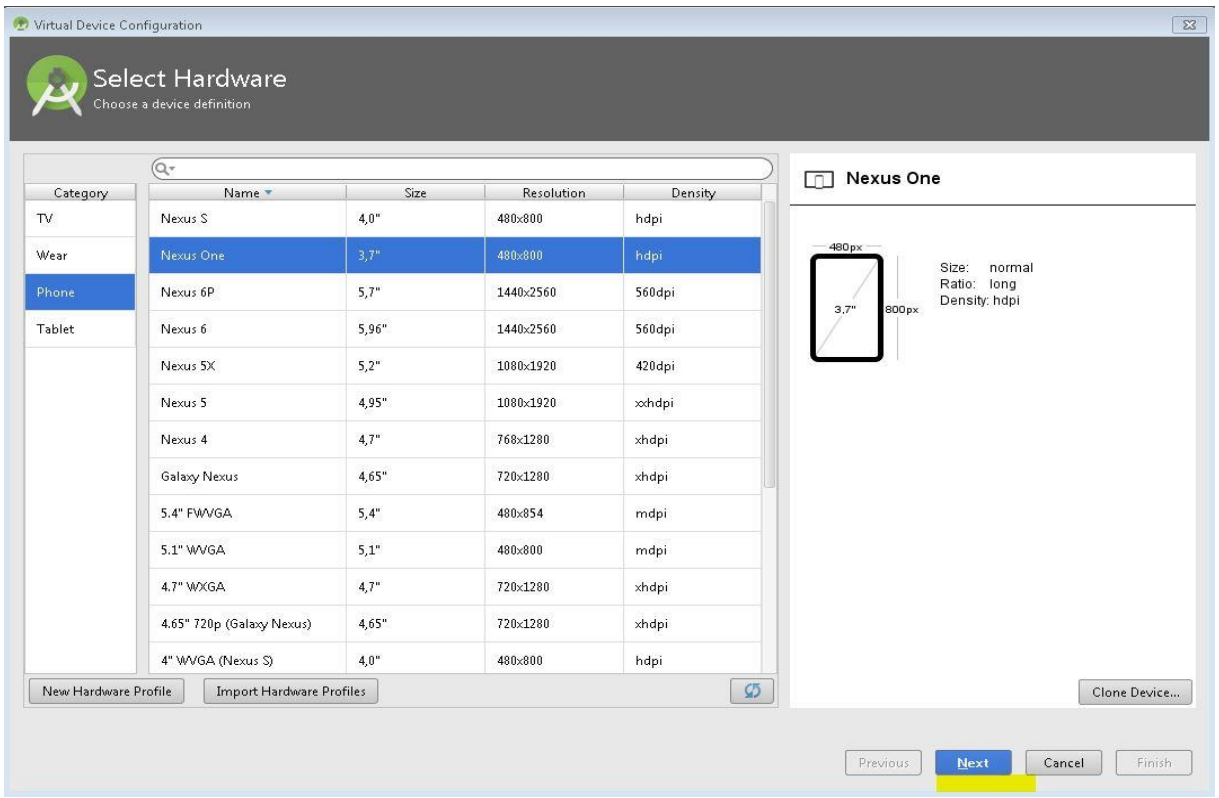
З’явиться вікно, в якому необхідно залишити все без змін (або ж змінити Layout name) і натиснути Finish.



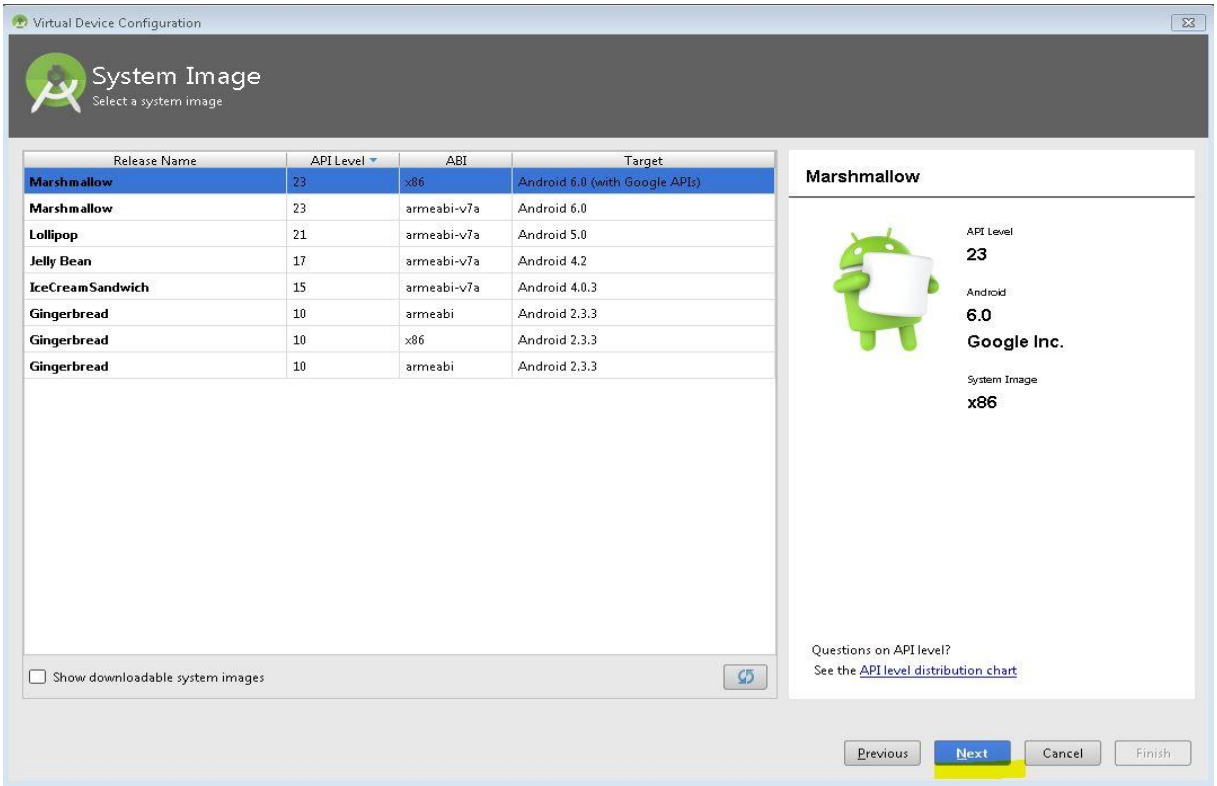
Створений проект відразу є працездатний мобільним додатком, який можна запустити на виконання.



Щоб запустити додаток на комп'ютері, потрібно спочатку створити і запустити віртуальний телефон (Tools / Android / AVD Manager / New).



Вибираємо необхідний API і тиснемо next.

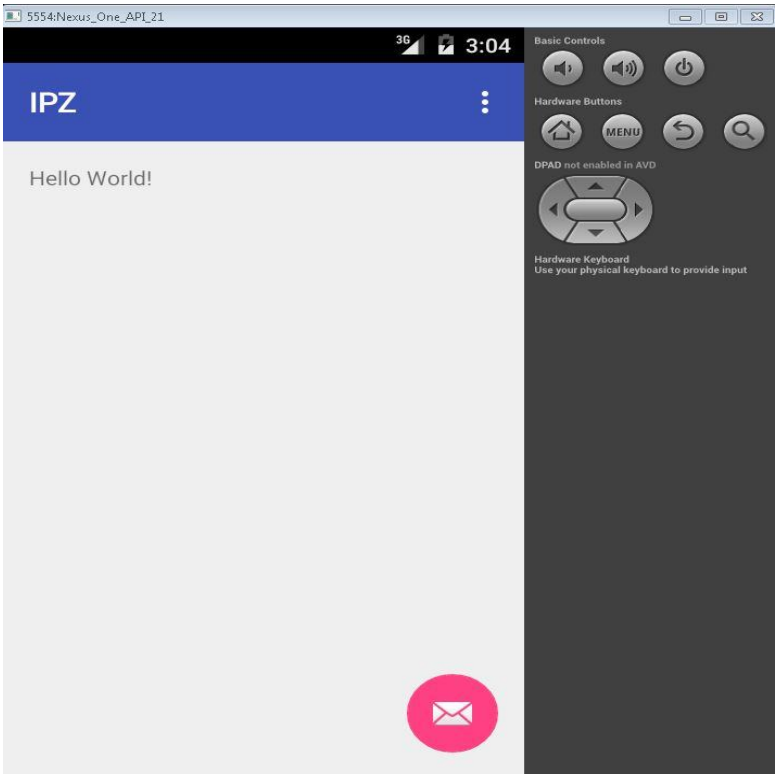


Якщо проект не містить помилок, то пройде деякий час (можливо, кілька хвилин), поки він відкомпілюється і встановиться на віртуальному телефоні. Розблокувавши екран віртуального телефону користувач бачить

запущений додаток з написом "Hello, world!". Це перший додаток створений для ОС Android.

У цей момент фаєрвол може запитувати дозвіл для виконання запитів до сервера, який необхідно схвалити.

Вікно із результатом роботи першого додатку подано нижче.

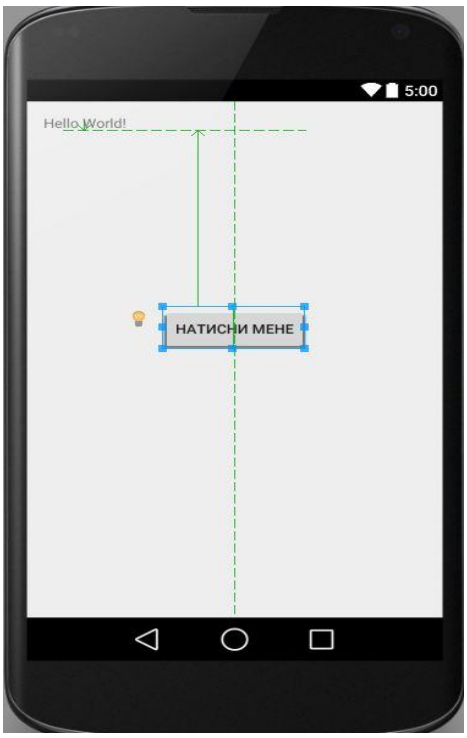


Відкомпільований і запущений проект вже містить напис "Hello, World!". Як прибрати цей напис і як замість нього додати щось інше?

Особливості редагування файлів проекту

У Project виберіть необхідний проект і натисніть app\res\layout\activity_main.xml

Відкриється вікно візуального редагування вмісту екрана додатку. З розташованого в лівій частині вікна палітри перетягніть на вікно програми кнопку "Button". Натисніть на цю кнопку і у вікні «Властивості» в правій частині екрана в поле текст введіть новий напис для неї :"Натисни мене!".



Таким же чином змініть і текст напису "Hello, World!" на свої ім'я та прізвище. Знову запустіть проект на виконання і у вікні віртуального телефону з'явиться новий варіант додатку.

У редакторі натисніть на кнопку і подивіться у вікні Властивості значення поля Id. Припустимо, там написано "@+id/button1". Це означає, що Id кнопки - "Button1". За цим Id до неї можна звертатися з програми.

Аналогічно, натиснувши на напис, можна дізнатися і його ID. Припустимо, в поле Id надпису записано «@+id/textView1». Значить Id надпису - “textView1”.

В Project для свого проекту виберіть SRC \ MainActivity.java

MainActivity.java відкриється у вікні редактора.

Щоб працювати з текстовим написом і кнопкою, для початку потрібно створити поля (змінні) цих типів. У Java для Android тип даних для кнопок називається Button, а тип даних для написів - TextView.

Додайте в клас MainActivity поле кнопки і поле текстового рядка, придумавши для них будь-які назви, наприклад katya і olesya:

TextView katya; Button olesya;

Android Studio підкреслить TextView і Button як помилки. Щоб усунути помилку, додайте в програму import обох типів даних.

Тепер у методі onCreate (він буде запускатися при ініціалізації вікна програми) додайте наступні два рядки:

katya = (TextView) findViewById (R.id.textView1);

olesya = (Button) findViewById (R.id.button1);

Сенс цих двох рядків в тому, що по впізнаним нами раніше Id текстового рядка і кнопки (textView1 і button1) ми знаходимо їх у вікні програми і запам'ятовуємо в спеціально для цього створених нами полях katya і olesya.

Тепер залишається тільки призначити обробник для події натискання на кнопку.

Для цього в цьому ж методі onCreate додайте рядок:

olesya.setOnClickListener (obrobka);

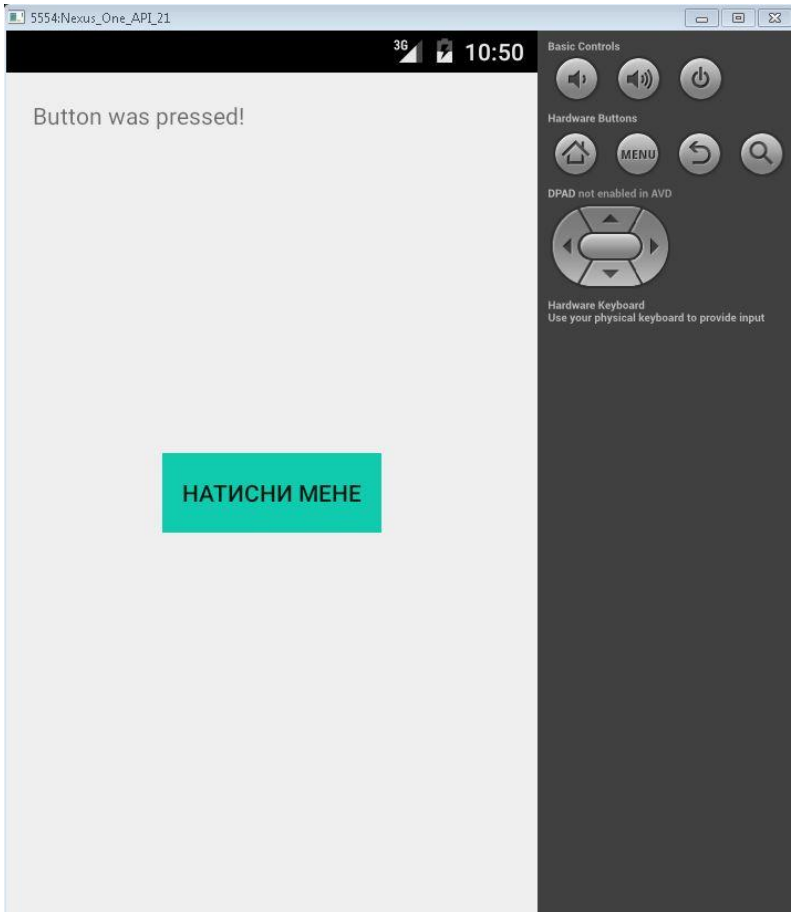
В клас додайте новий екземпляр класу OnClickListener наступним чином:

```
OnClickListener obrabotka = new OnClickListener () {
public void onClick (View v) {
katya.setText ("Button was pressed!");
}
};
```

Тут дія katya.setText ("Button was pressed!") буде виконуватися при натисканні кнопки. Замість цієї дії можна за аналогією прописати в якості реакції на натискання кнопки будь-яку іншу дію.

Android Studio підкреслить OnClickListener як помилку і потрібно його імпортувати в програму.

Так само доведеться імпортувати тип даних View, підвівши до нього мишку в Android Studio. Після цього помилок не залишиться і можна знову компілювати і запускати додаток. Цього разу після натискання на кнопку текст з "КН" зміниться на "Button was pressed!"



Якщо у вас є планшет або телефон, що працює під управлінням операційної системи Android, то спробуйте встановити туди написаний додаток і запустити його. Для цього з папки bin вашого проекту візьміть файл з ім'ям проекту та розширенням ".apk". Це стандартне розширення для програм в операційній системі Android. Самостійно розберіться або почитайте в Google, як встановлювати програму з apk-файлу.

Завдання для виконання

Створіть свою першу програму в Android Studio для ОС Android у відповідності до теоретичних відомостей поданих вище.

Вимоги до звіту

В звіт по роботі включіть текст програми, що реалізовує описане в хід роботах завдання та екранні форми, що відображають їх виконання.

Контрольні запитання

1. Які мови програмування підтримуються для розробки додатків під ОС Android?
2. Яка з IDE для розробки додатків Android найпопулярніша?
3. Чи достатньо проведення тестування програми на емуляторі?
4. Для яких платформ можуть бути написані додатки Android?
5. Опишіть процес встановлення додатку на пристрій із ОС Android.

ПРАКТИЧНА РОБОТА №3

Тема роботи: Робота із активностями, кольором та локалізацією в ОС Android.

Мета роботи: Отримати досвід роботи із процесамм відстеження станів активності, використанням значень рядків і кольорів та зміною локалізації у додатку.

Теоретичні відомості

При створенні екранів графічного інтерфейсу користувача успадковується клас Activity і використовуються View для взаємодії з користувачем.

Кожна активність - це екран (за аналогією з формою), який додаток може показувати користувачам. Чим складніший додаток, тим більше екранів (активностей) буде потрібно. При створенні додатку потрібно, як мінімум, початковий (головний) екран, який забезпечує основу користувальницького інтерфейсу. При необхідності цей інтерфейс доповнюється другорядними активностями, що призначені для введення інформації, її виведення та надання додаткових можливостей. Запуск (або повернення з) нової активності призводить до «переміщення» між екранами користувацького інтерфейсу.

Більшість активностей проектуються таким чином, щоб використовувати весь екранний простір, але можна також створювати напівпрозорі або плаваючі діалогові вікна.

Життєвий цикл активності

Для створення нової активності успадковується клас Activity. Всередині реалізації класу необхідно визначити користувальницький інтерфейс і реалізувати необхідний функціонал. Базовий каркас для нової активності показаний нижче:

```
package com.example.myapplication;
import android.app.Activity;
import android.os.Bundle;
public class MyActivity extends Activity {
/** Викликається при створенні Активності */
@Override
public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
}}
```

Базовий клас Activity являє собою порожній екран, тому перше, що потрібно зробити, це створити користувальницький інтерфейс за допомогою View і розмітки (Layout).

View - це елементи UI, які відображають інформацію і забезпечують взаємодію з користувачем. Android надає кілька класів розмітки (Layout), що

також називаються View Groups, які можуть містити всередині себе кілька View, для створення користувацького інтерфейсу програми.

Щоб призначити користувальницький інтерфейс для активності, всередині обробника onCreate використовується метод setContentView:

```
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
TextView textView = new TextView(this);
setContentView(textView);
}
```

У цьому прикладі як UI для активності виступає об'єкт класу TextView.

При створенні реальних додатків частіше застосовується метод проектування, що використовує ресурси програми, відокремлені від коду. Такий підхід дозволяє створювати додатки, що забезпечують високу якість реалізації UI, не залежно від умов роботи програми: додатки пропонують зручний для користувача мовний інтерфейс (залежить від локалізації), слабо залежать від вирішення і розмірів екрану і т. д.). Найголовніше, що така адаптація додатків до нових умов не вимагає змін в коді програми, потрібно тільки забезпечити необхідні ресурси (картинки, локалізовані рядки і т.д.). Стандартний для Android підхід показаний нижче:

```
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main); }
```

Для використання активності в додатку її необхідно зареєструвати в файлі маніфесті шляхом додавання елемента <activity> всередині вузла <application>, в іншому випадку її неможливо буде використовувати. Нижче показано, як створити елемент <activity> для активності MyActivity:

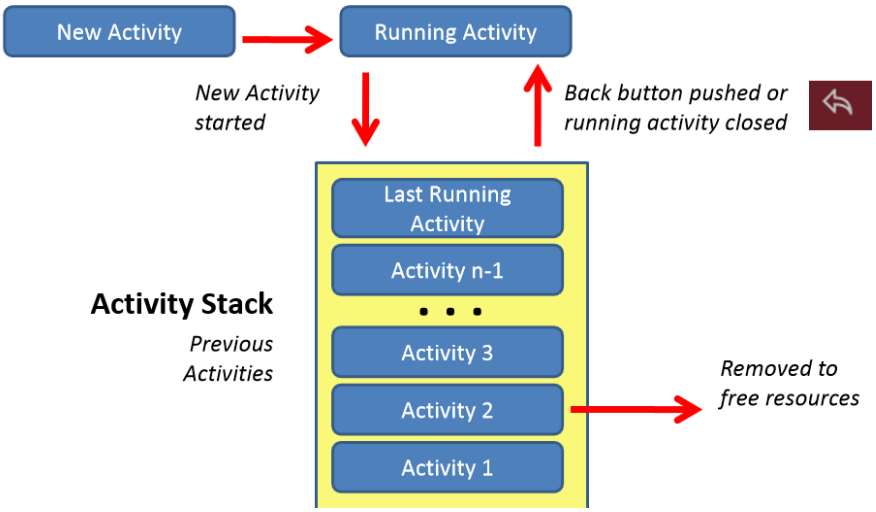
```
‹activity android:label="@string/app_name"
android:name=".MyActivity">
</activity>
```

У тезі <activity> можна додавати елементи <intent-filter> для вказівки Намірів (Intent), які активність буде відстежувати. Кожен «фільтр намірів» визначає одну або кілька дій (action) і категорій (category), які підтримуються активністю. Важливо знати, що активність буде доступна з головного меню запуску додатків тільки у випадку, якщо в маніфесті для неї вказано <intent-filter> для дії MAIN і категорії LAUNCHER, як показано у прикладі:

```
<activity android:label="@string/app_name"
android:name=".MyActivity">
<intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
```

Для створення додатків, які будуть правильно керувати ресурсами, що надають користувачеві зручний інтерфейс, важливо добре розуміти життєвий цикл активності. Це пов'язано з тим, що додатки Android не можуть контролювати свій життєвий цикл, ОС сама керує всіма процесами, і як наслідок активностями в середині них. При цьому, стан активності допомагає ОС визначити пріоритет батьківського додатку для цієї активності (Application). А пріоритет додатку впливає на те, з якою ймовірності його робота (і робота дочірніх активностей) буде перервана системою.

Стан кожної активності визначається її позицією в стеку (LIFO) активностей, запущених в даний момент. При запуску нової активності представлений нею екран розміщується на вершину стека. Якщо користувач натискає кнопку «назад» або ця активність закривається іншим чином, на вершину стека переміщається (і стає активною) активність, що знаходились нижче. Даний процес показаний на поданому нижче рисунку.



На пріоритет додатку впливає його пріоритетна активність. Коли диспетчер пам'яті ОС вирішує, яку програму закрити для звільнення ресурсів, він враховує інформацію про стан активності в стеку для визначення пріоритету додатку.

Стан активностей та відстеження його змін

Активності можуть знаходитися в одному з чотирьох станів:

• Активна (Active). Активність знаходиться на передньому плані (на вершині стека) і має можливість взаємодіяти з користувачем. Android намагатиметься зберегти її працездатність, при необхідності перериваючи роботу інших активних, що знаходяться на нижчих позиціях в стеку для надання необхідних ресурсів. При виході на передній план іншої активності робота даної активності буде припинена або зупинена.

• Призупинена (Paused). Активність може бути видима на екрані, але не може взаємодіяти з користувачем: в цей момент вона призупинена. Це трапляється, коли на передньому плані знаходяться напівпрозорі або плаваючі (наприклад, діалогові) вікна. Робота призупиненої активності може

бути припинена, якщо ОС необхідно виділити ресурси активності переднього плану. Якщо активність повністю зникає з екрану, вона зупиняється.

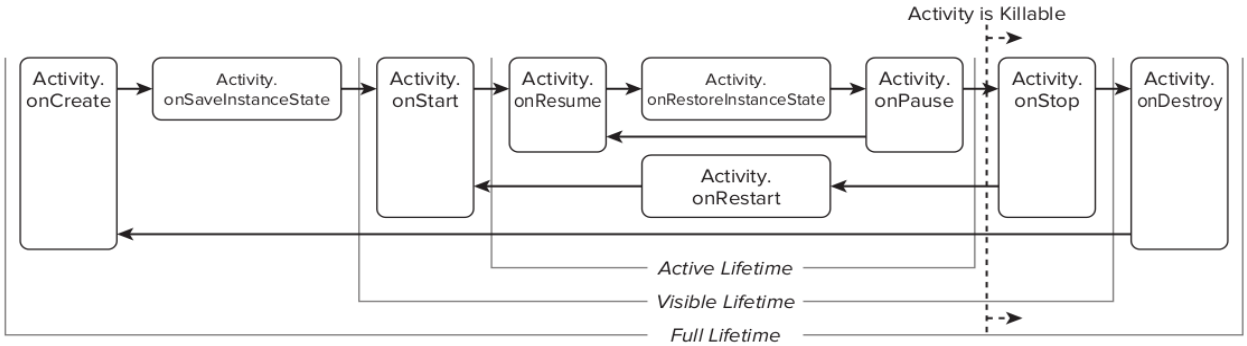
• Зупинена (Stopped). Активність невидима, вона знаходиться в пам'яті, зберігаючи інформацію про свій стан. Така активність стає кандидатом на передчасне закриття, якщо системі буде потрібно пам'ять для чогось іншого. При зупинці Активності розробнику важливо зберегти дані і поточний стан користувача інтерфейсу (стан полів введення, позицію курсора і т.д.). Якщо активність завершує свою роботу або закривається, вона стає неактивною.

• Неактивна (Inactive). Коли робота активності завершена, і перед тим, як вона буде запущена, дана активність знаходиться в неактивному стані. Такі активності видаляються з стека і повинні бути (пере) запущені, щоб їх можна було використовувати.

Зміна стану додатка - недетермінований процес, що керується виключно менеджером пам'яті Android. При необхідності Android спочатку закриває додатки, що містять неактивні активності, потім зупинені і, в крайньому випадку, припинені.

Для забезпечення повноцінного інтерфейсу додатку, зміни його стану повинні бути непомітні для користувача. Міняючи свій стан з призупиненого на зупинений або з неактивного на активний, активність не повинна зовні змінюватися. При зупинці або призупиненні роботи активності розробник повинен забезпечити збереження стану активності, щоб її можна було відновити при виході активності на передній план. Для цього в класі Activity міститься обробники подій, перевизначення яких дозволяє розробнику відслідковувати зміну станів активності.

Обробники подій класу Activity дозволяють відслідковувати зміни станів відповідного об'єкта Activity під час усього життєвого циклу, що відображає рисунок поданий нижче.



Показаний приклад із заглушками для таких методів - обробників подій:

```
package com.example.myapplication;
import android.app.Activity;
import android.os.Bundle;
public class MyActivity extends Activity {
// Викликається при створені Активності
@Override
```

```
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
// Ініціалізує Активність.
}
// Викликається після завершення метода onCreate
// Використовується для відновлення стану UI
@Override
public void onRestoreInstanceState(Bundle savedInstanceState) {
        super.onRestoreInstanceState(savedInstanceState);
        // Відновити стан UI з об'єкта savedInstanceState.
        // Даний об'єкт також був переданий методу onCreate.
}
// Викликається перед тим, як Активність знову стає видимою
@Override
public void onRestart(){
        super.onRestart();
        // Відновити стан UI з урахуванням того,
        // що дана Активність вже була видима.
}
// Викликається коли Активність стала видима
@Override
public void onStart(){
        super.onStart();
        // Проробити необхідні дії для
        // Активності, видимої на екрані
}
// Повинен викликатись на початку видимого стану.
// Насправді Android викликає даний обробник тільки
// для Активностей, відновлених з неактивного стану
@Override
public void onResume(){
        super.onResume();
        // Відновити призупинені оновлення UI,
        // потоки и процеси, «заморожені», коли
        // Активність була в неактивному стані
}
// Викликається перед виходом з активного стану,
// дозволяючи зберегти стан в об'єкті savedInstanceState
@Override
public void onSaveInstanceState(Bundle savedInstanceState) {
        // Об'єкт savedInstanceState буде в подальшому
        // переданий методам onCreate і onRestoreInstanceState
        super.onSaveInstanceState(savedInstanceState);
}
```

```
// Викликається перед виходом з активного стану
@Override
public void onPause(){
        // «Заморозити» оновлення UI, потоки чи
        // «трудомісткі» процеси, непотрібні, коли Активність
        // не на передньому плані
        super.onPause();
}
// Викликається перед виходом з видимого стану
@Override
public void onStop(){
        // «Заморозити» оновлення UI, потоки чи
        // «трудомісткі» процеси, непотрібні, коли Активність
        // не на передньому плані.
        // Зберегти всі дані і зміни в UI, так як
        // процес може бути в будь-який момент вбитий системою
        super.onStop();
}
// Викликається перед знищенням активності
@Override
public void onDestroy(){
        // Звільнити всі ресурси, включаючи працюючі потоки,
        // з'єднання з БД і т. д.
        super.onDestroy(); } }
```

Робота із ресурсами

Незалежно від використовуваного середовища розробки, досить розумно відокремлювати використовувані додатком ресурси від коду. Зовнішні ресурси легше підтримувати, оновлювати і контролювати. Така практика також дозволяє описувати альтернативні ресурси для підтримки додатком різних пристроїв та реалізовувати локалізацію додатків.

Додатки Android використовують різноманітні ресурси із зовнішніх (по відношенню до коду) файлів, від простих (рядки і кольору) до більш складних (зображення, анімації, візуальні стилі). Надзвичайно корисно також відокремлювати від коду такі важливі ресурси, як розмітки екранів (Layout), використовувані в активності. Android автоматично вибирає найбільш підходящі варіанти з дерева ресурсів додатків, що містять різні значення для різних апаратних конфігурацій, мов і регіонів, не вимагаючи при цьому жодного рядка коду.

Ресурси програми зберігаються в каталозі res в дереві каталогів проекту. Плагін ADT автоматично створює каталог res з підкаталогами values, layout і drawable- *, в яких зберігаються, відповідно: строкові константи, розмітка за замовчуванням іконка програми.

Для дев'яти головних типів ресурсів використовуються різні підкаталоги каталогу res, це:

• прості значення,
• зображення,
• розмітка,
• анімація,
• стилі,
• меню,
• налаштування пошуку,
• XML,
• «сирі» дані.

При збірці пакету .apk ці ресурси максимально ефективно компілюються і включаються в пакет.

Для роботи з ресурсами всередині коду плагін ADT автоматично генерує файл класу R, що містить посилання на всі ресурси. Імена файлів ресурсів можуть містити тільки латинські букви в нижньому регістрі, підкреслення і крапки.

Android підтримує наступні типи значень: рядки, кольори, розміри і масиви (рядкові та цілочисельні). Ці дані зберігаються у вигляді XML-файла в каталозі res/values. Нижче показано приклад подібного файлу:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<string name="app_name">To Do List</string>
<color name="app_background">#FF0000FF</color>
<dimen name="default_border">5px</dimen>
<array name="string_array">
    <item>Item 1</item>
    <item>Item 2</item>
    <item>Item 3</item>
</array>
<array name="integer_array">
    <item>3</item>
    <item>2</item>
    <item>1</item>
</array>
</resources>
```

У прикладі містяться всі доступні типи простих значень, але насправді кожен тип ресурсів зберігається в окремому файлі, наприклад, res/values/arrays.xml містить масиви, а res/values/strings.xml - строкові константи.

Строкові константи визначаються за допомогою тега <string>, як показано на прикладі:

```
<string name = "greeting_msg"> Привіт! </ string>
```

Для виділення тексту в рядках можна використовувати HTML-теги *<b>*, *<i>* і *<u>*.

приклад:

<string name = "greeting_msg"> <b> Привіт </ b>, дарлінг! </ string>

При необхідності використання даного рядка в коді програми використовується вищезгаданий клас R:

String greeting = getString (R.string.greeting_msg);

Робота із кольорами, зображеннями, стилями та темами

Для опису кольорів використовується тег <color>. Значення кольору вказуються в шістнадцятковому вигляді в одному з наступних форматів:

• # RGB
• # ARGB
• # RRGGBB
• # AARRGGBB

У прикладі показано опис напівпрозорого червоного кольору і непрозорого зеленого:

<color name="transparent_red">#77FF0000</color>

<color name="opaque_green">#0F0</color>

Посилання на розміри найчастіше зустрічаються всередині ресурсів зі стилями і розміткою, наприклад, при вказівці товщини рамки або величини шрифту. Для опису розмірів використовується тег <dimen> із зазначенням виду розмірності:

• px - реальні екранні пікселі,
• in - фізичні дюйми,
• pt - 1/72 дюйма, обчислюється з фізичного розміру екрану,
• mm - фізичні міліметри, обчислюється з фізичного розміру екрану,
• dp - «незалежні» від щільності екрану пікселі, дорівнюють одному пікселю при еталонній щільності 160 dpi; можна також вказувати як dip; найчастіше використовуються для вказання розмірів рамок і полів,
• sp - «незалежні» від масштабу пікселі, аналогічні dp, але враховують також налаштування користувацького шрифту (великий, дрібний, середній), тому рекомендуються для опису шрифтів.

Приклад опису «великого» шрифту і «стандартної» рамки:

<dimen name = "standard_border"> 5dp </ dimen>

<dimen name = "large_font_size"> 16sp </ dimen>

Стилі і теми дозволяють підтримувати єдність зовнішнього вигляду програми за допомогою атрибутів, використовуваних View, найчастіше це кольори і шрифти. Зовнішній вигляд програми легко змінюється при зміні стилів (тем оформлення) в маніфесті додатків.

Для створення стилю використовується тег *<style>* з атрибутом name, що містить елементи <item>, кожен з яких, у свою чергу, також має атрибут name, який вказує тип параметра (наприклад, колір або розмір). Всередині елемента *<item>* зберігається значення параметра:

```
<? xml version = "1.0" encoding = "utf-8"?>
<resources>
    <style name = "StyleName">
        <item name = "attributeName"> attributeValue </ item>
        [... Ще елементи <item> ...]
    </ style>
</ resources>
```

У тегу <*style*> можна вказати атрибут parent, що робить можливим «успадкування» стилів при необхідності внести в новий стиль незначні відмінності від наявного:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="NormalText">
        <item name="android:textSize">14sp</item>
        <item name="android:textColor">#111</item>
    </style>
    <style name="SmallText" parent="NormalText">
        <item name="android:textSize">8sp</item>
    </style>
</resources>
```

Ресурси Drawable містять растрові зображення. Це можуть бути складні складові ресурси, такі, як LevelListDrawables і StateListDrawables, подані у форматі XML, а також растрові зображення NinePatch. Ресурси Drawable зберігаються в каталогах res / drawable - у вигляді окремих файлів. Ідентифікаторами для таких ресурсів служать імена в нижньому регістрі (без розширення). Підтримуються формати PNG (рекомендований), JPEG та GIF.

Завдяки використанню ресурсів з розміткою (layout) розробник має можливість відокремити логіку програми від її зовнішнього вигляду. Розмітку, визначену у файлі формату XML, можна завантажити для використання в активності методом setContentView. Це реалізовано в поданому прикладі у методі onCreate:

```
setContentView (R.layout.main);
```

Кожен ресурс з розміткою зберігається в окремому файлі в каталозі res / layout. Файл використовується як ідентифікатор даного ресурсу (як звичайно, без розширення). При створенні навчального додатку у попередній роботі майстер створення нових проектів створив файл res/layout/main.xml, який піддавався редагуванню:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@color/screen_bkg_color"
    android:orientation="vertical" >
```

```
<TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="@color/view_bkg_color"
        android:text="@string/hello"
        android:textColor="@color/text_color" />
</LinearLayout>
```

Даний файл містить розмітку LinearLayout, яка є контейнером для елемента TextView, відображає вміст рядка з ім'ям hello, описану в ресурсі strings.

Анімація в Android

Android підтримує два види анімації: покрокову анімацію, послідовно виводить на екран зображення з заданою тривалістю, і анімацію, що базується на розрахунку проміжних кадрів, в цьому випадку застосовуються різні перетворення - обертання, розтягування, переміщення і затемнення. Всі ці трансформації описуються в XML-файлі в каталозі res / anim. Приклад файлу анімації, в якому цільовий елемент одночасно повертається на 270 градусів, стискається і поступово зникає:

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:interpolator="@android:anim/accelerate_interpolator">
    <rotate
        android:fromDegrees="0"
        android:toDegrees="270"
        android:pivotX="50%"
        android:pivotY="50%"
        android:startOffset="500"
        android:duration="1000" />
    <scale
        android:fromXScale="1.0"
        android:toXScale="0.0"
        android:fromYScale="1.0"
        android:toYScale="0.0"
        android:pivotX="50%"
        android:pivotY="50%"
        android:startOffset="500"
        android:duration="500" />
    <alpha
        android:fromAlpha="1.0"
        android:toAlpha="0.0"
        android:startOffset="500"
        android:duration="500" />
</set>
```

Ресурс, що описує покрокову анімацію, зберігається в каталозі res / drawable. У наступному прикладі описана анімація, що базується на послідовному відображенні шести зображень поїзда, кожне з яких (крім останнього) відображається протягом 200 мілісекунд. Зрозуміло, для використання такої анімації потрібні ресурси із зображеннями (Drawable), що знаходяться в цьому ж каталозі з іменами (як варіант, у форматі PNG) train1.png .. train6.png.

```
<animation-list
xmlns:android="http://schemas.android.com/apk/res/android"
android:oneshot="false">
<item android:drawable="@drawable/train1" android:duration="200" />
<item android:drawable="@drawable/train2" android:duration="200" />
<item android:drawable="@drawable/train3" android:duration="200" />
<item android:drawable="@drawable/train4" android:duration="200" />
<item android:drawable="@drawable/train5" android:duration="200" />
<item android:drawable="@drawable/train6" android:duration="1500" />
</animation-list>
```

Детальну інформацію про можливості анімації в Android можна знайти тут: **http://developer.android.com/guide/topics/graphics/animation.html**

Ресурси меню та зовнішні ресурси

Ресурси меню можуть використовуватися для опису як головного меню активності, так і контекстного, що з'являється при тривалому натисканні на який-небудь елемент користувацького інтерфейсу. Меню, описане у форматі XML, завантажується в додаток з допомогою методу inflate системного сервісу MenuInflater. Зазвичай це відбувається всередині методу onCreateOptionsMenu (для головного меню) або onCreateContextMenu (для контекстного меню), перевизначених в активності. Кожен екземпляр меню описується в окремому файлі XML в каталозі res / menu. Зазвичай імена файлів (без розширень) стають іменами ресурсів. Нижче наведено приклад простого ресурсу з меню, що має три пункти: Refresh, Settings і Quit:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/menu_refresh"
        android:title="Refresh" />
    <item android:id="@+id/menu_settings"
        android:title="Settings" />
    <item android:id="@+id/menu_quit"
        android:title="Quit" />
</menu>
```

Кожен з цих пунктів меню має унікальний ідентифікатор (*menu_refresh, menu_settings і menu_quit*), що дозволяє надалі оброблювачу меню визначити, який з пунктів був обраний користувачем.

Доступ до ресурсів в коді здійснюється за допомогою автоматично згенерованого класу R, точніше, його підкласів. Наприклад, клас R в нашому проекті виглядає так:

```
package com.example.helloandroidworld;
public final class R {
    public static final class attr {
    }
    public static final class color {
        public static final int screen_bkg_color=0x7f040001;
        public static final int text_color=0x7f040002;
        public static final int view_bkg_color=0x7f040000;
    }
    public static final class drawable {
        public static final int ic_launcher=0x7f020000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f050001;
        public static final int hello=0x7f050000; }}
```

Члени класів з іменами, відповідними ресурсам, є ідентифікаторами в таблиці ресурсів, а не самими екземплярами ресурсів.

Деякі методи і конструктори можуть брати в якості параметрів ідентифікатори ресурсів, в цьому випадку їх можна використовувати безпосередньо:

```
setContentView(R.layout.main);
Toast.makeText(this,R.string.awesome_error,Toast.LENGTH_LONG).show()
```

У випадку, якщо необхідний екземпляр ресурсу, потрібен доступ до таблиці ресурсів, здійснюється за допомогою екземпляра класу Resources. Цей клас містить геттери для всіх видів ресурсів, при цьому в якості параметрів використовуються ідентифікатори ресурсів з класу R:

```
// Отримуємо доступ до таблиці ресурсів
Resources r = getResources();
// і отримуємо необхідні экземпляри ресурсів
CharSequence greetingMsg = r.getText(R.string.greeting_message);
Drawable icon = r.getDrawable(R.drawable.app_icon);
int opaqueBlue = r.getColor(R.color.opaque_blue);
float borderWidth = r.getDimension(R.dimen.standard_border);
String[] stringArray = r.getStringArray(R.array.string_array);
int[] intArray = r.getIntArray(R.array.integer_array);
```

Для звернення до одного ресурсу всередині опису іншого ресурсу використовується наступна нотація:

```
attribute="@[packagename:]resourcetype/resourceidentifier"
```

Ім'я пакету використовується тільки при зверненні до ресурсів з іншого пакета, для звернення до своїх ресурсів його вказувати не потрібно. У нашому випадку таку нотацію можна побачити, наприклад, при описі елемента розмітки *TextView* у файлі res / layout / main.xml:

```
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:background="@color/view_bkg_color"
    android:text="@string/hello"
    android:textColor="@color/text_color" />
```

Аналогічним чином здійснюється доступ до системних ресурсів, як ім'я пакета при цьому вказується @android:

```
< EditText
android:id="@+id/myEditText"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@android:string/ok"
android:textColor="@android:color/primary_text_light" />
```

Зверніть увагу на атрибут android: id = "@ + id / myEditText". Такий запис дозволяє привласнити ідентифікатор вашому компоненту ресурсу (в даному випадку елементу розмітки) і надалі використовувати цей ідентифікатор для отримання примірника ресурсу.

Посилання на поточні візуальні стилі дозволяють використовувати діючі зараз атрибути стилів, замість того, щоб заново їх визначати. Для вказівки посилання на такий ресурс використовується символ ? Замість @:

```
< EditText
android:id="@+id/myEditText"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:hint="@string/edit_text_hint"
android:backgroundColor="?android:backgroundColor" />
```

Однією з основних переваг використання зовнішніх по відношенню до коду ресурсів - можливість використання механізму автоматичного вибору ресурсів. Користуючись описаним нижче механізмом, можна створювати індивідуальні ресурси для різних апаратних конфігурацій, мов, регіонів і т.д. Android під час виконання додатка сам вибере найбільш підходящі ресурси.

Для індивідуального налаштування програми доступні наступні можливості:

- MCC (Mobile Country Code) і MNC (Mobile Network Code),
- Мова і регіон. Наприклад, *en-rUS* для англійської мови в американському регіоні (маленька r - від «region»), *ua* для українського,
- Розмір екрана (*small, medium* або *large*),
- «Широкоформатність» екрана (*long* або *notlong*),

- Орієнтація екрана (*port, land* або *square*),
- Щільність пікселів на екрані (*ldpi, mdpi, hdpi або nodpi*),
- Тип сенсорного екрану (*notouch, stylus або finger*),
- Доступність клавіатури (*keysexposed, keyshidden або keyssoft*),
- Тип вводу (*nokeys, qwerty або 12key*),
- Спосіб навігації (*nonav, dpad, trackball або wheel*).

Альтернативні ресурси розташовуються в підкаталогах каталогу res, при цьому використовуються модифікатори стандартних імен підкаталогів з ресурсами. Наприклад, файл, що містить рядкові константи для англійської мови, буде розташовуватися за наступним шляхом: res/values-en/strings.xml модифікаторів в даному випадку є суфікс -en, доданий до імені каталога values.

Завдання для виконання

Завдання 1 «Відстеження станів Активності»

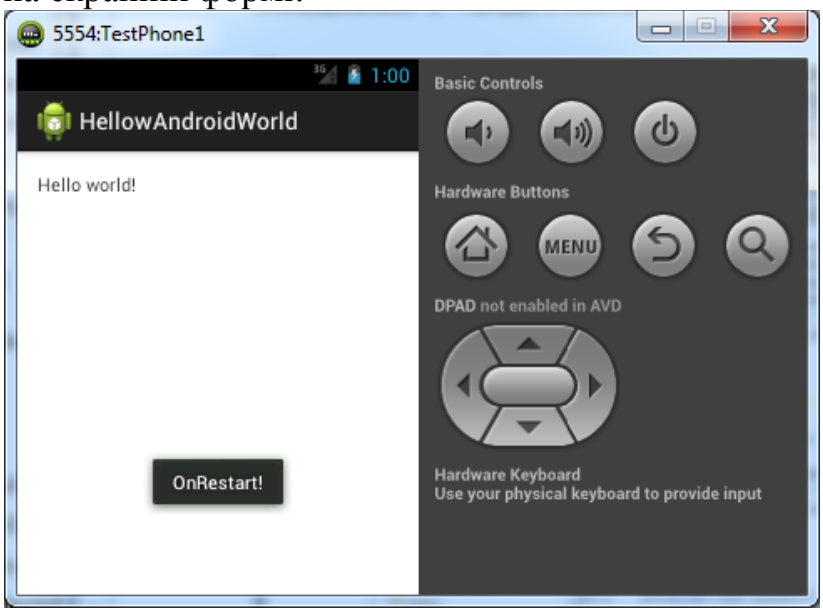
1. В наявному проекті HelloAndroidWorld відкрийте в редакторі клас HelloAndroidWorld.java.

2. Перевизначте методи onPause, onStart, onRestart для класу і внесіть зміни в метод onCreate:

```
package com.example.helloandroidworld;
import android.app.Activity;
import android.os.Bundle;
import android.widget.Toast;
public class HelloAndroidWorld extends Activity {
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Toast.makeText(this, "onCreate()", Toast.LENGTH_LONG).show();
}
@Override
protected void onPause() {
        Toast.makeText(this, "onPause()", Toast.LENGTH_LONG).show();
        super.onPause();
}
@Override
protected void onRestart() {
        super.onRestart();
        Toast.makeText(this, "onRestart()", Toast.LENGTH_LONG).show();
}
@Override
protected void onStart() {
```

super.onStart();
Toast.makeText(this, "onStart()", Toast.LENGTH_LONG).show();}}

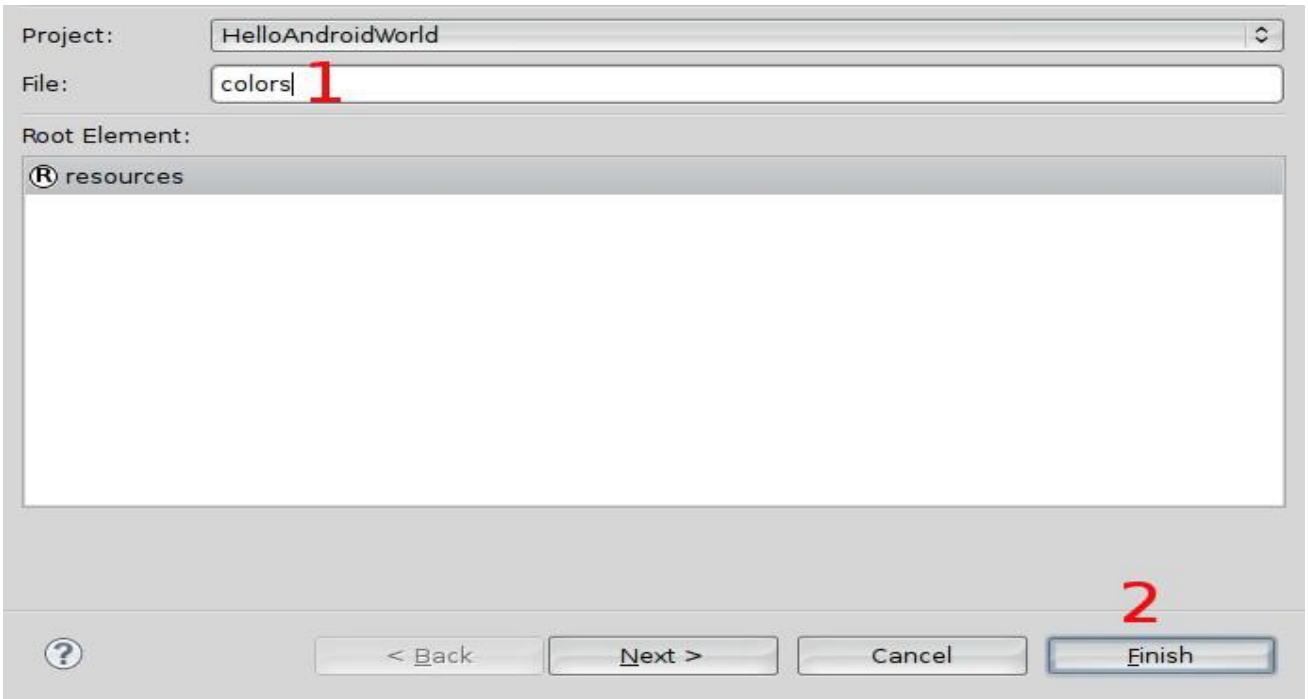
3. Запустіть додаток на виконання в емуляторі (Ctrl + F11, Android Project) і переконайтеся, що при зміні стану додатки HelloAndroidWorld на екрані емулятора з'являються відповідні повідомлення типу Toast, як показано нижче на екранній формі:



4. Поексперементуйте в додатку, щоб з'ясувати, за яких умов викликаються реалізовані обробники подій.

Завдання 2. «Використання значень рядків і кольорів»

1. В наявному проекті HelloAndroidWorld створіть файл colors в каталозі res / values: File → New → Android XML File:



2. Відредагуйте вміст файлу res / values / colors.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="view_bkg_color">#FF0</color>
    <color name="screen_bkg_color">#F88</color>
    <color name="text_color">#8004</color>
</resources>
```

3. Відредагуйте вміст файла res/values/strings.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello, Android World!</string>
    <string name="app_name">HelloAndroidWorld</string>
</resources>
```

4. Внесіть зміни у файл розмітки res / layout / main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    android:background="@color/screen_bkg_color">
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello"
        android:background="@color/view_bkg_color"
        android:textColor="@color/text_color"/>
</LinearLayout>
```

5. Збережіть всі незбережені файли і запустіть додаток.

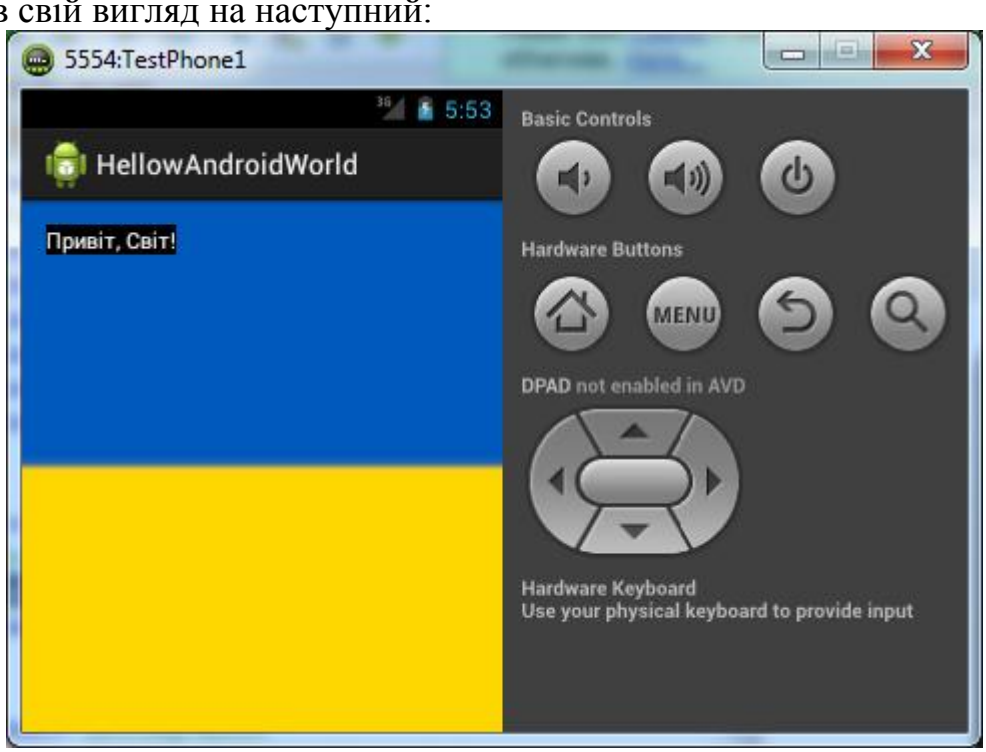
6. Впевніться, що зовнішній вигляд програми змінився відповідно до визначених ресурсів.

7.Поекспереметуйте зі значеннями кольорів, прозорістю і HTML-тегами в строкових константах для зміни зовнішнього вигляду програми.

Завдання 3 «Локалізація додатку»

1. Змініть ресурси програми HelloAndroidWorld так, щоб на екрані відображалось HelloAndroidWorld і HelloWorld

2. Додайте потрібні підкаталоги і ресурси в каталог res проекту HelloAndroidWorld, щоб при налаштуванні української мови додаток змінював свій вигляд на наступний:



3. Зробіть ті ж дії для польської і української мови:

4. Відновіть мовні налаштування на віртуальному пристрої.

Завдання для виконання

Виконайте завдань до виконання у відповідності до теоретичних відомостей поданих вище.

Вимоги до звіту

В звіт по роботі включіть текст програм, що реалізують описані завдання та екранні форми, що відображають їх виконання.

Контрольні запитання

1. Стани активностей та їх відстеження.
2. Особливості використання ресурсів у коді програми.
3. Робота із анімацією в Android.
4. Робота із кольорами та колірні моделі в Android.
5. Життєвий цикл активності.

ПРАКТИЧНА РОБОТА №4

Тема роботи: Робота із анімацією та макетами в ОС Android.

Мета роботи: Отримати досвід роботи із анімацією та ознайомитись із основними макетами для побудови користувацького інтерфейсу додатків під ОС Android.

Теоретичні відомості

У даній практичній роботі розглядаються особливості створення анімації та основних блоків для побудови користувацького інтерфейсу. Показано різні XML макети (компонування) в Android та практично реалізовано RelativeLayout і LinearLayout.

Створення власної реалізації класу Application дає можливість:

• контролювати стан додатку;

• передавати об'єкти між програмними компонентами;

• підтримувати і контролювати ресурси, які використовуються в декількох компонентах однієї програми.

При створенні операційною системою процесу, в якому буде виконуватися програма (з погляду Unix-подібних систем, процеси є контейнерами, в яких виконуються програми) створюється екземпляр класу Application, який описаний в маніфесті додатку. Таким чином, Application за своєю природою є сінглтоном (singleton) і повинен бути правильно реалізований, щоб надавати доступ до своїх методів і змінних.

Нижче показаний каркас для наслідування класу Application і використання його в якості сінглтона:

```
import android.app.Application;
import android.content.res.Configuration;
public class MyApplication extends Application {
    private static MyApplication singleton;
    // Повертає экземпляр данного классу
    public static MyApplication getInstance() {
        return singleton;
    }
    Override
    public final void onCreate() {
        super.onCreate();
        singleton = this; } }
```

Після створення реалізація класу Application повинна бути зареєстрована у маніфесті додатку, для цього використовується елемент <*application*>:

```
<application
    android:icon="@drawable/icon"
    android:name="MyApplication">
    [... інкапсульовані элементи ...]
```

</application>

Тепер при запуску програми буде створювати екземпляр реалізації класу Application. Якщо є необхідність зберігання стану програми і глобальних ресурсів, необхідно реалізувати відповідні методи у реалізації класу і використовувати їх в компонентах застосунку*:*

SomeObject value = MyApplication.getInstance().getGlobalStateValue();

MyApplication.getInstance().setGlobalStateValue(someObjectValue);

Такий підхід може бути ефективний при передачі об'єктів між слабозв’язаними частинами програми і для контролю за загальними ресурсами і станом додатку.

Обробка подій життєвого циклу додатку

Аналогічно обробникам подій класу Activity, обробники подій класу Application так само можна перевизначати для управління реакцією додатка на ті чи інші події життєвого циклу:

```
import android.app.Application;
import android.content.res.Configuration;
public class MyApplication extends Application {
    @Override
    public void onConfigurationChanged(Configuration newConfig) {
        super.onConfigurationChanged(newConfig);
    }
    @Override
    public void onCreate() {
        super.onCreate();
    }
    @Override
    public void onLowMemory() {
        super.onLowMemory();
    }
    @Override
    public void onTerminate() {
        super.onTerminate();}}
```

Призначення цих методів наступне:

• onCreate: викликається при створенні програми, перевизначається для створення і ініціалізації властивостей тив яких зберігаються стан програми або глобальні ресурси.

• onTerminate: викликається при передчасному завершенні роботи програми (але може і не бути викликатись, якщо додаток закривається ядром, для звільнення ресурсів для інших програм.

• onLowMemory: надає можливість додаткам звільнити додаткову пам'ять (коли ОС не вистачає ресурсів). Цей метод перевизначається для того, щоб очистити кеш або звільнити непотрібні в даний момент ресурси.

• onConfigurationChanged: перевизначається, якщо необхідно відстежувати зміни конфігурації на рівні додатку (такі, наприклад, як поворот екрану, закриття висувної клавіатури пристрою).

Поняття контексту

Клас android.context.Context є інтерфейсом для доступу до глобальної інформації про додаток. Це абстрактний клас реалізація якого забезпечується системою Android. Context дозволяє отримати доступ до специфічних для даного додатку ресурсів і класів, а також для виклику операцій на рівні додатку, таких, як запуск активності, відправка повідомлень, отримання намірів (Intent) та інше.

Даний клас також є базовим для класів Activity, Application і Service. Отримати доступ контексту можна за допомогою методів getApplicationContext, getContext, getBaseContext, а також просто за допомогою властивості this (зсередини активності або сервісу). Одним із способів при виклику статичного методу makeText класу Toast, що отримує контекст в якості першого параметра:

Toast.makeText(this, "onCreate()", Toast.LENGTH_LONG).show();

Типове використання контексту може бути таким:

TextView myTextView = new TextView(getContext());

ListAdapter listAdapter = new

SimpleCursorAdapter(getApplicationContext(), ...);

Доступ до стандартних глобальних ресурсів:

context.getSystemService(LAYOUT_INFLATER_SERVICE);

prefs=getApplicationContext().getSharedPreferences("PREFS",MODE_PRIVATE);

Неявне використання глобальних компонентів системи:

cursor = getApplicationContext().getContentResolver().query(uri, ...);

Інтерфейс користувача. Основні поняття і зв'язки між ними

Клас View є базовим класом для всіх візуальних елементів UI (елементів управління (Control) і віджетів (Widget)). Всі ці елементи, в тому числі і розмітка (Layout), є розширеннями класу View.

Групи (ViewGroup) - нащадки класу View; можуть містити в собі кілька дочірніх View. Розширення класу ViewGroup використовується для створення складних View, що складаються з взаємопов'язаних компонентів. Клас ViewGroup також є базовим для різних розміток (Layout).

Активності (Activity) – відображають екрани або вікна (з точки зору побудови UI), є «андроїдними еквівалентами» форм. Для відображення UI активності використовують View.

Для створення додатків з унікальними інтерфейсом розробнику іноді доводиться розширювати і модифікувати стандартні View, комбінуючи їх зі стандартними.

Android надає розробнику можливість використання набору готових елементів користувацького інтерфейсу, що прописані у класі View:

• TextView. Стандартний елемент, призначений для виведення тексту. Підтримує багаторядкове відображення, форматування і автоматичний перенос.

• EditText. Редаговане поле для введення тексту. Підтримує багаторядкове введення, перенесення слів на новий рядок і текст підказки.

• ListView. Група уявлень (ViewGroup), яка формує вертикальний список елементів, відображаючи їх у вигляді рядків всередині списку. Найпростіший об'єкт ListView використовує TextView для виводу на екран значень toString (), що належать елементом масиву.

• Spinner. Складовий елемент, що відображає TextView у поєднанні з відповідним ListView, який дозволяє вибрати елемент списку для відображення в текстовому рядку. Сам рядок складається з об'єкта TextView і кнопки при натисканні на яку починається діалог вибору. Зовні цей елемент нагадує тег <SELECT> в HTML.

• Button. Стандартна кнопка, яку можна натискати.

• CheckBox. Кнопка, що має два стани. Представлена у вигляді прапорця.

• RadioButton. «Радіокнопка», яка дозволяє вибрати тільки один з декількох варіантів.

• ViewFlipper. Група уявлень (ViewGroup), що дозволяє визначити набір елементів і горизонтальний рядок, в якому може виводитися тільки одне View. При цьому переходи між елементами, які відображаються здійснюються за допомогою анімації.

Android пропонує і інші реалізації View, такі, як елементи для вибору дати і часу, поля введення з автоматичним доповненням, галереї, вкладки і навіть карти (MapView).

Більш повний список підтримуваних системою View можна побачити за адресою http://developer.android.com/guide/tutorials/views/index.html

Крім готових View, розробник, при необхідності, може створювати власні, розширюючи клас View або його підкласи.

Розмітки в Android

Розмітка (Layout) є розширенням класу ViewGroup і використовується для розміщення дочірніх компонентів на екрані пристрою. Використовуючи вкладені розмітки, можна створювати користувацькі інтерфейси будь-якої складності.

Найбільш часто використовувані види розмітки:

• FrameLayout. Найпростіша розмітка, прикріплює кожне нове дочірнє подання до лівого верхнього кута екрану, накладаючи новий елемент на попередній та затуляє його.

• LinearLayout. Поміщає дочірні View в горизонтальний або вертикальний ряд. Вертикальна розмітка являє собою колонку, а

горизонтальна - рядок з елементами. Дана розмітка дозволяє задавати не тільки розміри, але і «відносну вагу» дочірніх елементів, завдяки чому можна гнучко контролювати їх розміщення на екрані.

• RelativeLayout. Найбільш гнучкий серед стандартних видів розмітки. Дозволяє вказувати позиції дочірніх View щодо меж вільного простору та інших View.

• TableLayout. Дозволяє розміщувати дочірні View всередині комірок «сітки», що складається з рядків і стовпців. Розміри комірок можуть залишатися постійними або автоматично розтягуватися при необхідності.

• Gallery. Представляє елементи у вигляді прокручуваного горизонтального списку (зазвичай графічні елементи).

Актуальну інформацію про властивості і можливості різних видів розмітки можна отримати за адресою:

http://developer.android.com/guide/topics/ui/layout-objects.html

Найбільш поширений спосіб реалізації розмітки екрану - використання зовнішніх ресурсів: XML-файлів, що описують розміщення елементів на екрані і їх параметри.

В попередніх практичних роботах розглядався вміст файла res / layout / main.xml для зміни зовнішнього вигляду додатку HelloAndroidWorld, а тепер докладніше розглянемо його вміст:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:orientation="vertical" >
        <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello"
        android:textColor="@color/text_color" />
</LinearLayout>
```

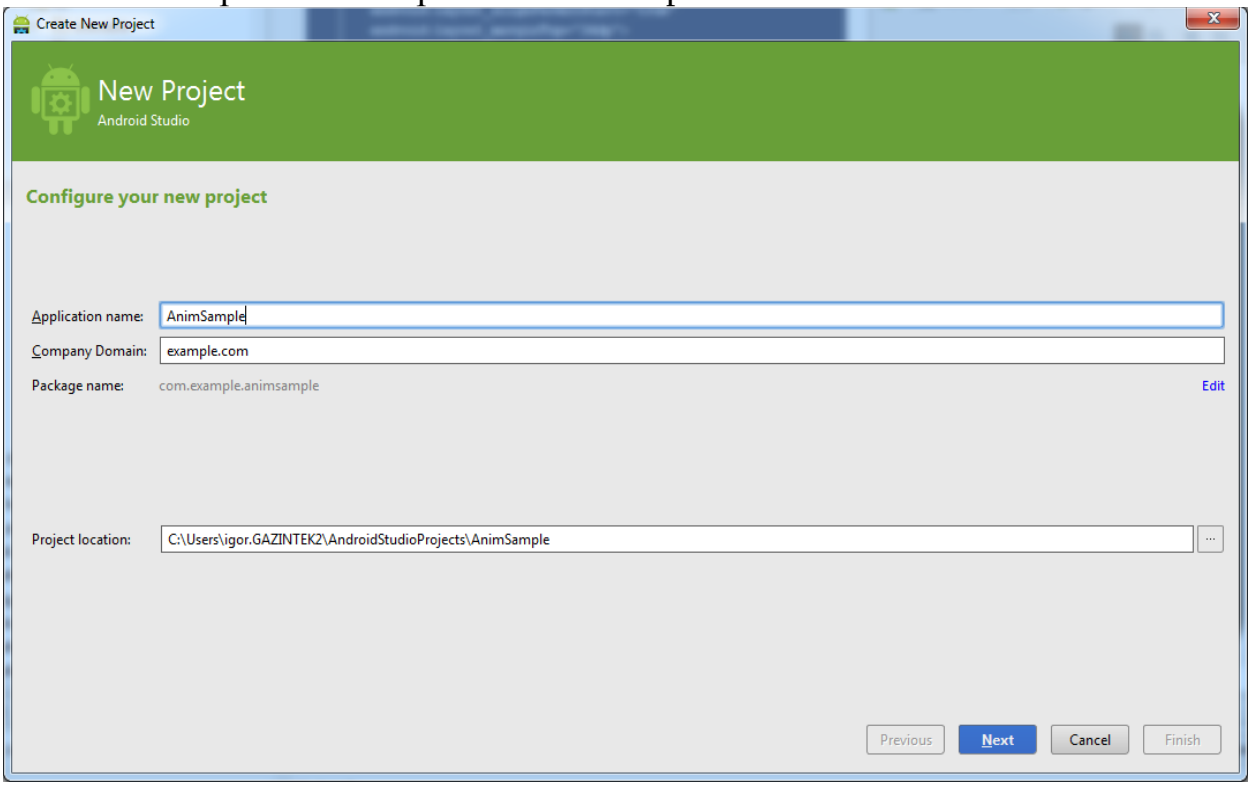
Кореневим елементом цієї розмітки є <LinearLayout>, що має один дочірній елемент <TextView>. Атрибути <LinearLayout> визначають простір імен «android» (xmlns:android="http://schemas.android.com/apk/res/android"), ширину (android:layout_width), висоту (android:layout_height)) і орієнтацію (android: orientation), яка визначає спосіб розміщення дочірніх елементів всередині LinearLayout: вертикально чи горизонтально. Зверніть увагу на відносне (щодо розмірів батьківського елементу), а не абсолютне (в пікселях) задання розмірів (ширини і висоти). Такий спосіб визначення розмірів є кращим і дозволяє створювати дизайн додатків, що не прив’язаний до розмірів екрану пристрою.

Елемент <TextView> в нашому випадку має наступні атрибути: ширину і висоту, а також посилання на рядок з ім'ям «hello» і колір, що використовується для відображення тексту «text_color».

Завдання для виконання

Завдання 1 «Робота із анімацією»

1. Створіть новий проект AnimSample.



2. В каталозі res створіть каталог anim, а в ньому файл з ім'ям ship_anim.xml, що описує анімацію. Відредагуйте файл, щоб він мав наступний вміст:

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android" >
<rotate
        android:duration="3333"
        android:fromDegrees="0"
        android:pivotX="50%"
        android:pivotY="50%"
        android:repeatCount="infinite"
        android:repeatMode="reverse"
        android:toDegrees="1080" />
<translate
        android:duration="1900"
        android:fromXDelta="-50%p"
        android:repeatCount="infinite"
        android:repeatMode="reverse"
```

```
        android:toXDelta="50%p" />
    <translate
        android:duration="1300"
        android:fromYDelta="-50%p"
        android:repeatCount="infinite"
        android:repeatMode="reverse"
        android:startOffset="123"
        android:toYDelta="50%p" />
    <alpha
        android:duration="500"
        android:fromAlpha="1.0"
        android:repeatCount="infinite"
        android:repeatMode="reverse"
        android:toAlpha="0.3" />
    <scale
        android:duration="10000"
        android:fromXScale="0.0"
        android:fromYScale="0.0"
        android:pivotX="50%"
        android:pivotY="50%"
        android:repeatCount="infinite"
        android:repeatMode="reverse"
        android:toXScale="2.5"
        android:toYScale="2.5" />
</set>
```

3. Додайте малюнок, до якого буде застосовуватися анімація (файл lander_plain.png), в каталог res / drawable-mdpi.

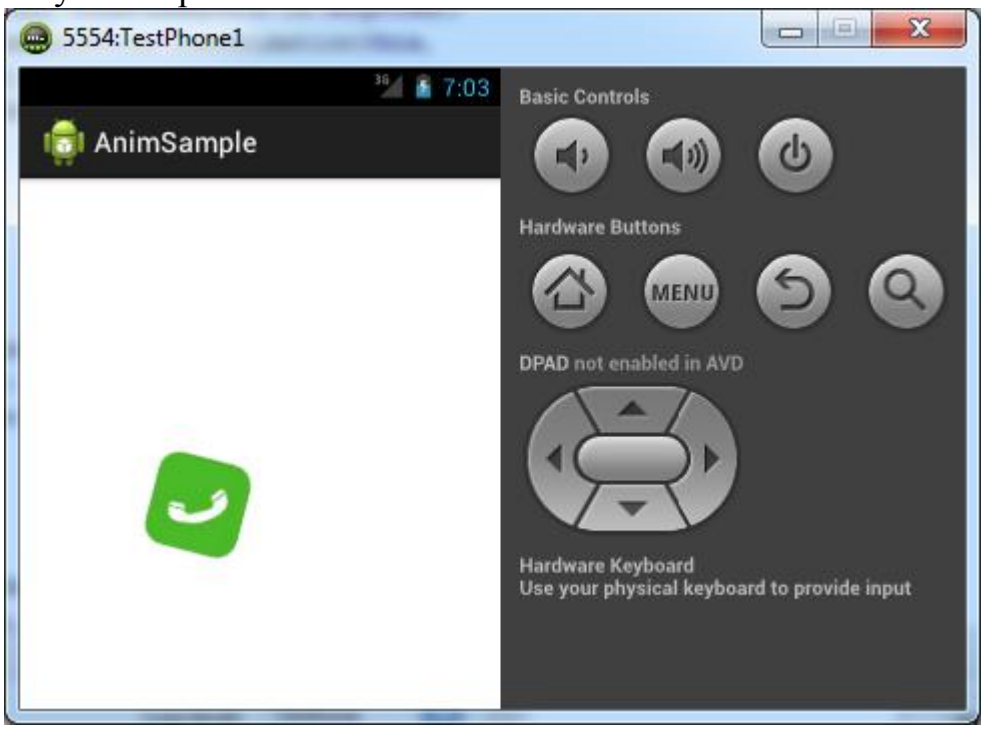
4. У файлі розмітки res / layout / main.xml замініть елемент TextView на ImageView з наступними атрибутами:

```
<ImageView
android:id="@+id/shipView"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:src="@drawable/lander_plain" />
```

5. В кінці методу onCreate Активності (вона в цьому проекті одна) додайте наступні рядки:

```
ImageView ship = (ImageView) findViewById(R.id.shipView);
Animation shipAnim = AnimationUtils.loadAnimation(this,
    R.anim.ship_anim);
ship.startAnimation(shipAnim);
```

6. Запустіть проект



Завдання 2 «Використання LinearLayout»

1. Продовжіть локалізацію додатку Hello Android World, тепер для французької мови. Для малювання прапора використовуйте вкладений LinearLayout з вертикальною орієнтацією дочірніх елементів:

2. Зверніть увагу, що при розміщенні елементів всередині вкладеного LinearLayout зручно вказувати атрибут android: layout_weight = "1", в цьому випадку дочірні віджети будуть розміщені по горизонталі рівномірно.

3. Після отримання потрібного результату поверніть стандартні мовні налаштування у віртуальному пристрої.

Завдання 3 «Використання RelativeLayout»

RelativeLayout є дуже корисним варіантом розмітки, що дозволяє створювати користувальницький інтерфейс без надмірної кількості вкладених елементів.

1. Створіть новий проект з ім'ям RelativeLayoutSample.

2. Додайте потрібні рядки в файл res / values / strings.xml і видаліть рядок з ім'ям hello:

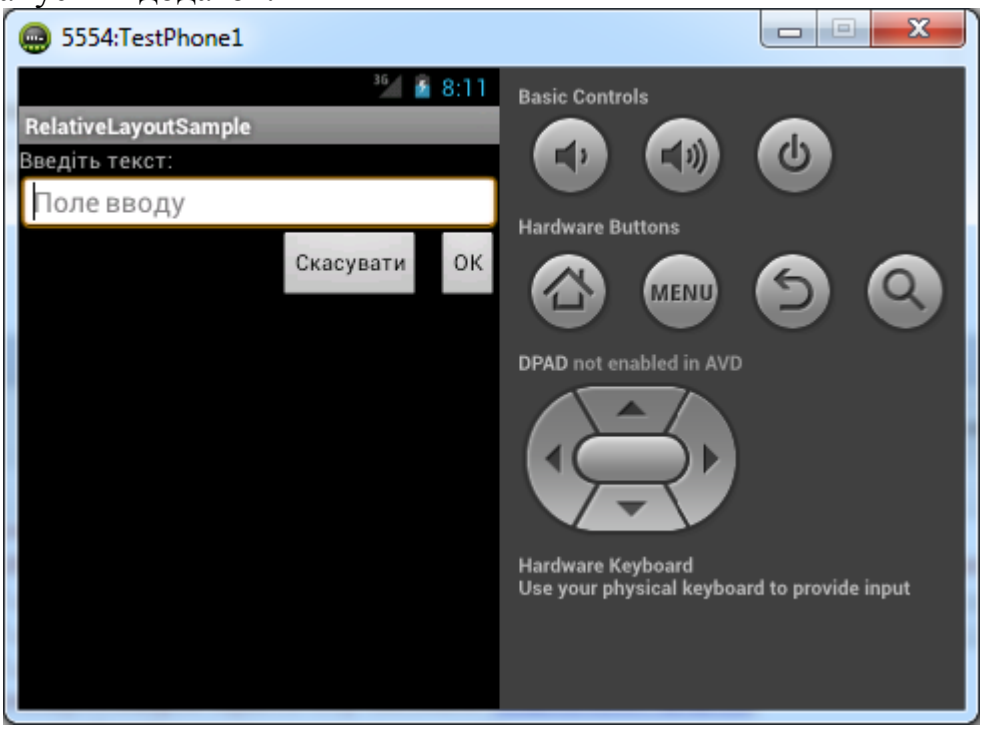
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="label_text">Введіть текст:</string>
    <string name="entry_hint">Поле введення</string>
    <string name="app_name">RelativeLayoutSample</string>
</resources>
```

3. Відредагуйте файл розмітки res / layout / main.xml так, щоб він мав наступний вміст:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
    <TextView
        android:id="@+id/label"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/label_text" />
    <EditText
        android:id="@+id/entry"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/label"
        android:background="@android:drawable/editbox_background"
        android:hint="@string/entry_hint" />
    <Button
        android:id="@+id/ok"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
```

android:layout_below="@id/entry"
android:layout_marginLeft="10dip"
android:text="@android:string/ok" />
<Button
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignTop="@id/ok"
android:layout_toLeftOf="@id/ok"
android:text="@android:string/cancel" />
</RelativeLayout>

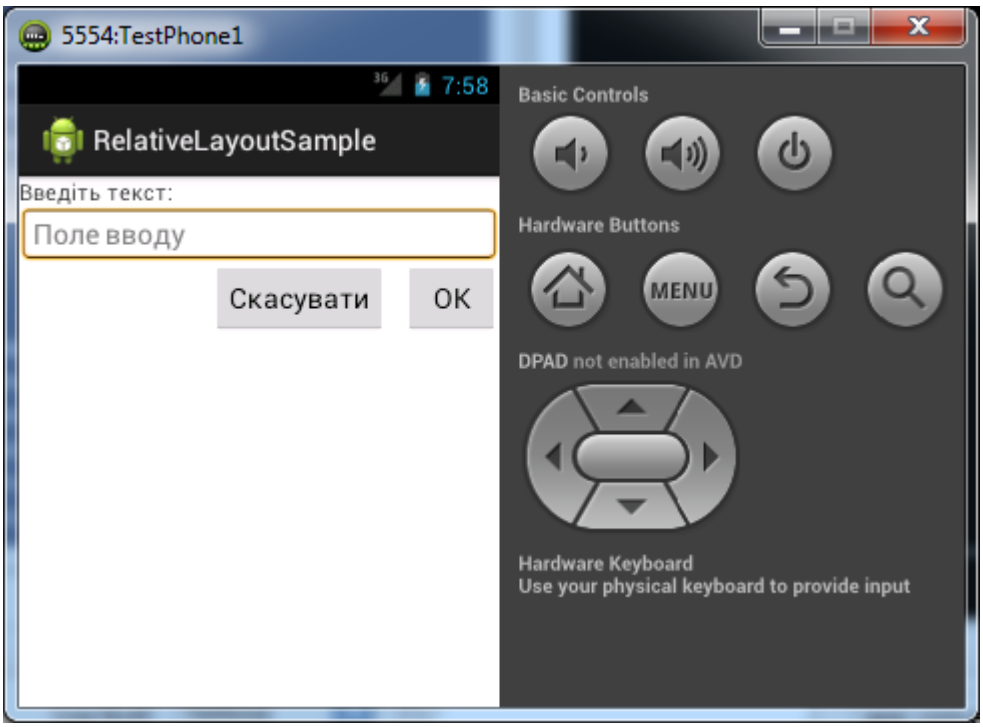
4.Запустіть додаток:



5. Відредагуйте файл AndroidManifest.xml, щоб змінити тему, використовувану додатком. Для цього у вузол <application> внесіть такі зміни:

<application
android:icon=*"@drawable/ic_launcher"*
android:label=*"@string/app_name"*
android:theme=*"@android:style/Theme.Light" >*

6. Запустіть додаток з новою темою:



7. Поексперементуйте з налаштуваннями мови у віртуальному пристрої і зверніть увагу, як при запуску програми змінюються написи на кнопках. Очевидно, що використання стандартних строкових значень дозволяє мінімізувати витрати на локалізацію додатків.

Завдання для виконання

Виконайте завдань до виконання у відповідності до теоретичних відомостей поданих вище.

Вимоги до звіту

В звіт по роботі включіть текст програм, що реалізують описані завдання та екранні форми, що відображають їх виконання.

Контрольні запитання

1. Наскільки добре платформа Android справляється з анімацією?
2. Чи існує спосіб зупинки відтворення анімації?
3. Які типи операцій підтримуються анімаціями руху?
4. Чи може бути використаний елемент-контейнер LinearLayout для розміщення всіх дочірніх елементів-уявлень View один за одним (по вертикалі)?
5. Назвіть віджети, які використовуються для створення користувацького інтерфейсу.

ПРАКТИЧНА РОБОТА №5

Тема роботи: Робота із віджетами та керуючими елементами.

Мета роботи: Ознайомитись з використанням віджетів TabWidget, WebView, ListView. Навчитися використовувати в інтерфейсі користувача різні керуючі елементи: кнопки із зображеннями, радіокнопки, чекбокси та ін.

Теоретичні відомості

«Табулювана» розмітка дозволяє створювати UI та змістові вкладки із використанням віджетів TabHost і TabWidget.

TabHost є кореневим вузлом в розмітці, що містить TabWidget для відображення «вкладок», і FrameLayout для відповідного їм контенту.

Відображення контенту вкладок можна реалізувати двома способами:

• описавши View для вмісту кожної вкладки всередині однієї і тієї ж активності;

• використовуючи вкладки для перемикання між різними активностями.

Вибір конкретної реалізації залежить від потреб розробника, але зазвичай більш привабливим є другий варіант, так в цьому випадку різні вкладки обробляються різними активностями, а не однією (досить громіздкою), що дозволяє робити код більш ясним і керованим. У даній практичній роботі використовуємо варіант реалізації з незалежними активностями, тому для реалізації трьох вкладок нам потрібні чотири активності (три для вкладок і одна - головна).

Адаптери в Android

Адаптери в Android є сполучними класами між даними програми та View. Адаптер відповідає за створення дочірніх View, що відображають кожен елемент всередині батьківського віджета, а також забезпечує доступ до вихідних даних, які використовуються додатком. View, які використовують прив'язку до адаптеру, повинні бути нащадками абстрактного класу AdapterView.

Android містить набір стандартних адаптерів, які доставляють дані в стандартні віджети для користувача інтерфейсу. Двома найбільш поширеними адаптерами є ArrayAdapter і SimpleCursorAdapter.

ArrayAdapter використовує механізм узагальнених типів (generics) мови Java для прив'язки батьківського класу AdapterView до масиву об'єктів зазначеного типу. За замовчуванням ArrayAdapter використовує метод toString () для кожного елемента в масиві, щоб створити і заповнити текстовими даними віджети TextView.

SimpleCursorAdapter прив'язує вказане в розмітці View до стовпців курсора, що став результатом запиту до СУБД або контент-провайдер. Для

його використання потрібно описати розмітку у форматі XML, а потім прив'язати кожен стовпець до View з цієї розмітки.

SimpleAdapter дозволяє прив'язати ListView до списку ArrayList, який містить об'єкти типу Map (асоціативні масиви, що містять пари «ключ-значення»). Для кожного такого об'єкта при відображенні використовується один елемент з ListView. Як і для SimpleCursorAdapter, для відображення застосовується XML-розмітка, до елементів якої прив'язуються члени кожного об'єкта типу Map.

Використання адаптерів для прив'язки даних

Щоб застосувати адаптер, необхідно викликати з View (нащадка абстрактного класу AdapterView) метод setAdapter () (або його більш конкретизований варіант). Приклад використання ArrayAdapter:

```
ArrayList<String> myStringArray = new ArrayList<String>();
ArrayAdapter<String> myAdapterInstance;
int layoutID = android.R.layout.simple_list_item_1;
myAdapterInstance = new ArrayAdapter<String>(this, layoutID,
        myStringArray);
myListView.setAdapter(myAdapterInstance);
```

Приклад використання SimpleCursorAdapter:

```
String uriString = "content://contacts/people/";
Cursor myCursor = managedQuery(Uri.parse(uriString), null,
        null, null, null);
String[] fromColumns = new String[] {People.NUMBER, People.NAME};
int[] toLayoutIDs = new int[] { R.id.nameTextView, R.id.numberTextView};
SimpleCursorAdapter myAdapter;
myAdapter = new SimpleCursorAdapter(this, R.layout.simplecursorlayout,
        myCursor, fromColumns, toLayoutIDs);
myListView.setAdapter(myAdapter);
```

Порядок виконання роботи

Завдання 1 «Робота із вкладками»

1. Створить новий проект з ім'ям TabWidgetSample.

2. Опишить потрібні рядки у файлі res / values / strings.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">TabWidgetSample</string>
    <string name="tab1_indicator">Students</string>
    <string name="tab2_indicator">Teachers</string>
    <string name="tab3_indicator">Classes</string>
    <string name="tab1_content">This is Students tab</string>
    <string name="tab2_content">This is Teachers tab</string>
    <string name="tab3_content">This is Classes tab</string>
```

```
</resources>
```

3. Відредагуйте файл res / layout / main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<TabHost xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@android:id/tabhost"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <LinearLayout
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:padding="5dp">
        <TabWidget
            android:id="@android:id/tabs"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content" />
        <FrameLayout
            android:id="@android:id/tabcontent"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:padding="5dp" />
    </LinearLayout>
</TabHost>
```

4. Створіть три активності з іменами StudentsActivity, TeachersActivity і ClassesActivity.

5. В кожній створеній активності перевизначите метод onCreate наступним чином (внісши відповідні зміни до ім'я ресурсу *tabX_content*):

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Resources res = getResources();
    String contentText = res.getString(R.string.tab2_content);
    TextView textView = new TextView(this);
    textView.setText(contentText);
    setContentView(textView); }
```

Зверніть увагу на те, що в кожній активності використовуються власні рядкові ресурси.

6. Замініть базовий клас для TabWidgetSampleActivity з Activity на TabActivity і перевизначите метод onCreate наступним чином:

```
@Override
public void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Resources res = getResources();
        String tab1Indicator = res.getString(R.string.tab1_indicator);
        String tab2Indicator = res.getString(R.string.tab2_indicator);
        String tab3Indicator = res.getString(R.string.tab3_indicator);
        TabHost tabHost = getTabHost();
        TabHost.TabSpec spec;
        Intent intent;
        intent = new Intent().setClass(this, StudentsActivity.class);
        spec = tabHost.newTabSpec("students").setIndicator(tab1Indicator)
                .setContent(intent);
        tabHost.addTab(spec);
        intent = new Intent().setClass(this, TeachersActivity.class);
        spec = tabHost.newTabSpec("teachers").setIndicator(tab2Indicator)
                .setContent(intent);
        tabHost.addTab(spec);
        intent = new Intent().setClass(this, ClassesActivity.class);
        spec = tabHost.newTabSpec("class").setIndicator(tab3Indicator)
                .setContent(intent);
        tabHost.addTab(spec);
        tabHost.setCurrentTab(1);
    }
```

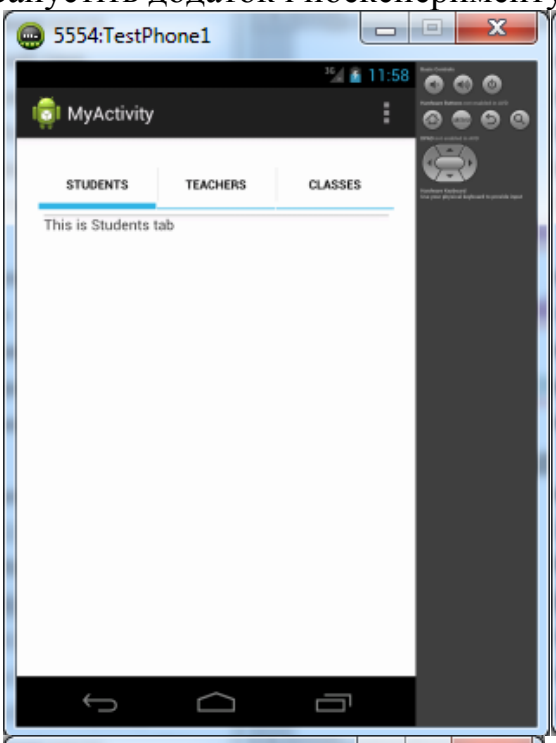
У цьому методі ми використовуємо для запуску потрібних активностей використовуються так звані явні наміри (Intent). Наміри та їх використання будуть розглянуті пізніше.

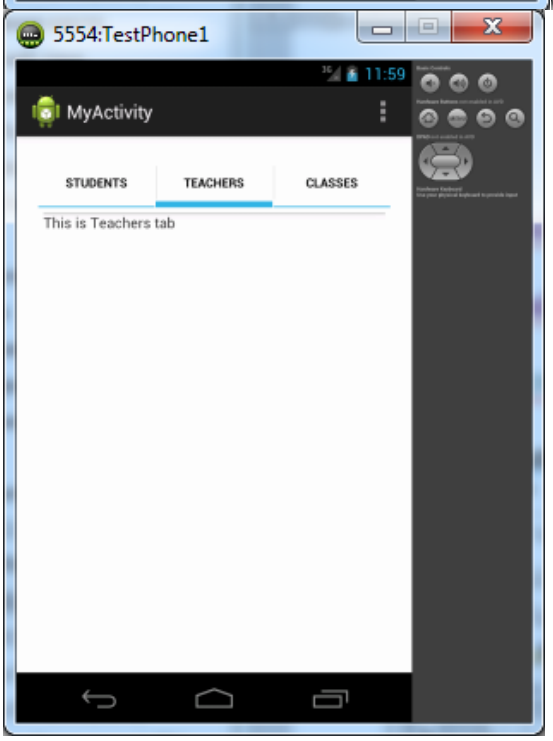
7. При спробі запуску проекту на виконання станеться аварійне завершення роботи програми, так як активності, створені для відображення контенту вкладок невідомі системі, тому що не описані в маніфесті додатку. Потрібно відредагувати файл AndroidManifest.xml так, щоб всі використовувані активності були в ньому описані. Крім того, для покращення зовнішнього вигляду інтерфейсу має сенс змінити використовувану тему оформлення на таку, де немає рядка заголовка. Вузол <application> повинен виглядати наступним чином:

```
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@android:style/Theme.NoTitleBar" >
        <activity
            android:name=".TabWidgetSampleActivity"
            android:label="@string/app_name" >
            <intent-filter>
            <action android:name="android.intent.action.MAIN" />
```

```
            <category
            android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".StudentsActivity" />
        <activity android:name=".TeachersActivity" />
        <activity android:name=".ClassesActivity" />
</application>
```

8. Запустіть додаток і поекспериментуйте з вкладками:





9. Локалізуйте програму за допомогою індивідуальних для різних мов строкових ресурсів.

Завдання 2 «Використання WebView»

У даній практичній роботі розглядається використання віджету web-браузера і застосовується ітеративний підхід до створення програми.

1. Створіть новий проект з ім'ям WebViewSample.

2. Відредагуйте файл res / layout / main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<WebView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/webview"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" />
```

3. Добавте в кінець методу onCreate активності WebView Sample Activity наступні рядки:

```
WebView webView = (WebView) findViewById(R.id.webview);
webView.getSettings().setJavaScriptEnabled(true);
webView.loadUrl("http://www.google.com.ua");
```

4. Запустіть додаток.

5. Очевидно, що в додатку відсутні повноваження на доступ до мережі. Додамо потрібні повноваження в маніфест додатка, додавши елемент <uses-permission> всередині кореневого вузла <mainfest>:

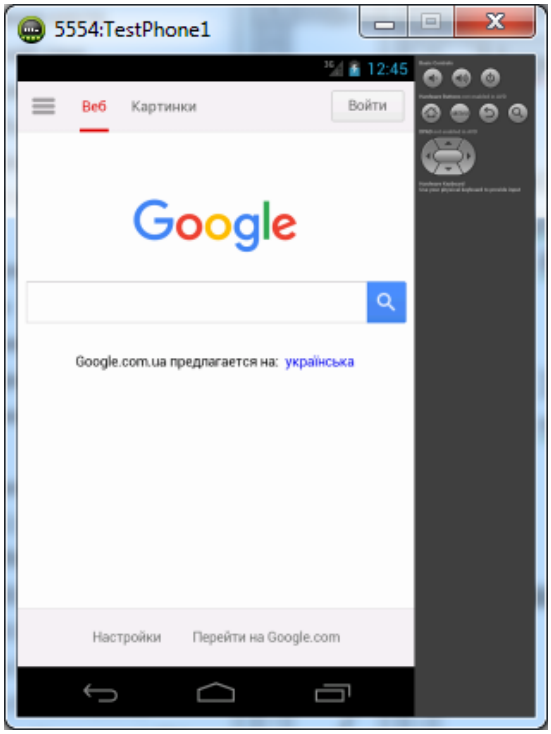
```
<uses-permission android:name="android.permission.INTERNET"/>
```

6. Запустимо проект.

7. Продовжимо покращувати додаток. Для збільшення корисної площі екрану заборонимо показ заголовка, для цього вкажемо відповідну тему у файлі маніфесту:

```
<activity
    android:name=".WebViewSampleActivity"
    android:label="@string/app_name"
    android:theme="@android:style/Theme.NoTitleBar" >
```

8. Запустомо проект і переконаємося тому, що (перша) мета досягнута.



9. В даний час посилання, що ведуть за межі сайту www.google.com.ua обслуговуються стандартним web-браузером, а не нашим WebView, оскільки воно не в змозі обробити ці запити і віджет webView автоматично посилає системі відповідний намір (Intent), обробляється стандартним браузером. У цієї проблеми є два рішення:

• додати до файлу маніфесту потрібний фільтр намірів (Intent Filter).

• перевизначити всередині нашої активності клас WebViewClient, щоб додаток міг обробляти свої власні запити на відображення web-ресурсів за допомогою наявного віджета WebView.

Другий варіант є більш прийнятним ще й тому, що в цьому випадку не буде розглядатися системою як альтернативний web-браузер, що сталося б у разі додавання фільтра намірів в маніфест файлі.

10. Винесемо об'єкт webView з методу onCreate і зробимо його членом класу, після чого додамо всередині активності новий клас WebViewSampleClient, що розширює клас WebViewClient. Встановимо свій обробник запитів на відображення web-ресурсів:

```
public class WebViewSampleActivity extends Activity {
    WebView webView;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        webView = (WebView) findViewById(R.id.webview);
        webView.getSettings().setJavaScriptEnabled(true);
        webView.loadUrl("http://www.ya.ru");
```

```
        webView.setWebViewClient(new WebViewSampleClient());
        }
    private class WebViewSampleClient extends WebViewClient {
        @Override
        public boolean shouldOverrideUrlLoading(WebView view,
    String url) {
            view.loadUrl(url);
            return true; }}}
```

11. Запустимо додаток і переконаємося, що залишився серйозний недолік - «неправильна» обробка натискання кнопки «назад», додаток при цьому закінчує свою роботу. Вирішення цієї проблеми просте і прямолінійне: зробимо свій обробник подій натискання на кнопки, в якому буде реалізовано тільки одну дію: якщо була натиснута кнопка «назад», віджет WebView отримає команду повернутися на попередню сторінку (якщо у нього є така можливість). Перевизначити метод onKeyDown в активності:

```
@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
    if ((keyCode == KeyEvent.KEYCODE_BACK) &&
webView.canGoBack()) {
        webView.goBack();
        return true;
    }
    return super.onKeyDown(keyCode, event);
}
```

12. Запустимо додаток і переконаємося, що мета досягнута.

Завдання 3 «Використання ListView»

1. Створіть новий Android проект ListView Sample.

2. В каталозі res / values створіть файл arrays.xml з наступним вмістом:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="stations">
        <item>Академмістчко</item>
        <item>Житомирська</item>
        <item>Святошин</item>
        <item>Нивки </item>
        <item>Берестейська</item>
        <item>Шулявська</item>
        <item>Політехнічний інститут </item>
        <item>Вокзальна</item>
        <item>Університет</item>
        <item>Татральна</item>
```

```
        <item>Хрещатик </item>
        <item>Арсенальна </item>
        <item>....</item>
        .....

    </string-array>
</resources>
```

3. В каталозі res / layout створіть файл list_item.xml з наступним вмістом:

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:padding="10dp"
    android:textSize="16sp" >
</TextView>
```

4. Модифікуйте метод onCreate активності:

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Resources r = getResources();
    String[] stationsArray = r.getStringArray(R.array.stations);
    ArrayAdapter<String> aa = new ArrayAdapter<String>(this,
    R.layout.list_item, stationsArray);
    setListAdapter(aa);
    ListView lv = getListView();
}
```

5. Змініть базовий клас активності з Activity на ListActivity.

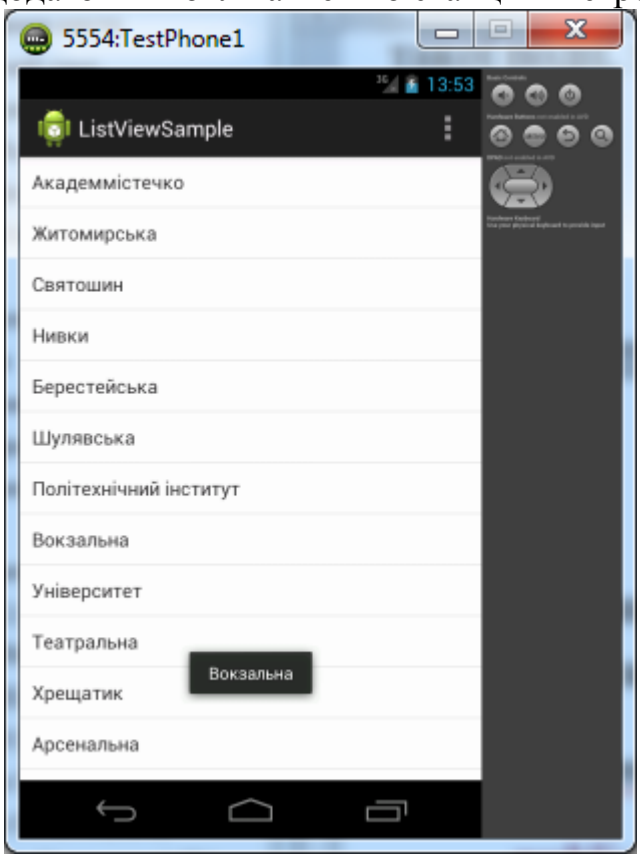
6. Запустіть додаток.

7. Для реакції на кліки по елементах списку потрібно додати обробник такої події, за допомогою методу setOnItemClickListener. В якості обробника буде використовуватися анонімний об'єкт класу OnItemClickListener. Додайте наступний код в потрібне місце:

```
lv.setOnItemClickListener(new OnItemClickListener() {
    public void onItemClick(AdapterView<?> parent, View v,
            int position, long id) {
        CharSequence text = ((TextView) v).getText();
        int duration = Toast.LENGTH_LONG;
        Context context = getApplicationContext();
```

```
            Toast.makeText(context, text, duration).show();
        }});
```

8. Запустіть додаток і «поклікайте» по станціях метро.



Завдання 4 «Використання керуючих елементів в інтерфейсі»

Мета - навчитися використовувати в інтерфейсі користувача різні керуючі елементи: кнопки із зображеннями, радіокнопки, чекбокси та ін.

1. Створіть новий проект ControlsSample.

2. Відредагуйте файл res / layout / main.xml так, щоб залишився тільки кореневий елемент LinearLayout. У нього надалі будуть додаватися необхідні дочірні елементи:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
</LinearLayout>
```

Для використання зображення замість тексту на кнопці потрібні три зображення для трьох станів кнопки: звичайного, вибраного («у фокусі») і нажатого. Всі ці три зображення з відповідними станами описуються в одному XML файлі, який використовується для створення такої кнопки:

- Скопіюйте потрібні зображення кнопки в каталог res / drawable-mdpi, для оновлення списку вмісту каталогу можна використовувати кнопку F5.

- В цьому ж каталозі створіть файл smile_button.xml, що описує, які зображення в яких станах кнопки потрібно використовувати.

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
        <item android:drawable="@drawable/smile_pressed"
              android:state_pressed="true"/>
        <item android:drawable="@drawable/smile_focused"
              android:state_focused="true"/>
        <item android:drawable="@drawable/smile_normal" />
</selector>
```

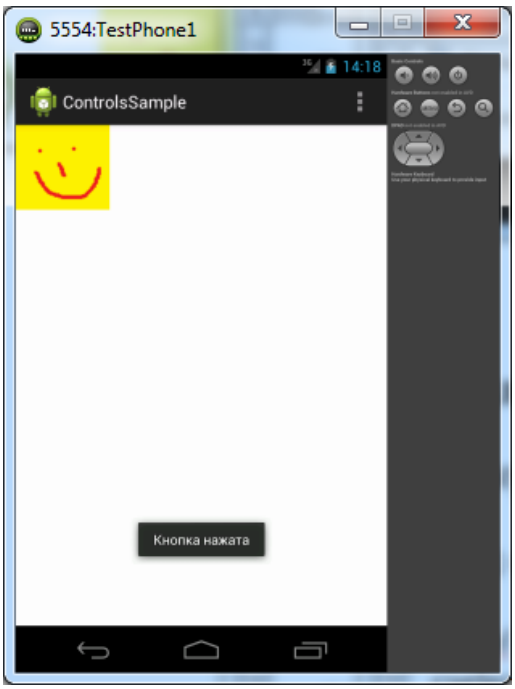
3. Додайте елемент Button всередині LinearLayout у файлі розмітки res / layout / main.xml:

```
<Button
android:id="@+id/button"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:background="@drawable/smile_button"
android:onClick="onButtonClicked"
android:padding="10dp" />
```

4. Зверніть увагу на атрибут android: onClick = "on Button Clicked", який вказує, який метод з активності буде використовуватися як обробник натискання на цю кнопку. Додайте цей метод в активність:

```
public void onButtonClicked(View v) {
Toast.makeText(this, "Кнопка нажата", Toast.LENGTH_SHORT).show();
}
```

5. Запустіть додаток і подивіться, як змінюється зображення кнопки в різних станах, а також як функціонує обробник натискання на кнопку.



Завдання 5 «Використання віджета CheckBox»

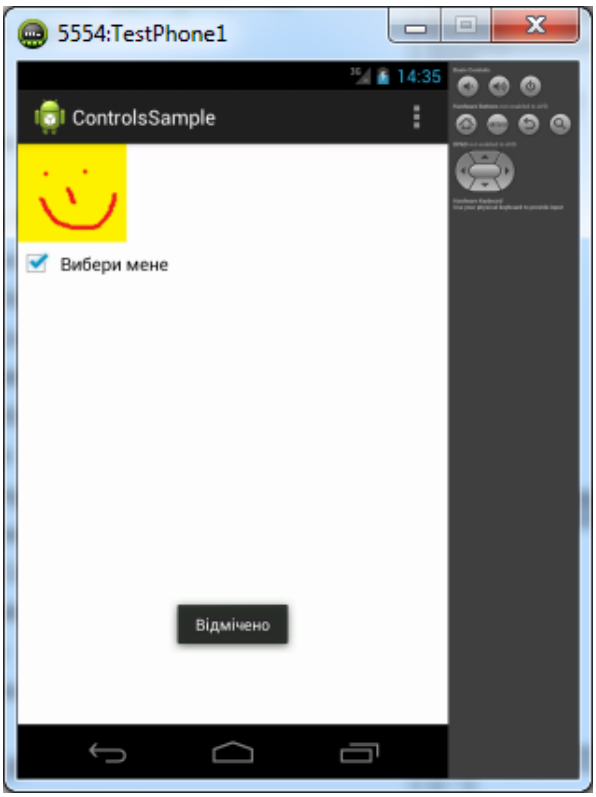
1. Добавте елемент CheckBox всередині LinearLayout у файлі розмітки res / layout / main.xml:

<CheckBox
android:id="@+id/checkbox"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:onClick="onCheckboxClicked"
android:text="Вибери мене" />

2. Атрибут android: onClick = "on Button Clicked" визначає, який метод з активності буде використовуватися як обробник натискання на віджет. Додайте цей метод в активність:

```
public void onCheckboxClicked(View v) {
if (((CheckBox) v).isChecked()) {
        Toast.makeText(this, "Оттмечено", Toast.LENGTH_SHORT).show();
} else {
        Toast.makeText(this, "Не відмічено",
Toast.LENGTH_SHORT).show(); }}
```

3. Запустіть додаток і подивіться на поведінку чекбокса в різних ситуаціях.



Завдання 6 «Використання віджета ToggleButton»

Даний віджет добре підходить в якості альтернативи радіо кнопкам і чекбоксам, коли потрібно перемикатися між двома взаємовиключними станами, наприклад, включити / виключити.

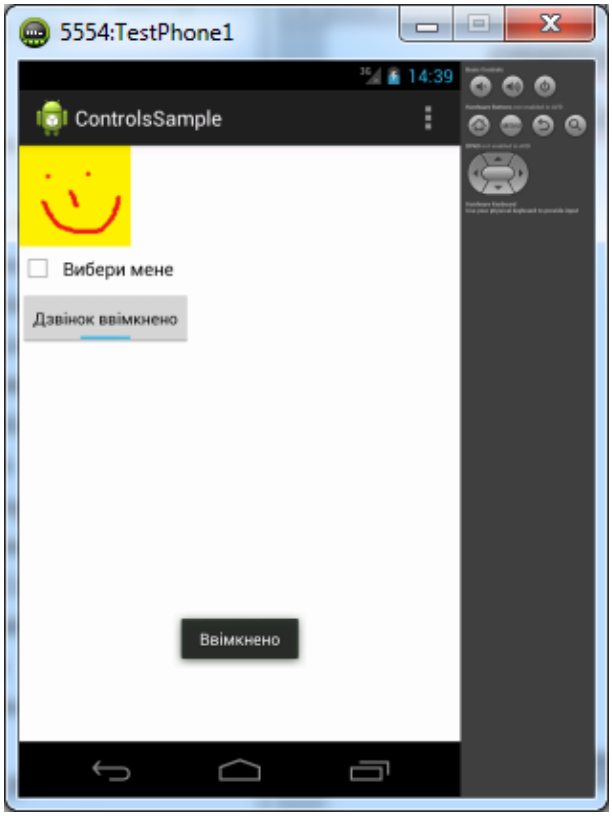
1. Добавте елемент ToggleButton всередині LinearLayout у файлі розмітки res / layout / main.xml:

<ToggleButton *android:id="@+id/togglebutton"*
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:textOn="Звонок включен"
android:textOff="Звонок выключен"
android:onClick="onToggleClicked"/>

2. Атрибут *android: onClick = "on Button Clicked"* визначає, який метод з активності буде використовуватися як обробник натискання на віджет. Додайте цей метод в активність:

```
public void onToggleClicked(View v) {
if (((ToggleButton) v).isChecked()) {
        Toast.makeText(this, "Ввівккнено", Toast.LENGTH_SHORT).show();
} else {
        Toast.makeText(this, "Вимкнено", Toast.LENGTH_SHORT).show();
}}
```

3. Запустіть додаток та провірте його функціонування.



Завдання 7 «Використання віджета RadioButton»

Радіокнопки використовуються для вибору між різними взаємовиключними варіантами. Для створення групи радіокнопок використовується елемент RadioGroup, усередині якого розташовуються елементи RadioButton.

1. Додайте наступні елементи розмітки всередині LinearLayout у файлі res / layout / main.xml:

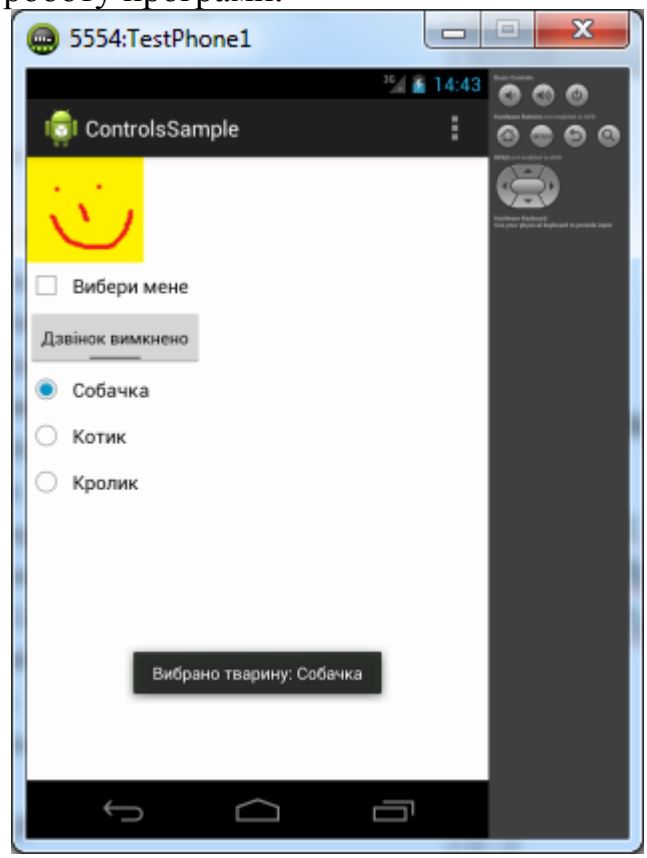
```
<RadioGroup
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical" >
<RadioButton
    android:id="@+id/radio_dog"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="onRadioButtonClicked"
    android:text="Собачка" />
<RadioButton
    android:id="@+id/radio_cat"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="onRadioButtonClicked"
    android:text="Котик" />
```

```
<RadioButton
        android:id="@+id/radio_rabbit"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="onRadioButtonClicked"
        android:text="Кролик" />
</RadioGroup>
```

2. Додайте метод on Radio Button Clicked в активність:

```
public void onRadioButtonClicked(View v) {
RadioButton rb = (RadioButton) v;
Toast.makeText(this, "Вибраний звір: " + rb.getText(),
            Toast.LENGTH_SHORT).show();}
```

3. Перевірте роботу програми.



Завдання 8 «Використання віджета EditText»

Віджет EditText використовується для введення тексту користувачем. Встановлений для цього віджета обробник натискань на кнопки буде показувати введений текст за допомогою Toast.

1. Добавте елемент EditText всередині LinearLayout у файлі розмітки res / layout / main.xml:

```
<EditText
```

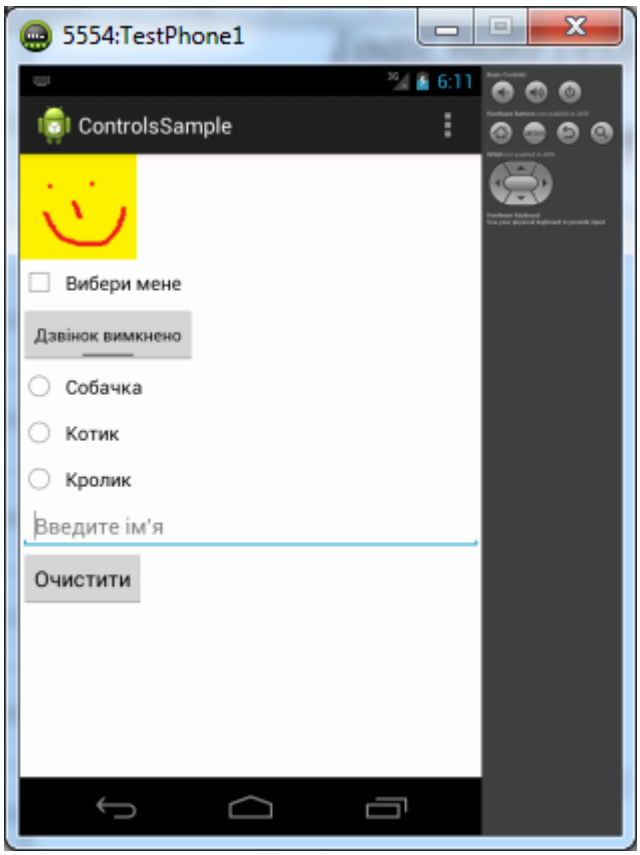
```
android:id="@+id/user_name"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:hint="Введіть ім'я"/>
```

2. Для обробки введеного користувачем тексту додайте наступний код в кінці методу onCreate. Зверніть увагу, цей обробник, на відміну від попередніх, повертає значення true або false. Семантика цих значень традиційна: true означає, що подію (event) оброблено і більше ніяких дій не потрібно, false означає, що подія не оброблено цим обробником і буде передана наступним обробникам в ланцюжку. У нашому випадку реагування відбувається тільки на натискання (ACTION_DOWN) кнопки Enter (KEYCODE_ENTER):

```
final EditText userName = (EditText) findViewById(R.id.user_name);
userName.setOnKeyListener(new View.OnKeyListener() {
@Override
public boolean onKey(View v, int keyCode, KeyEvent event) {
    if ((event.getAction() == KeyEvent.ACTION_DOWN)
            && (keyCode == KeyEvent.KEYCODE_ENTER)) {
        Toast.makeText(getApplicationContext(),
            userName.getText(),
            Toast.LENGTH_SHORT).show();
        return true;
    }
    return false;}});
```

3. Запустіть додаток і перевірте його роботу.

4. Добавте кнопку «Очистити» в розмітку і напишіть обробник, що очищує текстове поле (використовуйте метод setText віджета EditText).



5.Перевірте роботу програми.

Завдання для виконання

Виконайте завдань до виконання у відповідності до теоретичних відомостей поданих вище.

Вимоги до звіту

В звіт по роботі включіть текст програм, що реалізують описані завдання та екранні форми, що відображають їх виконання.

Контрольні запитання

1. У чому відмінність між методом setOnClickListener () і методом setOnitemClickListener () елемента ListView?
2. Який механізм виступає в ролі «сполучної ланки» між джерелом даних і елементом ListView?
3. Який елемент-контейнер найкраще підходить для вирівнювання дочірніх елементів View щодо батьківського елементу?
4. Особливості використання RelativeLayout.
5. Обробка подій життєвого циклу додатку.

ПРАКТИЧНА РОБОТА №6

Тема роботи: Наміри в ОС Android.

Мета роботи: Ознайомитись з викликом активності за допомогою явного наміру, з використанням неявних намірів, з одержанням даних з наміру, з використанням SharedPreferences для збереження налаштувань та зі створенням і використанням меню.

Теоретичні відомості

Наміри (Intent) в Android використовуються як механізму передачі повідомлень, який може працювати як всередині однієї програми, так і між додатками. Наміри можуть застосовуватися для:

• оголошення про бажання (необхідності) застосування запуску активності або сервісу для виконання певних дій;

• повідомлення про те, що сталась якась подія;

• явного запуску зазначеного сервісу або активності.

Останній варіант найбільш часто використовується.

Використання намірів для запуску активностей та обробка їх результатів

Щоб запустити потрібну активність, викликається метод start Activity (some Intent).

У конструкторі намірів можна явно вказати клас активності, яку потрібно запустити, або дію, яку потрібно виконати. У другому випадку система автоматично підбере потрібну активність, використовуючи механізм Intent Resolution. Метод startActivity знаходить та запускає активність, найбільш відповідну наміру користувача.

По закінченні роботи запущеної таким чином активності запустивши її активність не отримує ніяких повідомлень про результати обробки намірів. Якщо потрібно отримувати результати, використовується метод startActivityForResult.

Для явної вказівки того, яку активність (конкретний клас у додатку) потрібно запустити, створюються наміри за допомогою вказівки параметрів наступного конструктора: поточний Контекст додатка і клас Активності для запуску.

Intent intent = new Intent(MyActivity.this, MyOtherActivity.class);

startActivity(intent);

Активність, запущена за допомогою методу startActivity, повністю незалежна від активності, що її запустила і, відповідно, завершує свою роботу, нікому про це не повідомляючи. У той же час, програміст може запускати активності, «пов'язані» зі своїм «батьком». Такий спосіб відмінно підходить для ситуацій, коли «дочірня» активність повинна обробити введення даних від користувача і надати результати обробки «батьківської» активності. Активності, що запускаються таким чином (за допомогою методу

startActivityForResult) повинні бути «зареєстровані» в файлі маніфесту додатка.

На відміну від методу startActivity, метод startActivityForResult вимагає явної вказівки ще одного параметра - коду запиту (request code). Цей параметр використовується викликаючою активністю для визначення того, яка саме дочірня активність завершила роботу і (можливо) надала результати:

```
private static final int BUY_BEER = 1;
Intent intent = new Intent(this, MyOtherActivity.class);
startActivityForResult(intent, BUY_BEER);
```

Коли дочірня активність готова до завершення роботи, до виклику методу finish потрібно викликати метод setResult для передачі результатів викликаючої активності.

Метод setResult отримує два параметри: код повернення і сам результат, представлений у вигляді намірів.

Код повернення (result code), що повертається з дочірної активності - це, зазвичай, або Activity.RESULT_OK, або Activity.RESULT_CANCELED. У випадку, якщо дочірня активність завершить роботу без виклику методу setResult, код повернення, переданий батьківській активності, дорівнюватиме Activity.RESULT_CANCELED.

У деяких випадках може знадобитися використовувати власні коди повернення для обробки певних ситуацій. Метод setResult в якості коду повернення сприймає будь – яке цілочисельне значення.

Намір, що повертається як результат, часто містить URI, який вказує на конкретний об'єкт даних, та / або набір додаткових значень, записаних у властивість наміру extras, що використовуються для передачі додаткової інформації.

Приклад вказівки обробника натискання на кнопку button, що встановлює результат роботи і завершує поточну активність:

```
Button button = (Button) findViewById(R.id.ok_button);
button.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
                Intent result = new Intent();
                result.putExtra("teacher name", "Mike Varakin");
                setResult(RESULT_OK, result);
                finish();}     });
```

Коли дочірня активність завершує роботу, в батьківській активності викликається обробник onActivityResult, який отримує наступні параметри:

• Request code. Використаний при запуску дочірньої активності код запиту.

• Result code. Код повернення.

• Data. Намір, що використовується для упаковки повертаються даних.

Приклад обробника onActivityResult:

```
private static final int SELECT_VICTIM = 1;
```

```
private static final int DRINK_BEER = 2;
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data)
{
super.onActivityResult(requestCode, resultCode, data);
switch (requestCode) {
        case (SELECT_VICTIM): {
                if (resultCode == Activity.RESULT_OK) {
                        String selectedVictim = data.getStringExtra("victim");
                }
                break;
        }
case (DRINK_BEER): {
        if (resultCode == Activity.RESULT_OK) {
                // Обробити результати заказаної дії
        }
        break;      }}
```

Неявні наміри

Неявні наміри використовуються для запуску активності, для виконання зазначених дій в умовах, коли невідомо, яка саме активність (і з якого додатку) буде використовуватися.

При створенні наміру, який в подальшому буде переданий методом startActivity, необхідно призначити дію (action), яку потрібно виконати, і, можливо, вказати URI даних, які потрібно обробити. Також можна передати додаткову інформацію за допомогою властивості extras наміру. Android сам знайде підходящу активність (грунтуючись на характеристиках наміру) і запустить її. Приклад неявного виклику телефонного додзвонювача:

```
Intent intent = new Intent(Intent.ACTION_DIAL,
                Uri.parse("tel:(495)502-99-11"));
startActivity(intent);
```

Для визначення того, який саме компонент повинен бути запущений для виконання дій, зазначених у намірах, Android використовує фільтри намірів (Intent Filters). Використовуючи фільтри намірів, додатки повідомляють системі, що вони можуть виконувати певні дії (action) з певними даними (data) за певних умов (category) на замовлення інших компонентів системи.

Для реєстрації компонента додатку (активності або сервісу) в якості потенційного обробника намірів, потрібно додати елемент <intent-filter> як дочірнього елемента для потрібного компонента в маніфест файлі додатку. У елемента <intent-filter> можуть бути вказані такі дочірні елементи (і відповідні атрибути у них):

• <action>. Атрибут Android: name даного елемента використовується для зазначення назви дії, що може обслуговуватися. Кожен фільтр намірів

повинен містити не менше одного вкладеного елемента <action>. Якщо не вказати дію, жоден намір не буде «проходити» через цей фільтр. У додатку головної активності в маніфест файлі повинні бути вказаний фільтр намірів з дією android.intent.action.MAIN.

• <category>. Повідомляє системі, за яких обставин необхідно обслуговуватися дію (за допомогою атрибуту android: name). Всередині <intent-filter> може бути зазначено кілька категорій. Категорія android.intent.category.LAUNCHER вимагає активності, яка бажає мати «іконку» для запуску. Активності, що запускаються за допомогою методу startActivity, зобов'язані мати категорію android.intent.category.DEFAULT

• <data>. Дає можливість вказати тип даних, які може обробляти компонент. <intent-filter> може містити кілька елементів <data>. У цьому елементі можуть використовуватися наступні атрибути:

• android: host: ім'я хоста (наприклад, www.specialist.ru)
• android: mimetype: оброблюваний тип даних (наприклад, text / html)
• android: path: «шлях» всередині URI (наприклад, / course / android)
• android: port: порт сервера (наприклад, 80)
• android: scheme: схема URI (наприклад, http)

Приклад вказівки фільтра намірів:

```
<activity android:name=".MyActivity"
        android:label="@string/app_name" >
        <intent-filter>
          <action android:name="android.intent.action.SEND" />
          <category android:name="android.intent.category.DEFAULT" />
          <data android:mimeType="text/plain" />
        </intent-filter>
</activity>
```

При запуску активності за допомогою методу startActivity неявний намір зазвичай підходить тільки одній активності. Якщо для даного наміру підходять кілька активностей, користувачеві пропонується список варіантів.

Визначення того, які активності підходять для наміру, називається Intent Resolution. Його завдання - визначити найбільш підходящі фільтри намірів, що належать компонентам встановлених додатків. Для цього використовуються наступні перевірки в зазначеному порядку:

• Перевірка дій. Після цього кроку залишаються тільки компоненти додатків, у яких в фільтрі намірів вказано дію наміру. У випадку, якщо дія в намірі відсутня, збіг відбувається для всіх Фільтрів Намірів, у яких вказано хоча б одну дію.

• Перевірка категорій. Всі категорії, наявні у намірах, повинні бути присутніми в фільтрі намірів. Якщо у намірів немає категорій, то на даному етапі йому відповідають всі фільтри намірів, за одним винятком, згадуваним вище: активності, що запускаються за допомогою методу startActivity, зобов'язані мати категорію android.intent.category.DEFAULT, оскільки наміру, використаному в цьому випадку, за замовчуванням присвоюється дана

категорія, навіть якщо розробник не вказав нічого явно. Якщо у активності присутня дія android.intent.action.MAIN і категорія android.intent.category.LAUNCHER, йому не потрібно мати категорію android.intent.category.DEFAULT.

• Перевірка даних. Тут застосовуються такі правила:

- Намір, що не містить ні URI, ні типу даних, проходить через Фільтр, якщо він теж нічого перерахованого не містить.
- Намір, який має URI, але не містить тип даних (і тип даних неможливо визначити по URI), проходить через Фільтр, якщо URI Наміри збігається з URI Фільтра. Це справедливо лише у випадку таких URI, як mailto: або tel:, які не посилаються на реальні дані.
- Намір, що містить тип даних, але не містить URI підходить тільки для аналогічних фільтрів намірів.
- Намір, що містить і тип даних, і URI (або якщо тип даних може бути обчислений з URI), проходить цей етап перевірки, тільки якщо його тип даних присутній у фільтрі. У цьому випадку URI повинен співпадати з наміром вказаним URI виду content: або file:, а у фільтрі URI не зазначений. Тобто, передбачається, що якщо у компонента в фільтрі вказаний тільки тип даних, то він підтримує URI виду content: або file:.

У випадку, якщо після всіх перевірок залишається кілька додатків, користувачеві пропонується вибрати додаток самому. Якщо підходящих додатків не знайдено, в намірі активності виникає виняток.

Збереження стану і налаштувань додатку

Оскільки життєвий цикл додатків і активностей в Android може перерватися в будь-який момент, для підвищення привабливості та зручності користувацького інтерфейсу бажано мати можливість зберігати (і відновлювати) стан активності не тільки при виході з активного стану, а й між запусками. Android пропонує для цього наступні механізми:

• Загальні налаштування (Shared preferences): простий спосіб збереження стану UI і налаштування додатку, що використовує пари ключ / значення для зберігання примітивних типів даних і забезпечення доступу до них по імені.

• Збереження стану програми: для активностей існують обробники подій, що дозволяють зберігати і відновлювати поточний стан UI, коли виходить і повертається в активний режим. Для збереження стану використовується об'єкт класу Bundle, який передається методам onSaveInstanceState (для збереження стану), onCreate і onRestoreInstanceState (для відновлення). Для доступу до даних в цьому випадку також використовуються пари ключ / значення. Обробники подій з суперкласів беруть на себе основну роботу по збереженню та відновленню виду UI, фокусу полів і т. д.

• Пряма робота з файлами. Якщо не підходять описані вище варіанти, додаток може напряму читати і писати дані з файлу. Для цього можна використовувати стандартні класи та методи Java, що забезпечують введення / виведення, так і методи openFileInput і openFileOutput, надані Android, для спрощення читання і запису потоків, що відносяться до локальних файлів.

Клас SharedPreferences пропонує методи для збереження й отримання даних, що записуються у файли, доступні по замовчуванню тільки конкретному додатку. Такий спосіб зберігання забезпечує збереження цих даних не тільки в перебігу життєвого циклу додатка, але і між запусками і навіть перезавантаженням ОС.

Для збереження даних у файлі використовується транзакційний механізм: спочатку потрібно отримати об'єкт класу SharedPreferences.Editor для конкретного файлу налаштувань, після чого за допомогою методів виду put тип цього об'єкта та встановити потрібні значення. Запис значень проводиться методом commit.

Наведемо приклад збереження даних:

```
private static final String PREFS = "PREFS";
static final String KEY_STATION = "selectedStation";
private SharedPreferences prefs;
private static final String NOTHING_SELECTED = "Ничего не выбрано";
private String selectedStation;
prefs = getSharedPreferences(PREFS, MODE_PRIVATE);
Editor editor = prefs.edit();
editor.putString(KEY_STATION, selectedStation);
editor.commit();
```

Наведемо приклад отримання даних:

```
prefs = getSharedPreferences(PREFS, MODE_PRIVATE);
selectedStation = prefs.getString(KEY_STATION, NOTHING_SELECTED);
tv.setText(selectedStation);
```

Робота з файлами

Методи openFileInput і openFileOutput дають можливість працювати тільки з файлами, що знаходяться в «персональному» каталозі додатку. Як наслідок, вказівка роздільників ("/") в імені файлу призведе до викиду винятку.

За замовчуванням файли, відкриті на запис, перезаписуються. Якщо це не те, що потрібно, при відкритті встановлюється режим MODE_APPEND.

Приклад роботи з файлами:

```
String FILE_NAME = "app_data";
// Відкриття вихідного файлового потоку
FileOutputStream fos = openFileOutput(FILE_NAME,
Context.MODE_PRIVATE);
// Відкриття вхідного файлового потоку
FileInputStream fis = openFileInput(FILE_NAME);
```

String FILE_NAME = "app_data";

Через об'єкт Context в додатку також можливий доступ до двох корисних методів:

• fileList - повертає список файлів додатку;

• deleteFile видаляє файл з каталогу додатку.

Якщо додатку необхідно мати доступ до інформації, яку незручно зберігати, наприклад, в СУБД, ці дані можна використовувати у вигляді «сирих» ресурсів, записавши їх у файли в каталозі res / raw. Яскравий приклад подібних даних - словники.

Статичні файли, як випливає з назви, доступні тільки для читання, для їх відкриття використовується метод openRawResource:

Resources res = getResources();

InputStream file = res.openRawResource(R.raw.filename);

Робота із меню в Android

Використання меню в додатках дозволяє зберегти цінний екранний простір, який в іншому випадку було б зайнято відносно рідко використовуваними елементами інтерфейсу.

Кожна активність може мати меню, що реалізують специфічні функції. Можна використовувати також контекстні меню, індивідуальні для кожного подання на екрані.

В Android реалізована підтримка триступеневої системи меню, оптимізована, в першу чергу, для невеликих екранів:

• Основне меню розміщується внизу на екрані при натисканні на кнопку «menu» пристрою. Воно може відображати текст і іконки для обмеженого (за замовчуванням, не більше шести) числа пунктів. Для цього меню рекомендується використовувати іконки з колірною гамою у вигляді відтінків сірого з елементами рельєфності. Це меню не може містити радіокнопки і чекбокси. Якщо число пунктів такого меню перевищує максимально допустиме значення, в меню автоматично з'являється пункт з написом «ще» («more»). При натисканні на нього відобразиться Розширене меню.

• Розширене меню відображає прокручуваний список, елементами якого є пункти, що не увійшли до основного меню. У цьому списку не можуть відображатися іконки, але є можливість відображення радіокнопок і чекбоксів. Оскільки не існує способу відобразити розширене меню замість основного, про зміну стану якихось компонентів програми або системи рекомендується повідомляти користувача за допомогою зміни іконок або тексту пунктів меню.

• Дочірнє меню (меню третього рівня) може бути викликано з основного або розширеного меню і відображається у спливаючому вікні. Вкладеність не підтримується, і спроба викликати з дочірнього ще одне меню призведе до викиду винятку.

При зверненні до меню викликається метод onCreateOptionsMenu активності та для появи меню на екрані його потрібно перевизначити. Даний метод отримує параметр об'єкт класу Menu, який надалі використовується для маніпуляцій з пунктами меню.

Для додавання нових пунктів в меню використовується метод add об'єкта Menu з наступними параметрами:

• Група об'єднання пунктів меню для групової обробки.

• Ідентифікатор - унікальний ідентифікатор пункту меню. Цей ідентифікатор передається оброблювачу натискання на пункт меню - методу onOptionsItemSelected.

• Порядок - значення, яке вказує порядок у якому пункти меню будуть виводитися.

• Текст - напис на пункті меню.

Після успішного створення меню метод onCreateOptionsMenu повинен повернути значення true.

Наведений нижче приклад показує створення меню з трьох пунктів з використанням строкових ресурсів:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    super.onCreateOptionsMenu(menu);
    menu.add(0, Menu.FIRST, Menu.NONE, R.string.menu_item1);
    menu.add(0, Menu.FIRST+1, Menu.NONE, R.string.menu_item2);
    menu.add(0, Menu.FIRST+2, Menu.NONE, R.string.menu_item3);
    return true; }
```

Для пошуку пунктів меню за ідентифікатором можна використовувати метод findItem об'єкта Menu.

Найбільш корисними параметрами пунктів меню є наступні:

• Короткі заголовки, які використовуються у випадку, якщо пункт може відобразитися в основному меню. Встановлюється методом setTitleCondensed об'єкта класу MenuItem:

```
menuItem.setTitleCondensed ("заголовок");
```

• Іконки - ідентифікатор Drawable, що містить потрібну картинку:

```
menuItem.setIcon (R.drawable.menu_item_icon);
```

• Оброблювач вибору пункту меню який можна встановити, але не рекомендується з міркувань підвищення продуктивності, краще використовувати обробник всього меню (onOptionsItemSelected). Тим не менш, приклад:

```
menuItem.setOnMenuItemClickListener(new OnMenuItemClickListener() {
    public boolean onMenuItemClick(MenuItem _menuItem) {
    // обробити вибір пункту
    return true;
    }
});
```

• Намір - автоматично передається методу startActivity, якщо відбувається натиснення на пункт меню, який не було оброблено onMenuItemClickListener і onOptionsItemSelected:

menuItem.setIntent(new Intent(this, MyOtherActivity.class));

Безпосередньо перед виведенням меню на екран викликається метод onPrepareOptionsMenu поточної Активності, і перевизначаючи його можна динамічно змінювати стан пунктів меню: дозволяти / забороняти, робити невидимим, змінювати текст і т. д.

Для пошуку пункту меню, що підлягає модифікації можна використовувати метод findItem об'єкта Menu, переданого як параметр:

```
@Override
public boolean onPrepareOptionsMenu(Menu menu) {
    super.onPrepareOptionsMenu(menu);
    MenuItem menuItem = menu.findItem(MENU_ITEM);
    //модифікувати пункт меню....
    return true;
}
```

Android дозволяє обробляти всі пункти меню (вибирати їх) одним обробником onOptionsItemSelected. Вибраний пункт меню передається цьому оброблювачу в якості об'єкта класу MenuItem.

Для реалізації потрібної реакції на вибір пункту меню потрібно визначити, що саме було обрано. Для цього використовується метод getItemId переданий як параметр об'єкта, а отриманий результат порівнюється з ідентифікаторами, використаними при додаванні пунктів у меню в методі onCreateOptionsMenu.

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        // Перевірити кожен відомий пункт
        case (MENU_ITEM):
            // зробити що-небудь...
            return true;
    }
    return super.onOptionsItemSelected(item);
}
```

При появі на екрані, дочірні та контекстні меню виглядають однаково, у вигляді плаваючих вікон, але при цьому створюються по-різному.

Для створення дочірніх меню використовується метод addSubMenu об'єкта класу Menu:

```
SubMenu sub = menu.addSubMenu(0, 0, Menu.NONE, "дочірнє меню");
sub.setHeaderIcon(R.drawable.icon);
sub.setIcon(R.drawable.icon);
MenuItem submenuItem = sub.add(0, 0, Menu.NONE, "пункт дочірнього меню");
```

Як вже було сказано вище, вкладені дочірні меню Android не підтримує.

Найбільш поширеним способом створення контекстного меню в Android є перевизначення методу onCreateContextMenu активності:

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v,
        ContextMenu.ContextMenuInfo menuInfo) {
super.onCreateContextMenu(menu, v, menuInfo);
menu.setHeaderTitle("Конекстне меню");
menu.add(0, Menu.FIRST, Menu.NONE, "Пункт 1");
menu.add(0, Menu.FIRST+1, Menu.NONE, "Пункт 2");
menu.add(0, Menu.FIRST+2, Menu.NONE, "Пункт 3");}
```

Реєстрація обробника контекстного меню для потрібних View здійснюється за допомогою методу registerForContextMenu:

```
@Override
public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        tv = (TextView) findViewById(R.id.text_view);
        registerForContextMenu(tv);
}
```

Приклад обробника:

```
@Override
public boolean onContextItemSelected(MenuItem item) {
        switch (item.getItemId()) {
        case DELETE_ID:
                AdapterContextMenuInfo info = (AdapterContextMenuInfo)
        item
                                .getMenuInfo();
                db.deleteItem(info.id);
                populate();
                return true;
        }
        return super.onContextItemSelected(item);
}
```

Часто буває найзручніше описувати меню, в тому числі ієрархічні, у вигляді ресурсів. Про переваги такого підходу говорилося вище.

Меню традиційно описуються і зберігаються в каталозі res / menu проекту. Всі ієрархії меню (якщо є ієрархічні меню) повинні знаходитися в окремих файлах, а ім'я файлу буде використовувати як ідентифікатор ресурсу. Кореневим елементом файлу повинен бути тег <menu>, а пункти меню описуються тегом <item>. Властивості пунктів меню описуються відповідними атрибутами:

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
    <item
        android:id="@+id/item01"
        android:icon="@drawable/menu_item"
        android:title="Пункт 1">
    </item>
    <item
        android:id="@+id/item02"
        android:checkable="true"
        android:title="Пункт 2">
    </item>
    <item
        android:id="@+id/item03"
        android:title="Пункт 3">
    </item>
    <item
        android:id="@+id/item04"
        android:title="Дочірнє меню 1">
        <menu>
        <item
            android:id="@+id/sub1item01"
            android:title="Пункт дочірнього меню 1">
        </item>
    </menu>
</item>
<item
    android:id="@+id/item05"
    android:title="Дочірнє меню 2">
    <menu>
        <item
            android:id="@+id/sub2item01"
            android:title="Пункт дочірнього меню 2">
        </item>
    </menu>
  </item>
</menu>
```

Для створення об'єктів Menu з ресурсів в події onCreateOptionsMenu і onCreateContextMenu використовується метод inflate об'єкта типу MenuInflater:

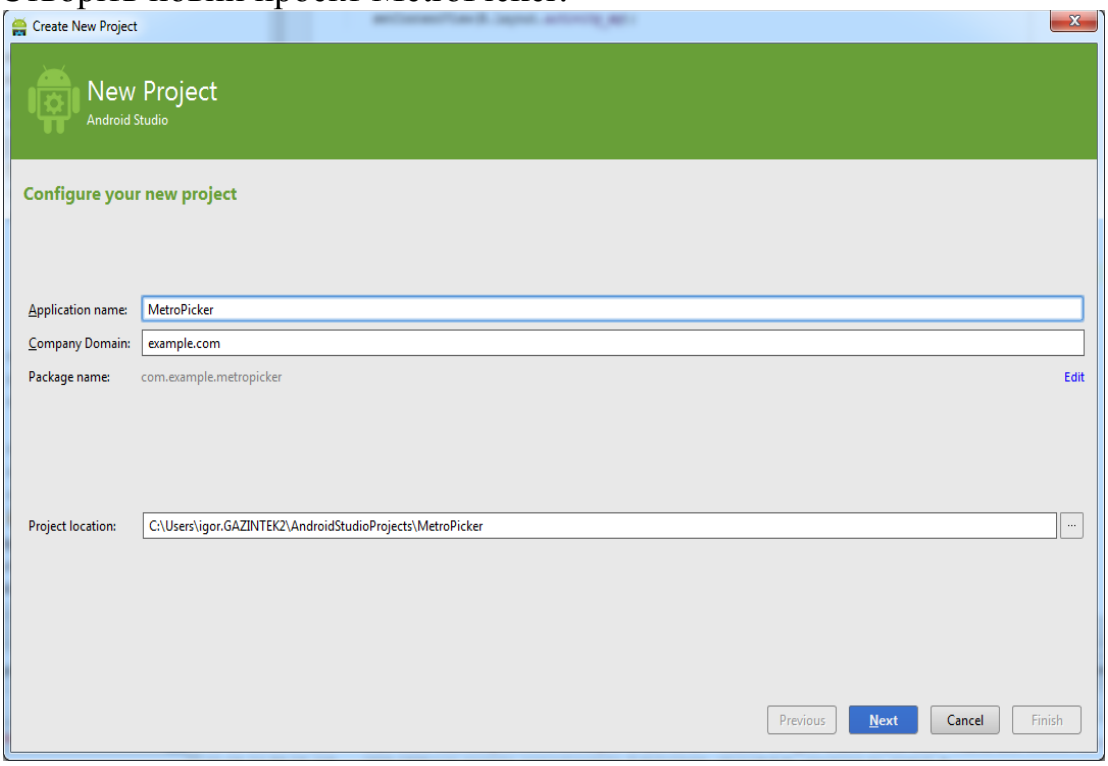
```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    super.onCreateOptionsMenu(menu);
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.menu1, menu);
```

```
        return true;
    }
```

Порядок виконання роботи

Завдання 1 «Виклик активності за допомогою явного наміру та отримання результатів роботи»

1. Створіть новий проект MetroPicker.



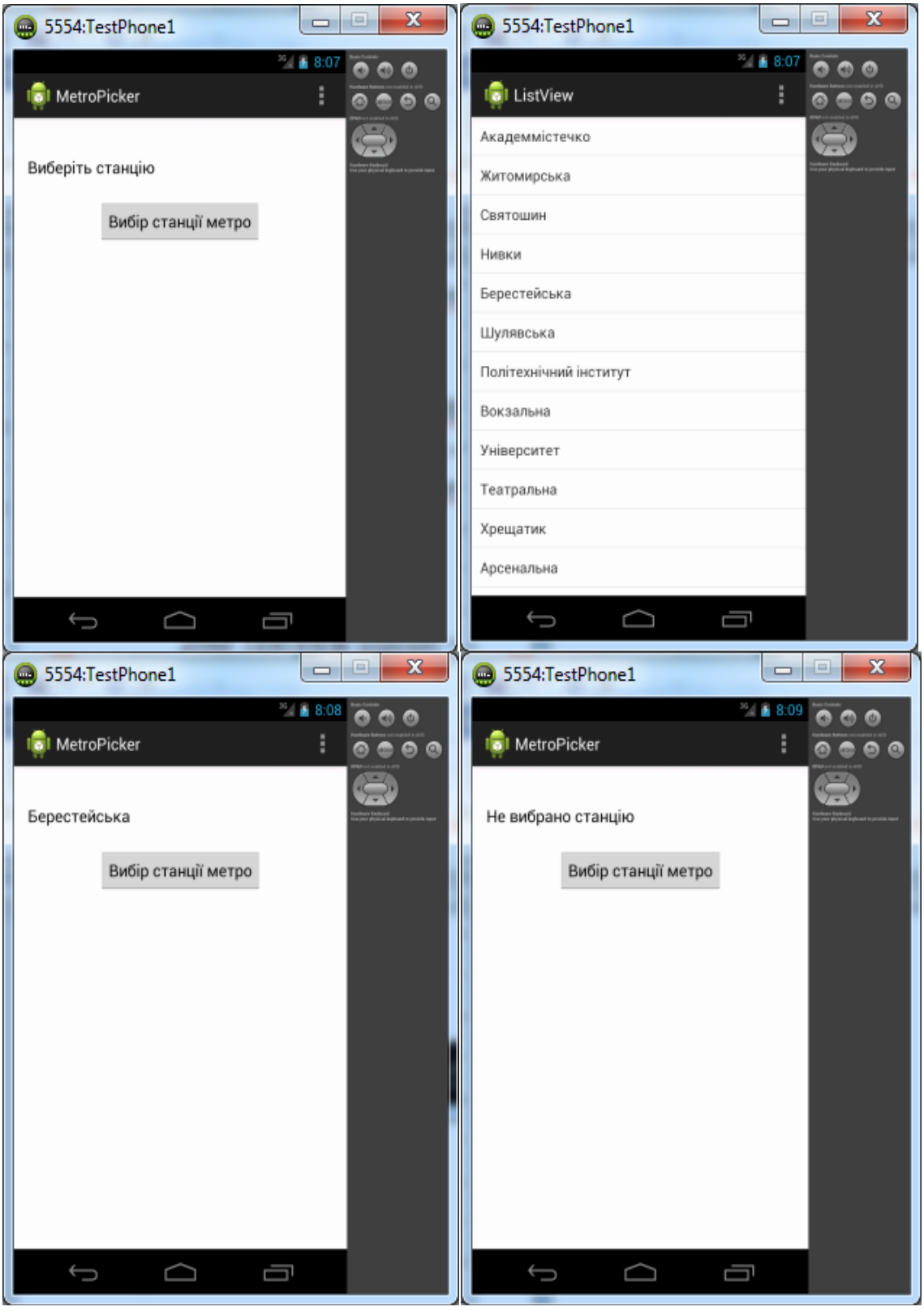
2. Додайте допоміжну активність ListView Activity для відображення і вибору станцій метро, як заготовки використовуйте результати попередніх практичних робіт.

3. Відредагуйте файл розмітки res / layout / main.xml: додайте кнопку вибору станції метро, присвоївши ідентифікатори віджетам TextView і Button для того, щоб на них можна було посилатися в коді.

4. Встановіть обробник натискання на кнопку в головній активності для виклику списку станції і вибору потрібної станції.

5. Напішіть потрібний обробник для установки обраної станції метро в віджет TextView батьківської активності (метод setText віджета TextView дозволяє встановити відображуваний текст). Не забудьте обробити ситуацію, коли користувач натискає кнопку «Назад» (в цьому випадку «ніякої станції не вибрано» і головна активність повинна сповістити про це користувача).

6. Впевніться в працездатності створеного додатка, перевіривши реакцію на різні дії потенційних користувачів.



Завдання 2 «Використання неявних намірів»

1. Змініть проект MetroPicker так, щоб для запуску активності ListView Activity використовувався неявний намір з дією (action), визначеною у вашому додатку і має значення "com.example.metropicker. intent.action.PICK_METRO_STATION".

2. Перевірте роботу програми. Визначення наміру, що викликав запуск активності. Оскільки об'єкти типу Intent служать, в тому числі, для передачі інформації між компонентами одного або декількох додатків, може виникнути необхідність у роботі з об'єктом наміру, що викликав активність.

Для отримання доступу до цього об'єкта використовується метод getIntent. приклад:

Intent intent = getIntent();

Маючи даний об'єкт, можна отримати доступ до інформації, що міститься в ньому:

• метод getAction повертає дію наміру;

• метод getData повертає дані наміру (зазвичай URI);

• набір методів для різних типів виду getТИПExtra дозволяє отримати доступ до типізованих значеннь, що зберігаються у властивості extras наміру.

Приклади:

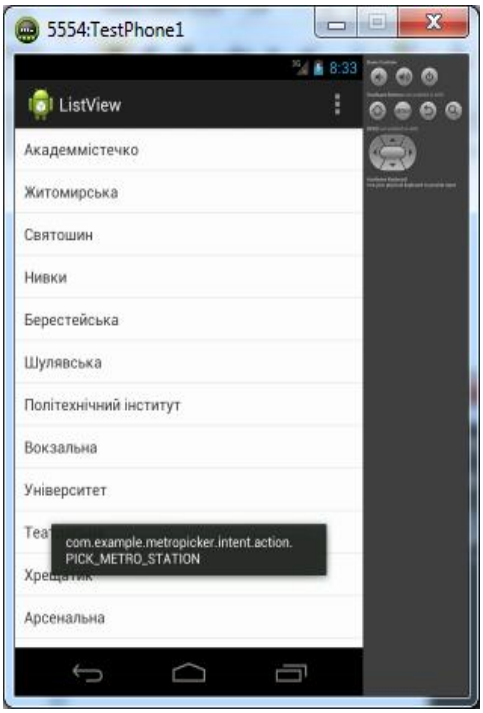
String action = intent.getAction();

Uri data = intent.getData();

Задання 3 «Одержання даних з наміру»

1. Модифікуйте методи onCreate з попередньої лабораторної роботи так, щоб за допомогою Toast вони показували дію викликаних їх намірів.

2. Перевірте роботу програми:



Завдання 4 «Використання SharedPreferences для збереження стану»

1. Модифікуйте методи onCreate і onActivityResult проекту MetroPicker для збереження обраної станції метро між запусками програми.

2. Перевірте працездатність програми.

Завдання 5 «Використання SharedPreferences для збереження налаштувань»

1. Модифікуйте проект Controls Sample так, щоб стан керуючих елементів зберігався і відновлювався між запусками програми.

2. Перевірте працездатність програми.

Завдання 6 «Створення і використання меню»

Модифікуйте проект MetroPicker наступним чином:

1. Додайте головне меню в Активність, відображаючу список станцій метро. У меню має бути один пункт «повернутися». Меню створіть динамічно в коді, без використання строкових ресурсів.

2. Динамічно створіть контекстне меню для TextView, що буде відображати обрану станцію метро головної активності. Вибір пункту меню повинен скидати обрану станцію.

3. Для головної активності створіть основне меню з двох пунктів: «скинути» і «вийти». Реалізуйте потрібні функції при виборі цих пунктів. Повторіть реалізацію п.п. 1, 2 і 3 за допомогою ресурсів, що описують меню.

Завдання для виконання

Виконайте завдань до виконання у відповідності до теоретичних відомостей поданих вище.

Вимоги до звіту

В звіт по роботі включіть текст програм, що реалізують описані завдання та екранні форми, що відображають їх виконання.

Контрольні запитання

1. Для чого використовують меню у додатку?
2. Який метод використовується для створння дочірніх меню?
3. Чи вірно це? Дані типу Calendar можна безпосередньо зберегти у екземплярі класу SharedPreferences.
4. Які типи даних підтримуються класом SharedPreferences?
5. Яка різниця між явним і неявним наміром?

ПРАКТИЧНА РОБОТА №7

Тема роботи: Робота з базами даних в ОС Android.

Мета роботи: Опанувати принципи роботи з СУБД SQLite без застосування спеціальних класів-адаптерів.

Теоретичні відомості

Робота з базами даних в Android

Механізм роботи з базами даних в Android дозволяє зберігати й обробляти структуровану інформацію. Будь-який додаток може створювати свої власні бази даних, над якими він буде мати повний контроль.

В Android використовується бібліотека SQLite, що представляє із себе реляційну СУБД, яка володіє наступними особливостями: вільно поширювана, підтримується стандартна мова запитів і транзакції, однорівнева, відмовостійка.

Курсори (Cursor) і ContentValues

Запити до СУБД повертають об'єкти типу Cursor. Для економії ресурсів використовується підхід, коли при отриманні даних не повертаються копії їх значень з СУБД, а створюється Cursor, що забезпечує навігацію і доступ до затребуваного набору вихідних даних. Методи об'єкта Cursor надають різні можливості навігації:

- moveToFirst
- moveToNext
- moveToPrevious
- getCount
- getColumnIndexOrThrow
- getColumnName
- getColumnNames
- moveToPosition
- getPosition

При додаванні даних в таблиці СУБД застосовуються об'єкти класу ContentValues. Кожен такий об'єкт містить дані одного рядка в таблиці і, по суті, є асоціативним масивом з іменами стовпців і відповідними значеннями.

Робота з СУБД SQLite

При створенні додатків, що використовують СУБД, у багатьох випадках зручно застосовувати інструменти, що називаються ORM (Object-Relationship Mapping), які відображають дані з однієї або декількох таблиць на об'єкти використовуваної мови програмування. Крім того, ORM дозволяють абстрагуватися від конкретної реалізації і структури таблиць та беруть на себе обов'язки по взаємодії з СУБД. На жаль, в силу обмеженості ресурсів мобільної платформи ORM зараз в Android практично не застосовується. Тим не менш, розумним підходом при розробці програми

буде інкапсуляція всіх взаємодій з СУБД в одному класі, методи якого будуть надавати необхідні послуги інших компонентів програми.

Доброю практикою є створення допоміжного класу, що бере на себе роботу з СУБД. Даний клас зазвичай інкапсулює взаємодії з базою даних, надаючи інтуїтивно зрозумілий строго типізований спосіб видалення, додавання і зміни об'єктів. Такий адаптер бази даних також повинен обробляти запити до БД і перевизначати методи для відкриття, закриття та створення бази даних. Його також зазвичай використовують як зручне місце для зберігання статичних констант, що відносяться до бази даних, таких, наприклад, як імена таблиць і полів. Нижче показаний приклад каркаса для реалізації подібного адаптера:

```
public class SampleDBAdapter {
private static final String DATABASE_NAME = "SampleDatabase.db";
private static final String DATABASE_TABLE = "SampleTable";
private static final int DATABASE_VERSION = 1;
// Ім’я поля індекса для
public static final String KEY_ID = "_id";
// Назва і номер п/п (індекс) кожного поля
public static final String KEY_NAME = "name";
public static final int NAME_COLUMN = 1;
// Для кожного поля опишіть константи аналогічним чином...
// SQL-запит для створення БД
private static final String DATABASE_CREATE = "create table "
        + DATABASE_TABLE + " (" + KEY_ID
        +" integer primary key autoincrement, " + KEY_NAME
        + " text not null);";
// Змінна для збережена об’єкта БД
private SQLiteDatabase db;
// Контекст додатку для
private final Context context;
// Экземпляр допоміжного класу для відкриття і поновлення БД
private myDbHelper dbHelper;
// Конструктор
public SampleDBAdapter(Context _context) {
    context = _context;
    dbHelper = new myDbHelper(context, DATABASE_NAME, null,
            DATABASE_VERSION);
}
// «Відкривачка» БД
public SampleDBAdapter open() throws SQLException {
    try {
        db = dbHelper.getWritableDatabase();
    } catch (SQLiteException e) {
        db = dbHelper.getReadableDatabase();
```

```
            }
            return this;
        }
        // Метод для закритя БД
        public void close() {
            db.close();
        }
        public long insertEntry(SampleObject _SampleObject) {
            // Тут створюється об'єкт ContentValues, що містить
            // потрібні поля і проводиться вставка
            return index;
        }
        // Метод для видалення рядка таблиці за індексом
        public boolean removeEntry(long _rowIndex) {
            return db.delete(DATABASE_TABLE, KEY_ID + "=" +
                    _rowIndex, null) > 0;
        }
        // Метод для отримання всіх даних.
        // Повертає курсор, який можна використовувати для
        // прив'язки до адаптерів  типу SimpleCursorAdapter
        public Cursor getAllEntries() {
        return db.query(DATABASE_TABLE, new String[] { KEY_ID, KEY_NAME
        },
                    null, null, null, null, null);
        }
        // Повертає екземпляр об'єкта за індексом
        public SampleObject getEntry(long _rowIndex) {
            // Отримайте курсор, який вказує на потрібні дані з БД
            // і створіть новий об'єкт, використовуючи ці дані
            // Якщо нічого не знайдено, поверніть null
            return objectInstance;
        }
        // Змінює об'єкт за індексом
        public boolean updateEntry(long _rowIndex, SampleObject _SampleObject)
{
        // створіть об'єкт ContentValues на основі властивостей SampleObject
        // і використовуйте його для оновлення рядка в таблиці
            return true; // Якщо вдалося оновити, інакше false :)
        }
        // Допоміжний клас для відкриття і поновлення БД
        private static class myDbHelper extends SQLiteOpenHelper {
            public myDbHelper(Context context, String name,
                    CursorFactory factory, int version) {
                    super(context, name, factory, version);
```

```
        }
    // Викликається при необхідності створення БД
    @Override
    public void onCreate(SQLiteDatabase _db) {
        _db.execSQL(DATABASE_CREATE);
    }
    // Викликається для поновлення БД, коли поточна версія БД
    // в додатку новіша, ніж у БД на диску
    @Override
    public void onUpgrade(SQLiteDatabase _db, int _oldVersion,
            int _newVersion) {
        // Видача повідомлення в журнал, корисно при відлагодженні
        Log.w("TaskDBAdapter", "Upgrading from version " + _oldVersion
                + " to " + _newVersion
                + ", which will destroy all old data");
        // Оновлюємо БД до нової версії.
        // У найпростішому випадку вбиваємо стару БД
        // і заново створюємо нову.
        // У реальному житті варто подумати про користувачів
        // вашої програми і їх реакцію на втрату старих даних.
        _db.execSQL("DROP TABLE IF EXISTS " + DATABASE_TABLE);
        onCreate(_db); } } }
```

Робота з СУБД без адаптера

При бажанні використовувати клас SQLiteOpenHelper (і взагалі адаптер БД) можна використати метод openOrCreateDatabase контексту програми:

```
private static final String DATABASE_NAME = "myDatabase.db";
private static final String DATABASE_TABLE = "mainTable";
private static final String DATABASE_CREATE = "create table "
        + DATABASE_TABLE + " ( _id integer primary key
    autoincrement,"
        + "column_one text not null);";
SQLiteDatabase myDatabase;
private void createDatabase() {
    myDatabase = openOrCreateDatabase(DATABASE_NAME,
    MODE_PRIVATE, null);
    myDatabase.execSQL(DATABASE_CREATE);
}
```

У цьому випадку можна працювати з базою даних за допомогою, наприклад, методу execSQL примірника БД, як показано у прикладі вище.

При роботі з базами даних в Android слід уникати зберігання BLOB'ів в таблицях через різке падіння ефективності роботи.

Як показано в прикладі адаптера БД, для кожної таблиці рекомендується створювати автоікрементне поле _id, яке буде унікальним

індексом для рядків. Якщо ж планується делегувати доступ до БД за допомогою контент-провайдерів, таке поле є обов'язковим.

Виконання запитів для доступу до даних

Для підвищення ефективності використання ресурсів мобільної платформи запити до БД повертають об'єкт типу Cursor, що в подальшому використовується для навігації і отримання значень полів. Виконання запитів здійснюється за допомогою методу query примірника БД, параметри якого дозволяють гнучко керувати критеріями вибірки:

```
// Отримати поля індексу, а також перше і третє з таблиці, без дублікатів
String[] result_columns = new String[] {KEY_ID, KEY_COL1, KEY_COL3};
Cursor allRows = myDatabase.query(true, DATABASE_TABLE, result_columns,
null, null, null, null, null, null);
// Отримати всі поля для рядків, де третє поле рівне необхідному значенню,
// результат відсортувати по п’ятому полю
String where = KEY_COL3 + "=" + requiredValue;
String order = KEY_COL5;
Cursor myResult = myDatabase.query(DATABASE_TABLE, null, where,
null, null, null, order);
```

Для отримання результатів запиту необхідно встановити курсор на потрібний рядок за допомогою методів виду moveTo. Після цього використовуються типізовані методи getТип, які отримують в якості параметра індекс (номер) поля в рядку. Як правило, ці значення є статичними константами адаптера БД:

```
String columnValue = myResult.getString(columnIndex);
```

У прикладі нижче показано, як можна підсумувати всі поля (типу float) з результатів виконання запиту (і отримати середню суму):

```
int KEY_AMOUNT = 4;
Cursor myAccounts = myDatabase.query("my_bank_accounts", null, null,
null, null, null, null);
float totalAmount = 0f;
// Переконуємося, що курсор не порожній
if (myAccounts.moveToFirst()) {
        // Проходимся по кожному рядку
        do {
                float amount = myAccounts.getFloat(KEY_AMOUNT);
                totalAmount += amount;
        } while(myAccounts.moveToNext());
}
float averageAmount = totalAmount / myAccounts.getCount();
```

Зміна даних в БД

У класі SQLiteDatabase, що містить методи для роботи з БД, є методи insert, update і delete, які інкапсулюють оператори SQL, необхідні для виконання відповідних дій. Крім того, метод execSQL дозволяє виконати будь допустимий код SQL (якщо ви, наприклад, захочете збільшити частку ручної праці при створенні програми). Слід пам'ятати, що при наявності активних курсорів після будь-якої зміни даних слід викликати метод refreshQuery для всіх курсорів, які мають відношення до змінюваних даних (таблиць).

Для вставлення рядків метод insert повинен отримувати (крім інших параметрів) об'єкт ContentValues, що містить значення полів вставленого рядка і повертає значення індексу:

```
ContentValues newRow = new ContentValues();
// Виконаємо для кожного потрібного поля в рядку
newRow.put(COLUMN_NAME, columnValue);
db.insert(DATABASE_TABLE, null, newRow);
```

При оновленні рядків БД також використовується ContentValues, що містить поля, які потрібно змінити. Параметр where вказує, які саме рядки потрібно змінити та має стандартний для SQL вигляд:

```
ContentValues updatedValues = new ContentValues();
// Повторюємо для всіх потрібних полів
updatedValues.put(COLUMN_NAME, newValue);
// вказуємо умову
String where = COLUMN_NAME + "=" + "'Бармаглот'";
// Обновляємо
db.update(DATABASE_TABLE, updatedValues, where, null);
```

Метод update повертає кількість змінених рядків. Якщо в якості параметра where передати null, будуть змінені всі рядки таблиці.

Видалення рядків Виконує схожим на update чином:

```
db.delete(DATABASE_TABLE, KEY_ID + "=" + rowId, null);
```

Порядок виконання роботи

Завдання 1 «Робота SQLite без застосування спеціальних класів-адаптерів»

1. Створіть новий проект SQLTest, головна активність якого буде розширювати клас ListActivity.

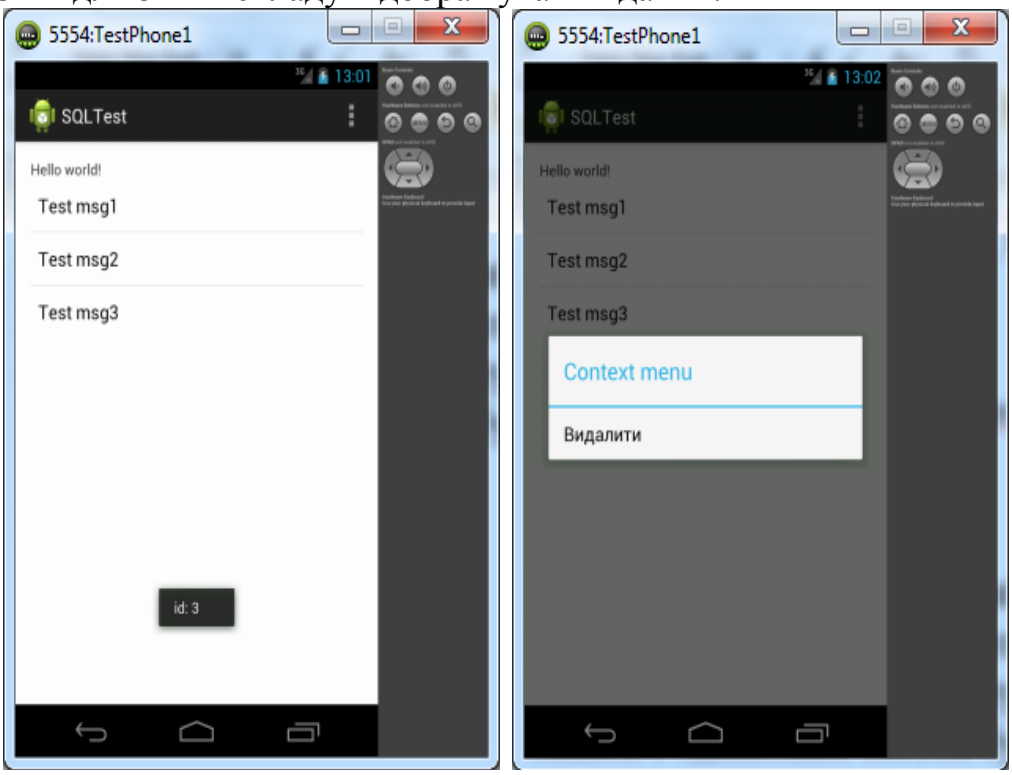
2. В методі onCreate головної активності зробіть відкриття або створення БД, використовуючи в якості основи інформацію з пункту «Робота з СУБД без адаптера» і методу onUpgrade допоміжного класу з попереднього йому пункту. Після створення БД створіть таблицю з будь-яким полем (плюс індекс), в яку додайте 3..5 унікальних записів зі строковим полем. Індекс повинен інкрементуватись автоматично.

3. В цьому ж методі зробіть запит до таблиці, який одержує всі рядки і за допомогою наявного курсора запишіть дані в масив рядків.

4. За допомогою додаткової розмітки і адаптера ArrayAdapter відобразіть отримані з СУБД дані на екрані активності.

5. Добавте обробник кліку на елемент списку, щоб при кліці запитували з БД індекс елементу з цим рядком і відобразіть його за допомогою Toast.

6. Добавте контекстне меню до елементів списку, що містить пункт «видалити» і реалізуйте видалення. Після видалення повинні проводиться дії з п.п. 3 і 4 для зміни складу відображуваних даних.



SimpleCursorAdapter дозволяє прив'язати курсор до ListView. використовуючи опис розмітки для відображення рядків і полів. Для однозначного визначення того, в яких елементах розмітки які поля, одержувані через курсор, слід відображати, використовуються два масиви: строковий з іменами полів рядків, і цілочисельний з ідентифікаторами елементів розмітки:

```
Cursor cursor = [ . . . запит до БД . . . ];
String[] fromColumns = new String[] {KEY_NAME, KEY_NUMBER};
int[] toLayoutIDs = new int[] { R.id.nameTextView, R.id.numberTextView};
SimpleCursorAdapter myAdapter;
myAdapter = new SimpleCursorAdapter(this, R.layout.item_layout, cursor,
            fromColumns, toLayoutIDs);
myListView.setAdapter(myAdapter);
```

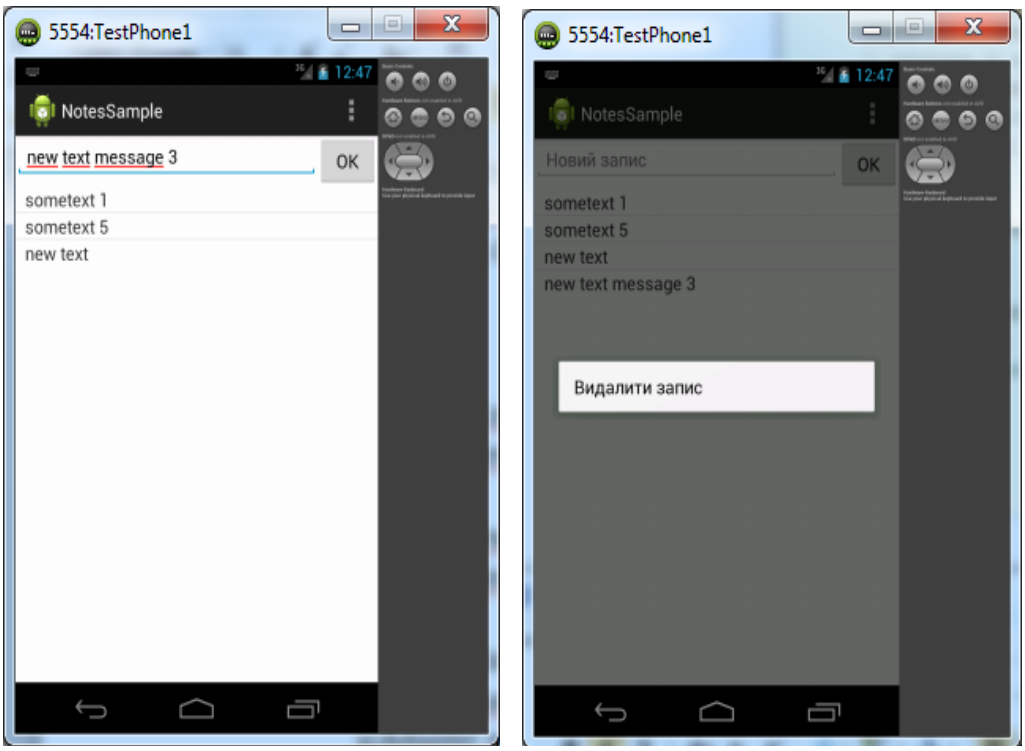
Завдання 2 «Робота SQLite з класом-адаптером»

1. Створіть новий проект NotesSample.

2. Для простоти використання всі дії із записами будуть проводитися в рамках однієї активності, тому використовуйте максимально спрощений інтерфейс, як показано на наступній сторінці. Для реалізації можна скористатися такою розміткою у файлі res / layout / main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:orientation="vertical" >
<RelativeLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" >
    <Button
        android:id="@+id/save_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:text="@android:string/ok" />
    <EditText
        android:id="@+id/edit_text"
        android:layout_width="fill_parent"
        android:layout_toLeftOf="@id/save_button"
        android:layout_height="wrap_content"
        android:hint = "Новий запис" />
</RelativeLayout>
<ListView
    android:id="@+id/myListView"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />
</LinearLayout>
```

3. Реалізуйте збереження записів після натискання на кнопку, а видалення через контекстне меню. Робота з СУБД повинна здійснюватися з використанням адаптера. Приклад обробника вибору пункту контекстного меню приведений у відповідному розділі даних методичних вказівок.



Завдання для виконання

Виконайте завдань до виконання у відповідності до теоретичних відомостей поданих вище.

Вимоги до звіту

В звіт по роботі включіть текст програм, що реалізують описані завдання та екранні форми, що відображають їх виконання.

Контрольні запитання

1. Як працює база даних SQLite?
2. Чому SQLite настільки популярна?
3. За допомогою якого метода можна закрити БД і для чого це роблять?
4. Яку роботу виконує клас SimpleCursorAdapter?
5. Наведіть приклад оновлення та видалення рядків з БД.

ПРАКТИЧНА РОБОТА №8

Тема роботи: Робота із контент-провайдерами.

Мета роботи: Отримати практичні навички створення власних контент-провайдерів.

Теоретичні відомості

Контент-провайдери надають інтерфейс для публікації та використання структурованих наборів даних, що базуються на URI з використанням простої схеми content: //. Їх використання дозволяє відокремити код додатків від даних, роблячи програму менш чутливою до змін в джерелах даних.

Для взаємодії з контент-провайдером використовується унікальний URI, який зазвичай формується таким чином:

content://<домен-розробника>.provider.<імя додатку>/<шлях до даних>

Класи, що реалізують контент-провайдери, найчастіше мають статичну строкову константу *CONTENT_URI*, яка використовується для звернення до даного контент-провайдеру.

Контент-провайдери є єдиним способом доступу до даних інших додатків і використовуються для отримання результатів запитів, оновлення, додавання і видалення даних. Якщо у програми є потрібні повноваження, вона може запитувати і модифікувати відповідні дані, що належать іншому додатку, у тому числі дані стандартних БД Android. У загальному випадку, контент-провайдери слід створювати тільки тоді, коли потрібно надати іншим програмам доступ до даних застосунку. В інших випадках рекомендується використовувати СУБД (SQLite). Тим не менш, іноді контент-провайдери використовуються всередині однієї програми для пошуку та оброблення специфічних запитів до даних.

Використання контент-провайдерів

Для доступу до даних якого-небудь контент-провайдера використовується об'єкт класу ContentResolver, який можна отримати за допомогою методу getContentResolver контексту програми для зв'язку з постачальником в якості клієнта:

```
ContentResolver cr = getApplicationContext (). GetContentResolver ();
```

Об'єкт ContentResolver взаємодіє з об'єктом контент-провайдера, відправляючи йому запити клієнта. Контент-провайдер обробляє запити і повертає результати обробки.

Контент-провайдери представляють свої дані у вигляді однієї або декількох таблиць подібно таблицями реляційних БД. Кожен рядок при цьому є окремим «об'єктом» з властивостями, зазначеними у відповідних іменованих полях. Як правило, кожен рядок має унікальний цілочисельний індекс із ім'ям «_id», який служить для однозначної ідентифікації необхідного об'єкта.

Контент-провайдери, зазвичай надають мінімум два URI для роботи з даними: один для запитів, що вимагають всі дані відразу, а інший - для звернення до конкретного «рядка». В останньому випадку в кінці URI додається / <номер-рядка> (який збігається з індексом «_id»).

Запити на отримання та зміну даних

Запити на отримання даних схожі на запити до БД, при цьому використовується метод query об'єкта ContentResolver. Відповідь також приходить у вигляді курсору, «націленого» на результуючий набір даних (вибрані рядки таблиці):

```
ContentResolver cr = getContentResolver();
// отримати дані всіх контактів
Cursor c = cr.query(ContactsContract.Contacts.CONTENT_URI, null, null,
        null, null);
// отримати всі рядки, де третє поле має конкретне
// значення і впорядкувати за п'ятим  полем
String where = KEY_COL3 + "=" + requiredValue;
String order = KEY_COL5;
Cursor someRows = cr.query(MyProvider.CONTENT_URI, null, where,
null, order);
```

Витягання даних-результатів запиту за допомогою курсору було розглянуто раніше.

Для зміни даних застосовуються методи insert, update і delete об'єкта ContentResolver. Для масової вставки також існує метод bulkInsert.

Приклад додавання даних:

```
ContentResolver cr = getContentResolver();
ContentValues newRow = new ContentValues();
// повторюємо для кожного поля в рядку:
newRow.put(COLUMN_NAME, newValue);
Uri myRowUri = cr.insert(SampleProvider.CONTENT_URI, newRow);
// масова вставка:
ContentValues[] valueArray = new ContentValues[5];
// тут заповнюємо масив
// робимо вставку
int count = cr.bulkInsert(MyProvider.CONTENT_URI, valueArray);
```

При вставці одного елемента метод *insert* повертає URI вставленого елемента, а при масовій вставці повертається кількість вставлених елементів.

Приклад видалення:

```
ContentResolver cr = getContentResolver();
// видалення конкретного рядка
cr.delete(myRowUri, null, null);
// видалення декількох рядків
String where = "_id < 5";
cr.delete(MyProvider.CONTENT_URI, where, null);
```

Приклад зміни:

```
ContentValues newValues = new ContentValues();
newValues.put(COLUMN_NAME, newValue);
String where = "_id < 5";
getContentResolver().update(MyProvider.CONTENT_URI, newValues,
where, null);
```

Створення контент-провайдерів

Для створення власного контент-провайдера потрібно розширити клас ContentProvider і перевизначити метод onCreate, щоб проініціалізувати джерело даних, яке потрібно опублікувати. Каркас для класу показаний нижче:

```
public class NewProvider extends ContentProvider {
public final static Uri CONTENT_URI=Uri.parse("URI провайдера");
@Override
public int delete(Uri uri, String selection, String[] selectionArgs) {
        // видалення даних
        return 0;
}
@Override
public String getType(Uri uri) {
        // Повертає тип MIME для зазначених об'єктів (об'єкта)
        return null;
}
@Override
public Uri insert(Uri uri, ContentValues values) {
        // Додавання даних
        return null;
}
@Override
public boolean onCreate() {
        // Ініціалізація джерела даних
        return false;
}
@Override
public Cursor query(Uri uri, String[] projection, String selection,
                String[] selectionArgs, String sortOrder) {
        // Стандартний запит
        return null;
}
@Override
public int update(Uri uri, ContentValues values, String selection,
                String[] selectionArgs) {
        // Обновлення данних
```

```
        return 0;
    }
}
```

Традиційно URI повинні бути представлені двома способами, одним - для доступу до всіх значень певного типу, інший - для вказівки на конкретний екземпляр даних.

Наприклад, URI content: //com.android.provider.metropicker/stations міг би використовуватися для отримання списку всіх станцій (метро), а content: //com.android.provider.metropicker/stations/17 - для станції з індексом 17 .

При створенні контент-провайдера зазвичай застосовується статичний об'єкт класу UriMatcher, який служить для визначення деталей запиту до контент-провайдеру. Використання UriMatcher дозволяє «розвантажити» логіку програми і уникнути множинного порівняння строкових об'єктів за рахунок централізованого «відображення» URI різних видів на цілочисельні константи. Особливо він корисний у випадках, коли створюваний контент-провайдер обслуговує різні URI для доступу до одного і того ж джерела даних. Використовувати UriMatcher дуже просто: всередині класу контент-провайдера можна додати такий код:

```
// Константи для різних типів запитів
private static final int ALL_STATIONS = 1;
private static final int SINGLE_STATION = 2;
private static final UriMatcher uriMatcher;
// заповнення UriMatcher'a
// Якщо URI закінчується на / stations - це запит про всі станції
// Якщо на stations / [ID] - про конкретну станцію
static {
    uriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
    uriMatcher.addURI("com.example.provider.metropicker", "stations",
            ALL_STATIONS);
    uriMatcher.addURI("com.example.provider.metropicker",
"stations/#",
            SINGLE_STATION);
}
```

У подальшому отримані в запиті до контент-провайдеру URI перевіряється в методах класу наступним чином:

```
@Override
public Cursor query(Uri uri, String[] projection, String selection,
        String[] selectionArgs, String sortOrder) {
    switch (uriMatcher.match(uri)) {
    case ALL_STATIONS:
        // Повернути курсор, який вказує на вибірку з  усіма
    станціями
    case SINGLE_STATION:
        // Витягнути ID станції з URI:
```

```
            String _id = uri.getPathSegments().get(1);
            // Повернути курсор, який вказує на вибірку з однією
        станцією.
        }
        return null;
    }
```

При наповненні об'єкта UriMatcher шаблонами можуть застосовувати в «#» і «*» в якості спеціальних символів: # в шаблоні збігається з будь-яким числом, а * - з будь-яким текстом.

Метод getType класу ContentProvider зазвичай повертає один тип даних для масової вибірки, а інший - для одиночних записів. У нашому випадку це могло б виглядати так:

```
    @Override
    public String getType(Uri uri) {
        switch (uriMatcher.match(uri)) {
        case ALL_STATIONS:
            return "vnd.com.example.cursor.dir/station";
        case SINGLE_STATION:
            return "vnd.com.example.cursor.item/station";
        default:
            throw new IllegalArgumentException("Unsupported URI: " +
            uri);
        }
    }
```

Для того, щоб Android знав про існування і міг використовувати (через ContentResolver) ваш контент-провайдер, його потрібно описати в маніфесті додатка. Мінімальний опис виглядає так:

```
    <provider
    android:name=".NewProvider"
    android:authorities="com.android.provider.metropicker" >
    </provider>
```

В даному випадку зазначені тільки обов'язкові атрибути елемента <provider>: ім'я класу контент-провайдера і його область відповідальності. Більш повну інформацію про можливі атрибутах можна отримати на сайті developer.android.com:
http://developer.android.com/guide/topics/manifest/provider-element.html

Порядок виконання роботи

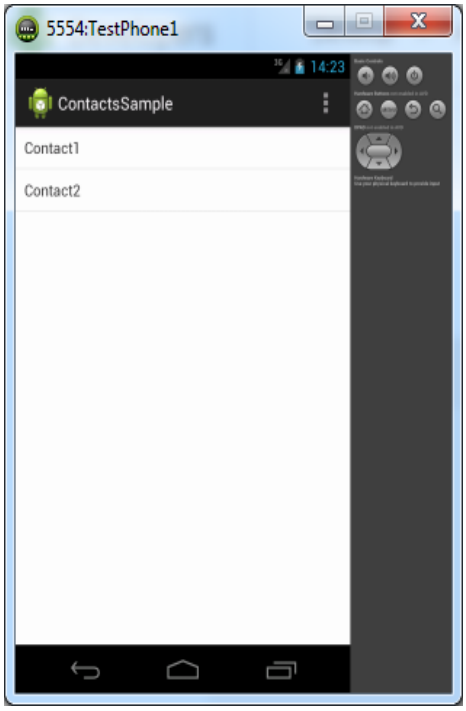
Завдання 1 «Отримання списку контактів»

Для читання інформації про контакти використовується контент-провайдер ContactsContract, точніше, один з його підкласів. Для цієї практичної роботи скористаємося провайдером ContactsContract.Contacts. Для читання контактів додатком потрібні повноваження READ_CONTACTS.

1. Додайте кілька контактів в емуляторі (оскільки потрібно тільки коротке ім'я контакту, інші поля можна не заповнювати :).

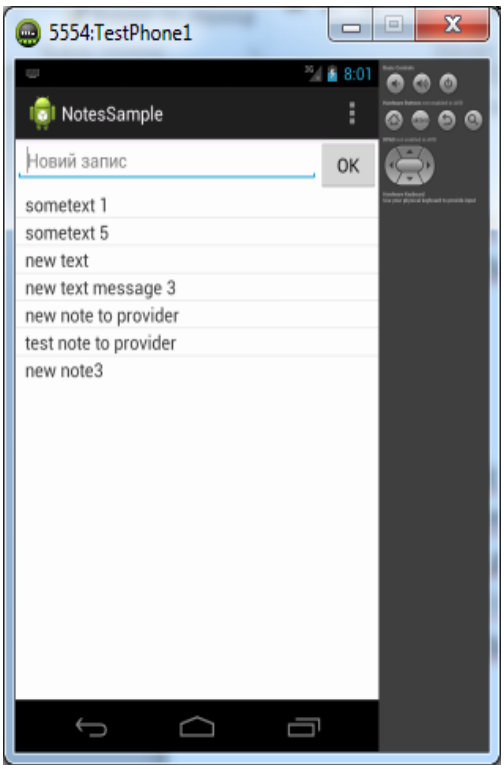
2. Створіть новий проект ContactsSample.

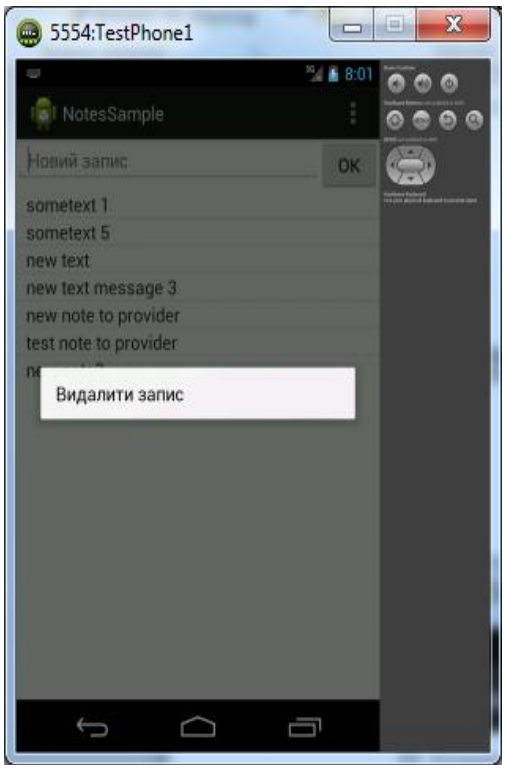
3. Виведіть імена всіх контактів (за допомогою ListView), використовуючи для отримання інформації URI ContactsContract.Contacts.CONTENT_URI. Необхідне ім'я поля для прив'язки адаптера знайдіть серед статичних констант класу ContactsContract.Contacts.



Завдання 2 «Створення контент-провайдера»

В якості основи візьміть завдання «Робота з SQLite з класом-адаптером» і замініть пряме звернення до СУБД на взаємодію з контент-провайдером, який буде інкапсуллювати всю взаємодію зі «справжніми» даними в окремому класі.





Крім простого відображення web-контенту за допомогою віджета WebView, розробник має можливість «низькорівневої роботи з різноманітними мережевими сервісами. Для цього всього лише потрібно створити мережеве підключення (запит до сервера), отримати, обробити і відобразити дані у потрібному вигляді. Традиційно найбільш зручними форматами для мережевого обміну даними вважаються XML і JSON.

Зрозуміло, будь-який додаток, що використовує мережеві підключення, повинен мати в маніфесті відповідні повноваження:

<uses-permission android:name="android.permission.INTERNET" />

Для створення потоку даних від сервера можна використовувати клас HttpUrlConnection, що є розширенням класу UrlConnection з пакету java.net. Пакети java.net і android.net містять класи, що дозволяють керувати мережевими з'єднаннями. Більш детальну інформацію про класу HttpUrlConnection з прикладами використання можна отримати тут: http://developer.android.com/reference/java/net/HttpURLConnection.html

Простий приклад створення з'єднання:

```
private static final String some_url = "....";
. . . .
try {
        // Створюємо об'єкт типу URL
        URL url = new URL(getString(R.string.rates_url));
        // З'єднуємося
        HttpURLConnection httpConnection = (HttpURLConnection) url
                        .openConnection();
        // Отримуємо код відповіді
        int responseCode = httpConnection.getResponseCode();
```

```
        // Якщо код відповіді хороший, парсер потік (відповідь сервера)
        if (responseCode == HttpURLConnection.HTTP_OK) {
            // Якщо код відповіді хороший, обробляємо відповідь
            InputStream in = httpConnection.getInputStream();
        } else {
        // Зробити оповіщення про помилки, якщо код відповіді
        нехороший
            }
    } catch (MalformedURLException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
```

Метою даного завдання є створення простого додатка, що одержує курси іноземних валют по відношенню до гривні у форматі XML і відображає дані у вигляді списку (*ListView*). Для отримання даних буде використовуватися URL http://bank-ua.com/export/currrate.xml

Відповідь сервера виглядає приблизно так:

```
<ValCurs Date="04.04.2017" name="Foreign Currency Market">
<Valute ID="R01010">
    <NumCode>036</NumCode>
    <CharCode>AUD</CharCode>
    <Nominal>1</Nominal>
    <Name>Австралийский доллар</Name>
    <Value>30,4632</Value>
</Valute>
. . . . . .
</ValCurs>
```

Для кожної валюти (елемент *Valute*) буде потрібно витягти значення дочірніх елементів *CharCode*, *Nominal*, *Name* і *Value*. Значення атрибута *Date* кореневого елемента (*ValCurs*) використовуватиметься для зміни заголовка програми.

Завдання 3 «Робота із віддаленою БД»

1. Створіть новий проект CurrencyRates. Головна (і єдина) активність з таким же ім'ям (CurrencyRates) повинна розширювати клас ListActivity.

2. Відредагуйте Маніфест додатку, додавши в нього необхідні повноваження і тему для активності (android:theme= "@android:style/Theme.Light").

3. Файл строкових ресурсів strings.xml (в каталозі res / values) відредагуйте наступним чином:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<string name="app_name"> Курс гривні НБУ </string>
<string name="rates_url"> http://bank-ua.com/export/currrate.xml
</string>
</resources>
```

4. Для відображення інформації потрібно створити розмітку для елементів списку. У каталозі res / layout створіть файл item_view.xml з наступним вмістом:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:orientation="horizontal" >
<TextView
    android:id="@+id/charCodeView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="#FF8"
    android:minWidth="45sp"
    android:padding="4dp"
    android:textColor="#00F"
    android:textStyle="bold"
    android:gravity="center"
    android:shadowDx="8"
    android:shadowDy="8"
    android:shadowColor="#000"
    android:shadowRadius="8"/>
<TextView
    android:id="@+id/valueView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textColor="#008"
    android:background="#FFE"
    android:minEms="3"
    android:padding="3dp" />
<TextView
    android:id="@+id/nominalView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:padding="3dp" />
<TextView
```

```
        android:id="@+id/nameView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ellipsize="marquee"
        android:singleLine="true" />
</LinearLayout>
```

5. Вся логіка додатка буде зосереджена в класі Currency Rates, тому інші зміни будуть стосуватися тільки цього класу. Додайте необхідні константи:

```
private final static String KEY_CHAR_CODE = "CharCode";
private final static String KEY_VALUE = "Value";
private final static String KEY_NOMINAL = "Nominal";
private final static String KEY_NAME = "Name";
```

6. Тіло методу onCreate повинно містити тільки два рядки:

```
super.onCreate(savedInstanceState);
populate();
```

Оскільки Currency Rates є спадкоємцем List Activity, викликати setContentView не потрібно. Метод populate наповнюватиме ListView вмістом за допомогою адаптера (SimpleAdapter), заповнивши його даними, отриманими від методу getData.

7. Додайте метод populate, в якому створюється і налаштовується адаптер:

```
private void populate() {
    ArrayList<Map<String, String>> data = getData();
    String[] from = { KEY_CHAR_CODE, KEY_VALUE,
    KEY_NOMINAL, KEY_NAME };
    int[] to = { R.id.charCodeView, R.id.valueView, R.id.nominalView,
            R.id.nameView };
    SimpleAdapter sa = new SimpleAdapter(this, data,
    R.layout.item_view,
            from, to);
    setListAdapter(sa);}
```

8. Додайте метод getData. Саме в ньому буде створюватися і оброблятися з'єднання з сервером, а також аналізуватися XML-дані і заповнюватися список, що буде відображатися адаптером. Метод getData об'ємніший інших, але нічого складного в ньому немає (варто відзначити, що інтерфейси Document, Element і NodeList повинні імпортуватися з пакету org.w3c.dom):

```
private ArrayList<Map<String, String>> getData() {
ArrayList<Map<String, String>> list =
```

```
        new ArrayList<Map<String, String>>();
Map<String, String> m;
try {
        // створюємо об'єкт URL
        URL url = new URL(getString(R.string.rates_url));
        // З'єднуємося
        HttpURLConnection httpConnection =
                (HttpURLConnection) url.openConnection();
        // Отримуємо від сервера код відповіді
        int responseCode = httpConnection.getResponseCode();
        // Якщо код відповіді хороший, парсер потік (відповідь сервера),
        // встановлюємо дату в заголовку програми та
        // заповнити list
        if (responseCode == HttpURLConnection.HTTP_OK) {
                InputStream in = httpConnection.getInputStream();
                DocumentBuilderFactory dbf = DocumentBuilderFactory
                                .newInstance();
                DocumentBuilder db = dbf.newDocumentBuilder();
                Document dom = db.parse(in);
                Element docElement = dom.getDocumentElement();
                String date = docElement.getAttribute("Date");
                setTitle(getTitle() + " на " + date);
                NodeList nodeList = docElement
                        .getElementsByTagName("Valute");
                int count = nodeList.getLength();
                if (nodeList != null && count > 0) {
                        for (int i = 0; i < count; i++) {
                                Element entry = (Element) nodeList.item(i);
                                m = new HashMap<String, String>();
                                String charCode = entry
                                        .getElementsByTagName(KEY_CHAR_CODE)
                                        .item(0).getFirstChild().getNodeValue();
                                String value = entry
                                        .getElementsByTagName(KEY_VALUE)
                                        .item(0).getFirstChild().getNodeValue();
                                String nominal = "за " + entry
                                        .getElementsByTagName(KEY_NOMINAL)
                                        .item(0).getFirstChild().getNodeValue();
                                String name = entry
                                        .getElementsByTagName(KEY_NAME)
                                        .item(0).getFirstChild().getNodeValue();
                                m.put(KEY_CHAR_CODE, charCode);
                                m.put(KEY_VALUE, value);
                                m.put(KEY_NOMINAL, nominal);
```

```
                    m.put(KEY_NAME, name);
                    list.add(m);}}
} else {
// Зробити оповіщення про помилки, якщо код відповіді нехороший
}
} catch (MalformedURLException e) {
        e.printStackTrace();
} catch (IOException e) {
        e.printStackTrace();
} catch (ParserConfigurationException e) {
        e.printStackTrace();
} catch (SAXException e) {
        e.printStackTrace();
}
return list;};
```

9. Перевірте працездатність програми. У реальній програмі слід відслідковувати наявність підключення пристрою до мережі, відслідковувати помилки з'єднання, а також отримувати дані з мережі в окремому потоці.

Завдання для виконання

Виконайте завдань до виконання у відповідності до теоретичних відомостей поданих вище.

Вимоги до звіту

В звіт по роботі включіть текст програм, що реалізують описані завдання та екранні форми, що відображають їх виконання.

Контрольні запитання

1. Для чого потрібен Content Provider?
2. Чи кожен Content Provider надає відкритий URI?
3. Як задекларувати Content Provider у файлі маніфесту додатку?
4. Які параметри необхідні для того, щоб зробити запити до Content Provider?
5. Як створити контент Content Provider?

СПИСОК ЛІТЕРАТУРИ

1. Голощапов А. Л. Google Android: программирование для мобильных устройств / Голощапов А. Л. – СПб. : БХВ-Петербург, 2011. – 448 с. : ил. + CD-ROM – (Профессиональное программирование).
2. С. Хашими, С. Разработка приложений для Android / С. Хашими, С. Коматинени, Д. Маклин – СПб. : Питер, 2011. – 736 с. : ил.
3. Дэрси Л. Android за 24 часа. Программирование приложений под операционную систему Google / Дэрси Л., Кондер Ш. – М. : Рид Групп, 2011. – 464 с.
4. Б. Эккель. Философия Java. Библиотека программиста. 4-е издание. - СПб. : Питер, 2011. – 400 с.
5. Васильев А. Н. Java. Обьектно-ориентированое програмирование. Учебное пособие. Стандарт третьего поколения. СПб. : Питер, 2010. – 400 с.
6. Хорстманн К.С., Корнелл Г. Библиотека профессионала: Java 2. – М.: Изд. дом «Вильямс», 2004. Т. I: Основы – 848 с. (316 р.) Т. II: Тонкости программирования – 1120 с. (365 р.)
7. Как программировать на Java. Книга 1: Основы программирования. – М.: Бином-пресс, 2003 – 848 с
8. Дейтел Х.М. и др. Технологии программирования на Java 2. Книга 1: Графика, JavaBeans, интерфейс пользователя – М.: Бином-пресс, 2003 (210 р.)
9. Бишоп Д. Java 2 (Серия «Эффективное программирование») – СПб.: Питер, 2002 – 592 с.
10. Блох Д. Java Эффективное программирование. – М. Лори, 2002 – 240 с.
11. Мак-Лахлин Б. Java и XML. СПб.: Символ-Плюс, 2002 – 544 с.
12. Стелтинг С., Маассен О. Применение шаблонов Java. Библиотека профессионала – М.: Изд. дом «Вильямс», 2002 – 576 с.

Підписано до друку - р.
Формат 84х118\32. Папір офсетний. Друк на різографі.
Умов.-друк. арк. 7,8. Зам. №2242.
Тираж 100 прим.