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

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт
з курсу: «Програмування на мові Java»

для студентів освітньо-кваліфікаційного рівня «бакалавр»
напряму підготовки 6.050201 – «Системна інженерія»

Тернопіль - 2018

Методичні вказівки до виконання лабораторних робіт з дисципліни «Програмування на мові Java» для освітнього ступеня – бакалавр, спеціальності 151 – «Автоматизація та комп'ютерно-інтегровані технології» галузі знань 15 – «Автоматизація та приладобудування» / Укл.: Заставний О.М. – Тернопіль: ТНЕУ, 2018. – 37с.

Методичні рекомендації до виконання лабораторних робіт складаються з частин, що рекомендовані освітньо-професійної програми підготовки бакалавра галузі знань 15 Автоматизація та приладобудування, спеціальність 151 – «Автоматизація та комп'ютерно-інтегровані технології»

Укладачі: Заставний Олег Михайлович к.т.н.,
Сегін Андрій Ігорович, к.т.н., доцент

Рецензенти: Сабадаш І.О. к.т.н., директор інституту мікропроцесорних систем керування об'єктами електроенергетики

Івасьєв С.В. к.т.н., старший викладач кафедри кібербезпеки Тернопільського національного економічного університету

Розглянуто та схвалено на засіданні кафедри спеціалізованих комп'ютерних систем, протокол № 2 від 11 вересня 2018р.

Розглянуто та схвалено науково-методичною комісією з автоматизації та комп'ютерно-інтегрованих технологій, протокол №2 від 11.09.2018р.

Розглянуто та схвалено науково-методичною радою факультету комп'ютерних інформаційних технологій, протокол №1 від 29.08.2018

ЗМІСТ

Лабораторна робота № 1

ТЕМА. *Основи мови Java*

МЕТА. *Навчитись складати елементарні консольні додатки на мові Java*

Завдання роботи

1. Компіляція готової програми

1.1 Запустіть програму hello.java.

2. Ініціалізація та перетворення примітивних типів

2. 1. Створіть простий консольний додаток, в якому ініціюються змінні всіх примітивних типів:

byte - 8 біт, від -128 до 127
short - 16 біт,2 байта, від -32768 до 32767
int - 32 біта, від -2147483648 до 2147483647
long - 64 біта
float - 32 біта
double - 64 біта
boolean - false|true
char - один символ

Знайдіть відповіді на питання:

2.2. Що містить змінна кожного примітивного типу яка була оголошена, але їй не було присвоєно початкового значення?

2.3. Що містить змінна типу byte, якщо присвоїти їй значення 256?

2.4. Які значення мають наведені нижче вирази? (int)’a’;
(byte)34256;
(char)»fasdf»;
(char)39;
(boolean) 1;
(boolean) ‘’
(boolean)’a’

3. Масиви

3.1 Створіть масив A з 5 елементів, в якому послідовно зберігаються числа 1,2,3,4.

Знайдіть відповідь на питання:

3. 2. Яке значення мають елементи A[0], A[1], A[4], A[5]?

4. Оператори

4.1. Продемонструйте виконання операторів

A+=B
A-=B
A*=B
A/=B
A%B
A++
A--
~
&
|
^
>>
>>>
<<

Приклад:

```
class OpDemo {
  public static void main(String args[]) {
    double a = 1;
    System.out.println("Було a = " + a);
    a+= 5;
    System.out.println("Стало a = " + a);
   }
 }
```

5. Цикли та умовні оператори

5.1. Створіть програму яка надрукує всі прості числа, менші від 100.

6. Прості класи

6.1. Створіть клас point, який представляє точку з трьома координатами (x, y, z) і не має конструктора.

Знайдіть відповідь на питання:

6.2.Чи створить хоч яка-небудь з наведених нижче двох команд об'єкт класу point?

```
point p=new point(1.2, 3.5, 4.5);
point p=new point();
```

6.3. Додайте в клас point конструктор з трьома дійсними аргументами.

Знайдіть відповідь на питання:

6.4.Яка з наведених нижче двох команд створить об'єкт класу point?

```
point p=new point(1.2, 3.5, 4.5);
point p=new point();
```

7. Просте успадкування

7.1. Розгляньте код

```
class A{
  char a='a';
  void callme() {
     System.out.println("Я - callme з класу A ");
  }
}
class B extends A {
   char b='b';
   void callme() {
       System.out.println("Я - callme з класу B ");
   }
}

class Dispatch{
  public static void main(String args[]){
     A a = new A();
     A ab = new B();
     B b = new B();
     /**/
  }
}
```

7. 2. Визначте, що надрукують команди

```
a.callme();
ab.callme();
b.callme();
System.out.println(a.a);
System.out.println(ab.a);
System.out.println(b.a);
System.out.println(a.b);
System.out.println(ab.b);
System.out.println(b.b);
```

якщо помістити їх замість коментаря /**/ ?

Хід роботи

1. Робота в середовищі розробки NetBeans

1.1. Встановіть компілятор Java та середовище розробки NetBeans.

2. Створення в NetBeans найпростішого додатка Java:

1. Запустити інтегроване середовище розробки (IDE) NetBeans,

2. Вибрати в головному меню File/New Project…
3. В діалозі вибрати General / Java Application
4. Натиснути кнопку Next
5. На наступному кроці варто прибрати відмітку біля поля Create Main Class та надрукувати назву проекта
6. Натиснути кнопку Finish
7. На цьому етапі ви маєте новий проект і можете створювати новий клас:
8. Вибрати в головному меню File/New File…
9. В діалозі вибрати Java / Java Main Class
10. Натиснути кнопку Next
11. В полі Class Name надрукуйте назву класа (ця ж сама назва буде і назвою класу)
12. Натиснути кнопку Finish
13. Щоб скомпілювати та виконати програму, натисніть правою кнопкою миші на файл з текстом програми та виберіть команду "Run File"

3. Встановлення компілятора

3.1. Завантажте файл з компілятором java (jdk1.1.8.rar) та розгорніть його в директорію C:\jdk1.1.8

3. 2. Відкрийте з допомогою файлового менеджера FAR директорію C:\jdk1.1.8\compiler_example та знайдіть файли compile.bat, run.bat та hello.java

- Файл compile.bat містить команди, призначені для компіляції:

```
set JDKHOME=C:\jdk1.1.8
set CLASSPATH=.;%JDKHOME%\lib\classes.zip
%JDKHOME%\bin\javac %1 %2 %3 %4 %5
```

- Файл run.bat містить команди, призначені для виконання програми:

```
set JDKHOME=C:\jdk1.1.8
set CLASSPATH=.;%JDKHOME%\lib\classes.zip
%JREHOME%\bin\java -classpath %CLASSPATH% %1 %2 %3 %4 %5 %6
```

3.3. Для перевірки правильності інсталяції наберіть в командному рядку

```
compile hello.java
run hello
```

Якщо компілятор встановлено правильно, програма надрукує слово Hello.

4. Стандартний вивід в консольних додатках:

4.1. Для виводу рядків на консоль використовуйте метод System.out.println

Зразок:

```
System.out.println("myIntArray["+i+"]"+" = "+myIntArray[i]);
```

Увага! В деяких виразах навмисне допущені помилки. Одним з ваших завдань є їх виявлення.

Лабораторна робота № 2

ТЕМА. *Створення простих класів в Java*

МЕТА. *Навчитись створювати прості класи та об'єкти на мові Java*

Завдання роботи

1. Створіть на Java прості класи та продемонструйте їх функції:

Варіант 1

Клас «комплексне число», який дозволяє знаходити модуль комплексного числа, його аргумент, виконує основні арифметичні операції з числами: додавання, віднімання, множення, ділення.

Варіант 2

Клас «вектор», який описує вектор у 3-вимірному декартовому просторі. Методи класу дозволяють знаходити модуль вектора, множити вектор на число, додавати вектори, множити скалярно і векторно на інший вектор.

Варіант 3

Клас «матриця 2х2», який описує матрицю з 2 стовпчиками і 2 рядками. Методи класу дозволяють знаходити детермінант, обернену матрицю, множити на іншу матрицю, додавати до іншої матриці, множити матрицю на число.

Варіант 4.

Клас «трикутник», який описує трикутник з заданими сторонами. Методи класу дозволяють знаходити периметр трикутника, площу трикутника, кути при кожній вершині.

Варіант 5

Клас «квадрат», який описує квадрат з заданими сторонами. Методи класу дозволяють знаходити периметр та площу квадрата, довжину діагоналей.

Варіант 6

Клас «коло», який описує коло з заданим радіусом. Методи класу дозволяють знаходити довжину кола, площу круга. Сторону вписаного правильного трикутника.

Варіант 7

Клас «куб», який описує куб з заданим ребром. Методи класу дозволяють знаходити сумарну площу граней, об'єм куба, довжину великої діагоналі.

Варіант 8

Клас «конус», який описує правильний конус з заданою висотою та радіусом основи. Методи класу дозволяють знаходити площу бічної основи та об'єм конуса.

Варіант 9

Клас «призма», який описує прямокутну призму заданої висоти, в основі якої лежить правильний трикутник з заданою стороною. Методи класу дозволяють знаходити площу бічної основи та об'єм призми.

Вказівка: математичні функції містяться в класі java.lang.Math

Лабораторна робота № 3

ТЕМА. *Ініціалізація та очистка об'єктів в Java*

МЕТА. *Ознайомитися з послідовністю ініціалізації в Java*

Завдання роботи

Дослідіть наведені нижче приклади і знайдіть відповідь на запопоновані нижче питання.

1. Використання конструктора

Приклад 1.

```
class Rock {
  int f(){System.out.println("Rock.f()");}
}
 public class SimpleConstructor {
  public static void main(String[] args) {
        Rock r1 = new Rock();
        r1.f();
  }
}
```

Запитання

1.2. Чи можлива ініціалізація об'єкта без конструктора?

1.3. Чи успішно створиться об'єкт в Прикладі 1? Чому?

Приклад 2

```
class Rock {
  void f(){System.out.println("Rock.f()");}
  Rock(int i){
    System.out.println(
      "Creating Rock number " + i);
  }
}
 public class SimpleConstructor {
  public static void main(String[] args) {
        Rock r1 = new Rock();
        r1.f();
  }
}
```

Запитання

1.4. Чи успішно створиться об'єкт в Прикладі 2? Чому?

Приклад 3

2. Перевантаження конструкторів

```
class Rock {
  void f(){System.out.println("Rock.f()");}
  Rock(){
        System.out.println("Creating Rock ");
  }
   Rock(int i){
    System.out.println(
      "Creating Rock number " + i);
  }
}
 public class SimpleConstructor {
  public static void main(String[] args) {
        Rock r1 = new Rock();
        Rock r2 = new Rock(2);
        r1.f();
        r2.f();
  }
}
```

Питання

2.1. Чи може клас мати кілька конструкторів?

3. Автоматична ініціалізація примітивних типів

```
class Measurement {
  boolean t;
  char c;
  byte b;
  short s;
  int i;
  long l;
  float f;
  double d;
  void print() {
    System.out.println(
      "Data type      Initial value\n" +
      "boolean        " + t + "\n" +
      "char           [" + c + "] "+ (int)c +"\n"+
      "byte           " + b + "\n" +
      "short          " + s + "\n" +
      "int            " + i + "\n" +
      "long           " + l + "\n" +
      "float          " + f + "\n" +
      "double         " + d);
  }
}
```

```
 public class InitialValues {
  public static void main(String[] args) {
    Measurement d = new Measurement();
    d.print();
  }
}
```

Питання

3.1. Що надрукує наведений вище код?

4. Ініціалізація в місці оголошення атрибута

```
class Measurement {
  boolean t;
  char c;
  byte b=new_b();
  short s=13;
  int i=new_i();
  long l;
  float f;
  double d;
  void print() {
    System.out.println(
      "Data type      Initial value\n" +
      "boolean        " + t + "\n" +
      "char           [" + c + "] "+ (int)c +"\n"+
      "byte           " + b + "\n" +
      "short          " + s + "\n" +
      "int            " + i + "\n" +
      "long           " + l + "\n" +
      "float          " + f + "\n" +
      "double         " + d);
  }
  int new_i(){
    System.out.println("Creating i");
    return this.s+7;
  }
  int new_b(){
    System.out.println("Creating b");
    return this.s+7;
  }
  Measurement(){
     System.out.println("Constructor Measurement()");
  }
}
 public class InitialValues {
  public static void main(String[] args) {
    Measurement d = new Measurement();
    d.print();
  }
```

```
}
```

Питання

4.1. Що надрукує наведений вище код? Чому?

5. Демонстрація послідовності ініціалізації

```
// Коли конструктор викликається для створення
// об'єкта Tag, ви побачите повідомлення:
class Tag {
  int t;
  Tag(int marker) {
    System.out.println("Creating Tag(" + marker + ")");
    this.t= marker;
  }
  p(){
    System.out.println("Tag(" + t + ")");
  }
}

class Card {
  Tag t1 = new Tag(1); // Перед конструктором
  Card() {
    // Указывает, что мы в конструкторе:
    System.out.println("Card()");
    t3 = new Tag(33); // Повторная инициализация t3
  }
  Tag t2 = new Tag(2); // После конструктора
  void f() {
    System.out.println("f()");
  }
  Tag t3 = new Tag(3); // В конце
}

public class OrderOfInitialization {
  public static void main(String[] args) {
    Card t = new Card();
    t.f(); // Показывает завершение конструктора
    t.t1.p();
    t.t2.p();
    t.t3.p();
  }
} ///:~
```

Питання

5.1. Що надрукує наведений вище код? Чому?

6. Ініціалізація статичних атрибутів

```
class Bowl {
  Bowl(int marker) { System.out.println("Bowl(" + marker + ")");  }
```

```
  void f(int marker) { System.out.println("f(" + marker + ")");  }
}

class Table {
  static Bowl b1 = new Bowl(1);
  Table() {
    System.out.println("Table()");
    b2.f(1);
  }
  void f2(int marker) {
    System.out.println("f2(" + marker + ")");
  }
  static Bowl b2 = new Bowl(2);
}

class Cupboard {
  Bowl b3 = new Bowl(3);
  static Bowl b4 = new Bowl(4);
  Cupboard() {
    System.out.println("Cupboard()");
    b4.f(2);
  }
  void f3(int marker) {
    System.out.println("f3(" + marker + ")");
  }
  static Bowl b5 = new Bowl(5);
}

public class StaticInitialization {
  public static void main(String[] args) {
    System.out.println(
      "Creating new Cupboard() in main");
    new Cupboard();
    System.out.println(
      "Creating new Cupboard() in main");
    new Cupboard();
    t2.f2(1);
    t3.f3(1);
  }
  static Table t2 = new Table();
  static Cupboard t3 = new Cupboard();
}
```

Питання

6.1. Чим статичні атрибути відрізняються від інших?
6.2. Що надрукує наведений вище код?
6.3. Скільки разів виконається конструктор Bowl(5)?
6.4.Скільки разів виконається конструктор Bowl(3)?

7. Явна ініціалізація блоком static

```
// Явная static инициализация
// с предложением "static".

class Cup {
  Cup(int marker) {
    System.out.println("Cup(" + marker + ")");
  }
  void f(int marker) {
    System.out.println("f(" + marker + ")");
  }
}

class Cups {
  static Cup c1;
  Cup c2;
  static {
    c1 = new Cup(1);
  }
  {
    c2 = new Cup(2);
  }
  Cups() {
    System.out.println("Cups()");
  }
}

public class ExplicitStatic {
  public static void;; main(String[] args) {
    System.out.println("Inside main()");
    Cups.c1.f(99);  // (1)
  }
  static Cups x = new Cups();  // (2)
  static Cups y = new Cups();  // (2)
}
```

Питання

7.1. Що надрукує наведений вище код?
7.2. Як можна викликати метод, не створюючи об'єкта?
7.3. Скільки разів виконається конструктор Cup(2)?
7.4. Скільки разів виконається конструктор Cup(1)?

8. Ініціалізація та успадкування

```
class Art {
  Art() {
    System.out.println("Art constructor");
  }
}
```

```
class Drawing extends Art {
 Drawing() {
   System.out.println("Drawing constructor");
 }
}
public class Cartoon extends Drawing {
 Cartoon() {
   System.out.println("Cartoon constructor");
 }
 public static void main(String[] args) {
   Cartoon x = new Cartoon();
 }
}
```

Питання

8.1. В якій послідовності викликаються конструктори?

9. Прибирання сміття

```
//: c04:Garbage.java
// Демонстрація прибирання сміття
// і фіналізації

class Chair {
 static boolean gcrun = false;
 static boolean f = false;
 static int created = 0;
 static int finalized = 0;
 int i;
 Chair() {
   i = ++created;
   if(created == 47)
     System.out.println("Created 47");
 }
 public void finalize() {
   if(!gcrun) {
     // Перший раз викликається finalize():
     gcrun = true;
     System.out.println(
       "Beginning to finalize after " +
       created + " Chairs have been created");
   }
   if(i == 47) {
     System.out.println(
       "Finalizing Chair #47, " +
       "Setting flag to stop Chair creation");
     f = true;
   }
   finalized++;
   if(finalized >= created)
```

```
    System.out.println(
      "All " + finalized + " finalized");
  }
}

public class Garbage {
  public static void main(String[] args) {
    // До тогочасу, поки прапорець не встановлений,
    // створюються Chairs и Strings:
    while(!Chair.f) {
      new Chair();
      new String("To take up space");
    }
    System.out.println(
      "After all Chairs have been created:\n" +
      "total created = " + Chair.created +
      ", total finalized = " + Chair.finalized);
    // Необов'язкові аргументи форсують
    // прибирання сміття і фіналізацію
    if(args.length > 0) {
      if(args[0].equals("gc") ||
         args[0].equals("all")) {
        System.out.println("gc():");
        System.gc();
      }
      if(args[0].equals("finalize") ||
         args[0].equals("all")) {
        System.out.println("runFinalization():");
        System.runFinalization();
      }
    }
    System.out.println("bye!");
  }
} ///:~
```

Питання

9.1. Чи завжди виконується прибирання сміття?

9.2.Як примусити віртуальну машину Java почати прибирання сміття?

9. 3. В який момент викликається метод finalize()?

Лабораторна робота № 4

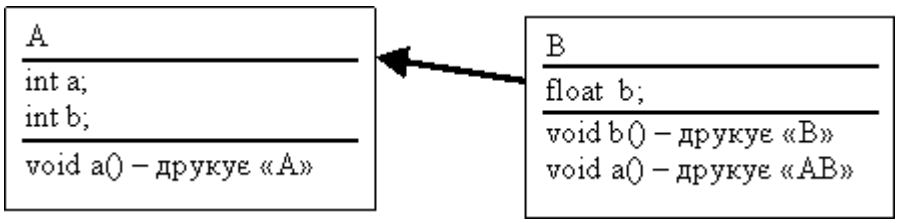
ТЕМА. *Дослідження успадкування в Java*

МЕТА. *Ознайомитись з будовою Java – аплетів.*

Завдання роботи

1. Просте успадкування

1.1 Створіть клас А та його спадкоємця – клас В.



1.2. Створіть об'єкти класів А та В. Продемонструйте виклики всіх методів класу А та всіх методів класу В.

Знайдіть відповідь на запитання:

1.3. Як створити клас – спадкоємець?

1.4.Що таке перекриття методів та змінних?

1.5. Чи можна перекривати поле класу однойменним полем іншого типу? (Наприклад, якщо поле b в класі А – ціле число, а поле b в класі В – десятковий дріб).

1.6. Чи можна перетворити об'єкти типу В в об'єкти типу А? А навпаки?

1.7. Що надрукує наведена нижче програма?

```
public class in1 {
  public static void main(String[] args) {
    a oa=new a();
    ab ob=new ab();
    oa.prn();
    ob.prn();
    System.out.println(ob.a);
    ((a)ob).prn();
  }
}

class a{
  int a=0;
  void prn(){System.out.println(a);}
}

class ab extends a{
  String a="aabb";
```

```
}
```

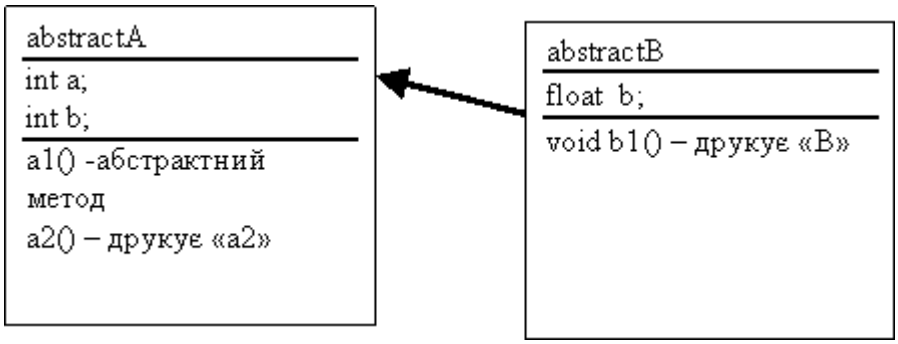
2. Заборона перекриття
2.1. Змініть розв'язок завдання 1 так, щоб заборонити перекриття методу a(). Продемонструйте дію заборони.

Знайдіть відповідь на запитання:
2.2. Як заборонити перекриття метода?

3. Заборона успадкування
3.1. Змініть розв'язок завдання 1 так, щоб заборонити успадкування методу a(). Продемонструйте дію заборони. <

Знайдіть відповідь на запитання:
3.2.Як заборонити успадкування метода?

4. Абстрактні класи
4.1. Створіть абстрактні класи abstractA та abstractB

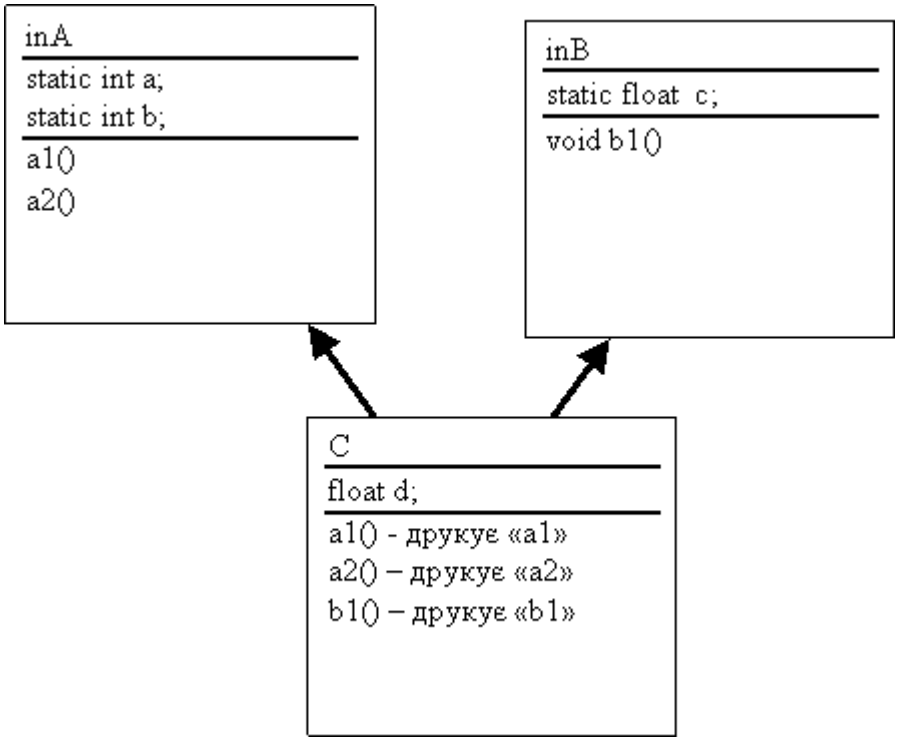


4.2.Створіть об'єкти класів abstractA та abstractB. Продемонструйте виклики методів.

Вказівка:
Для створення об'єкта абстрактного класу потрібно створити клас-спадкоємець, який реалізує всі абстрактні методи, а потім скористатись перетворенням типів.

Знайдіть відповідь на запитання:
4.3.Чи можна створити об'єкт абстрактного класу з допомогою оператора new?
4.4. Чи існують інші можливості створення об'єкт абстрактного класу? Які?

5. Інтерфейси
5.1.Створіть інтерфейси inA, inB та їх спадкоємця – клас C. Продемонструйте роботу методів.



Знайдіть відповідь на запитання:

5.2.Чи може інтерфейс мати не статичні поля?

5.3.Скільки предків-класів може мати клас в Java?

5.4.Скільки предків-інтерфейсів може мати клас в Java?

6. Вплив модифікаторів доступу на успадкування

6.1 Створіть в одній директорії клас A та його спадкоємця – клас B:

```
class A{
/*модифікатор1*/ void p(){System.out.println("A.p()");}
}

class AB extends A{
/*модифікатор2*/ void p(){System.out.println("AB.p()");}
}
```

По черзі замініть /*модифікатор1*/ та /*модифікатор2*/ модифікаторами доступу private, protected, public та випишіть у вигляді таблиці допустимі комбінації:

Класи в одному пакеті (директорії):	
/*модифікатор1*/	/*модифікатор2*/
Відсутній	Відсутній
Public	public
...	...
...	...
...	...
...	...
...	...

Вказівка: Обов’язково дослідіть випадок, коли модифікатор відсутній.

6.2 Створіть в різних директоріях клас A та його спадкоємця – клас B:

```
Package test1;
class A{
  /*модифікатор1*/  void p(){System.out.println("A.p()");}
}


Package test2;
class AB extends A{
 /*модифікатор2*/ void p(){System.out.println("AB.p()");}
}
```

По черзі замініть /*модифікатор1*/ та /*модифікатор2*/ модифікаторами доступу private, protected, public та випишіть у вигляді таблиці допустимі комбінації:

Класи в різних пакетах(директоріях):	
/*модифікатор1*/	/*модифікатор2*/
Відсутній	Відсутній
public	public
...	...
...	...
...	...
...	...
...	...
...	...

Вказівка: Обов’язково дослідіть випадок, коли модифікатор відсутній.

Лабораторна робота № 5

ТЕМА. *Графічний інтерфейс програм на Java*

МЕТА. *Вивчення графічного інтерфейсу користувача в програмах на мові Java*

Завдання роботи

В файлі **Calculator.java** міститься код програми "Калькулятор". Код містить помилку, в результаті якої кнопки математичних операцій ("+", "-", "*", і т.д.) функціонують неправильно. Виправте помилки так, щоб калькулятор працював коректно.

```
import java.io.*;
import java.lang.*;
import java.awt.*;
import java.awt.event.*;
/*
 * Простий калькулятор
 */
class Calculator extends Frame implements ActionListener{
  // арифметичні змінні
  private char savedAction;
  private boolean clear;
  private float savedValue;

  // змінні графічного інтерфейсу
  private TextField display;
  private Panel buttonsPanel;
  private Button buttons[];

  public Calculator()
  {
    setupGUI();
    this.addWindowListener(new WindowAdapter(){
      public void windowClosing(WindowEvent ev){
        System.exit(0);
      }
    }
    );

   // при натисканні кнопки з ариф. діями в текстовім полі стирається інформація
```

```
    clear = true;
  }

public Insets insets() {
return new Insets(30, 15, 15, 15) ;

}

  private void setupGUI()
  {
     this.setLayout( new BorderLayout() );

  // створення однострокової області для вводу тексту
display = new TextField("0"); //створили новый об'єект
this.add(display, BorderLayout.NORTH); //добавили його у вікно верхнього рівня


// створюєм панель
buttonsPanel = new Panel(); //створили новий об'єкт Panel()
buttonsPanel.setLayout( new GridLayout(4,5) );//вказали що панель має
                                    // 4 рядки  и 5 стовбці
this.add( buttonsPanel, BorderLayout.CENTER );//добавили панель в центр вікна

    //створюєм 20 кнопок
    buttons = new Button[20];
    for(int i=0; i<buttons.length; i++){
      buttons[i] = new Button();  // створили новий об'єкт Button()
      buttons[i].addActionListener( this );
      buttonsPanel.add( buttons[i] ); //  добавили об'єкт до панелі
    }

    // підписуєм кнопки
      buttons[0].setLabel("7");
      buttons[1].setLabel("8");
      buttons[2].setLabel("9");
      buttons[5].setLabel("4");
      buttons[6].setLabel("5");
      buttons[7].setLabel("6");
      buttons[10].setLabel("1");
      buttons[11].setLabel("2");
      buttons[12].setLabel("3");
      buttons[15].setLabel("0");

    // підписуєм кнопки з ар.діями
      buttons[3].setLabel("/");
      buttons[8].setLabel("*");
      buttons[13].setLabel("-");
      buttons[18].setLabel("+");
```

```
    buttons[17].setLabel(".");
    buttons[19].setLabel("=");
    buttons[4].setLabel("clear");
    buttons[9].setLabel("%");
    buttons[14].setLabel("sqrt");
    buttons[16].setLabel("+/-");

  //обчислюєм разміри кнопок, текстовых полей
    pack();
}

public void actionPerformed( ActionEvent e )
{
  String str;
  char cmd;

  str = e.getActionCommand();
  cmd = (str.toCharArray())[0];// рядок перетворюється в масив із нього береться
                               // перша буква

  // display or calulate
  if( Character.isDigit( cmd ) || cmd == '.' )
  {
    if( clear )
    {
     try{
        savedValue = (new Float(display.getText())).floatValue();
      }
      catch(NumberFormatException e1){
        display.setText("");
      }
      clear = false;
    }
    // add new digit to display
    display.setText( display.getText() + cmd );
  }
  else
  {

    calculate(cmd);
  }
}

private void calculate(char cmd)
{
  float curValue;
  float newValue = 0;
  // get current value of display
```

```
try{
  curValue = (new Float(display.getText())).floatValue();
}
catch(NumberFormatException ep)
{
     curValue = 0;
}

System.out.println("savedAction="+savedAction);
if( savedAction == 0 )
{
  clear = true;
  savedAction = cmd;
  switch(savedAction)
  {
   case 's':
            newValue = (float)Math.sqrt( (double) curValue);
            savedAction = 0;
            display.setText(""+newValue);
            break;
   case 'c':
            newValue = 0;
            savedAction = 0;
            display.setText(""+newValue);
            break;
   case '[':
            newValue = - curValue;
            savedAction = 0;
            display.setText(""+newValue);
            break;
  }
}
else
{
  switch(savedAction)
  {
    case '/':
            newValue = savedValue / curValue;
            break;
    case '*':
            newValue = savedValue * curValue;
            break;
    case '-':
            newValue = savedValue - curValue;
            break;
    case '+':
            newValue = savedValue + curValue;
            break;
```

```
      case '%':
              newValue = savedValue % curValue;
              break;
    }

    display.setText(""+newValue);
    clear = true;

    if( cmd == '=' )
    {

      savedAction = 0;
    }
    else
    {

      savedAction = cmd;
    }
   }
  }
 //запустіть із командного рядка
 public static void main(String args[])
 {
   new Calculator().show();
 }

}
```

Вказівки

В поточній версії програми обробник подій розрізняє команди за написами на кнопках. Для правильної роботи потрібно присвоїти кожній кнопці команду, яка не буде залежати від напису. Розгляньте довідку для класу java.awt.Button та зверніть увагу на метод void setActionCommand(String command)

Лабораторна робота № 6

ТЕМА. *Створення java-аплетів*

МЕТА. *Ознайомитися з будовою Java-аплетів*

Завдання роботи

1. Дослідіть аплет "Годинник" з набору безкоштовних прикладів Sun Java SDK (Вихідний код приклада):

```
/*
 * Copyright (c) 2003 Sun Microsystems, Inc. All  Rights Reserved.
* @(#)Clock.java1.12 03/01/23
 */

import java.util.*;
import java.awt.*;
import java.applet.*;
import java.text.*;

/**
 * Time!
 *
 * @author Rachel Gollub
 * @modified Daniel Peek replaced circle drawing calculation, few more
changes
 */
public class Clock extends Applet implements Runnable {
    private volatile Thread timer;       // The thread that displays
clock
    private int lastxs, lastys, lastxm,
                lastym, lastxh, lastyh;  // Dimensions used to draw hands
    private SimpleDateFormat formatter;  // Formats the date displayed
    private String lastdate;             // String to hold date displayed
    private Font clockFaceFont;          // Font for number display on
clock
    private Date currentDate;            // Used to get date to display
    private Color handColor;             // Color of main hands and dial
    private Color numberColor;           // Color of second hand and
numbers
    private int xcenter = 80, ycenter = 55; // Center position

    public void init() {
        int x,y;
        lastxs = lastys = lastxm = lastym = lastxh = lastyh = 0;
        formatter = new SimpleDateFormat ("EEE MMM dd hh:mm:ss yyyy",
```

```
                                            Locale.getDefault());
        currentDate = new Date();
        lastdate = formatter.format(currentDate);
        clockFaceFont = new Font("Serif", Font.PLAIN, 14);
        handColor = Color.blue;
        numberColor = Color.darkGray;

        try {
            setBackground(new
Color(Integer.parseInt(getParameter("bgcolor"),
                                                    16)));
        } catch (NullPointerException e) {
        } catch (NumberFormatException e) {
        }
        try {
            handColor = new
Color(Integer.parseInt(getParameter("fgcolor1"),
                                                  16));
        } catch (NullPointerException e) {
        } catch (NumberFormatException e) {
        }
        try {
            numberColor = new
Color(Integer.parseInt(getParameter("fgcolor2"),
                                                    16));
        } catch (NullPointerException e) {
        } catch (NumberFormatException e) {
        }
        resize(300,300);              // Set clock window size
    }

    // Paint is the main part of the program
    public void update(Graphics g) {
        int xh, yh, xm, ym, xs, ys;
        int s = 0, m = 10, h = 10;
        String today;

        currentDate = new Date();

        formatter.applyPattern("s");
        try {
            s = Integer.parseInt(formatter.format(currentDate));
        } catch (NumberFormatException n) {
            s = 0;
        }
        formatter.applyPattern("m");
        try {
            m = Integer.parseInt(formatter.format(currentDate));
        } catch (NumberFormatException n) {
            m = 10;
        }
        formatter.applyPattern("h");
        try {
```

```
            h = Integer.parseInt(formatter.format(currentDate));
        } catch (NumberFormatException n) {
            h = 10;
        }

        // Set position of the ends of the hands
        xs = (int) (Math.cos(s * Math.PI / 30 - Math.PI / 2) * 45 +
xcenter);
        ys = (int) (Math.sin(s * Math.PI / 30 - Math.PI / 2) * 45 +
ycenter);
        xm = (int) (Math.cos(m * Math.PI / 30 - Math.PI / 2) * 40 +
xcenter);
        ym = (int) (Math.sin(m * Math.PI / 30 - Math.PI / 2) * 40 +
ycenter);
        xh = (int) (Math.cos((h*30 + m / 2) * Math.PI / 180 - Math.PI /
2) * 30
                    + xcenter);
        yh = (int) (Math.sin((h*30 + m / 2) * Math.PI / 180 - Math.PI /
2) * 30
                    + ycenter);

        // Get the date to print at the bottom
        formatter.applyPattern("EEE MMM dd HH:mm:ss yyyy");
        today = formatter.format(currentDate);

        g.setFont(clockFaceFont);
        // Erase if necessary
        g.setColor(getBackground());
        if (xs != lastxs || ys != lastys) {
            g.drawLine(xcenter, ycenter, lastxs, lastys);
            g.drawString(lastdate, 5, 125);
        }
        if (xm != lastxm || ym != lastym) {
            g.drawLine(xcenter, ycenter-1, lastxm, lastym);
            g.drawLine(xcenter-1, ycenter, lastxm, lastym);
        }
        if (xh != lastxh || yh != lastyh) {
            g.drawLine(xcenter, ycenter-1, lastxh, lastyh);
            g.drawLine(xcenter-1, ycenter, lastxh, lastyh);
        }
        // Draw date and hands
        g.setColor(numberColor);
        g.drawString(today, 5, 125);
        g.drawLine(xcenter, ycenter, xs, ys);
        g.setColor(handColor);
        g.drawLine(xcenter, ycenter-1, xm, ym);
        g.drawLine(xcenter-1, ycenter, xm, ym);
        g.drawLine(xcenter, ycenter-1, xh, yh);
        g.drawLine(xcenter-1, ycenter, xh, yh);
        lastxs = xs; lastys = ys;
        lastxm = xm; lastym = ym;
        lastxh = xh; lastyh = yh;
        lastdate = today;
```

```
    currentDate = null;
}
public void paint(Graphics g) {
    g.setFont(clockFaceFont);
    // Draw the circle and numbers
    g.setColor(handColor);
    g.drawArc(xcenter-50, ycenter-50, 100, 100, 0, 360);
    g.setColor(numberColor);
    g.drawString("9", xcenter-45, ycenter+3);
    g.drawString("3", xcenter+40, ycenter+3);
    g.drawString("12", xcenter-5, ycenter-37);
    g.drawString("6", xcenter-3, ycenter+45);

    // Draw date and hands
    g.setColor(numberColor);
    g.drawString(lastdate, 5, 125);
    g.drawLine(xcenter, ycenter, lastxs, lastys);
    g.setColor(handColor);
    g.drawLine(xcenter, ycenter-1, lastxm, lastym);
    g.drawLine(xcenter-1, ycenter, lastxm, lastym);
    g.drawLine(xcenter, ycenter-1, lastxh, lastyh);
    g.drawLine(xcenter-1, ycenter, lastxh, lastyh);
}

public void start() {
    timer = new Thread(this);
    timer.start();
}

public void stop() {
    timer = null;
}

public void run() {
    Thread me = Thread.currentThread();
    while (timer == me) {
        try {
            Thread.currentThread().sleep(100);
        } catch (InterruptedException e) {
        }
        repaint();
    }
}

public String getAppletInfo() {
    return "Title: A Clock \n"
        + "Author: Rachel Gollub, 1995 \n"
        + "An analog clock.";
}
public String[][] getParameterInfo() {
    String[][] info = {
        {"bgcolor", "hexadecimal RGB number",
```

```
             "The background color. Default is the color of your
browser."},
            {"fgcolor1", "hexadecimal RGB number",
             "The color of the hands and dial. Default is blue."},
            {"fgcolor2", "hexadecimal RGB number",
             "The color of the second hand and numbers. Default is dark
gray."}
        };
        return info;
    }
}
```

2. Змінити зовнішній вигляд годинника в аплеті. Новий вигляд схематично показано на малюнку:

Лабораторна робота № 7

ТЕМА. *Клієнт та сервер на Java*

МЕТА. *Вивчення засобів комунікації засобами TCP/IP в Java*

Завдання роботи

Варіант 1

Нижче наведено вихідні коди для HTTP-клієнта та HTTP- сервера. Скомпілюйте та запустіть їх, виправивши помилки.

Код HTTP - клієнта

```
import java.net.*;
import java.io.*;
import java.util.*;
class Client{
  public static void main(String[] args){
    if (args.length != 3){
      System.err.println("Usage: Client host port file");
      System.exit(0) ;
    }
    String host = args[0];
    int port = Integer.parselnt(args[1]);
    String file = args[2];
    try{
      Socket sock = new Socket(host, port);
      PrintWriter pw = new PrintWriter(new OutputStreamWriter( sock.getOutputStreamf)), true);
      pw.println("POST " + file + " HTTP/1.l\n");
      BufferedReader br = new BufferedReader(new InputStreamReader(sock.getlnputStream() ) ) ;
      String line = null;
      line = br.readLine();
      StringTokenizer st = new StringTokenizer(line);
      String code = null;
      if ((st.countTokens() >= 2) && st.nextToken().equals("POST")){
        if ((code = st.nextToken()) != "200") {
          System.err.println("File not found, code = " + code);
          System.exit (0);
        }
      }
      while ((line = br.readLine()) != null) System.out.println{line);
      sock.close();
```

```
    }catch(Exception e){
      System.err.println(e);
    }
  }
}
```

Код HTTP - сервера

```
import j ava.net.*;
import java.io.*;
import j ava.uti1.*;
class Server!
public static void main(String[] args){
try{
ServerSocket ss = new ServerSocket(Integer.parselnt(args[0]));
while (true)
new HttpConnect(ss.accept());
}catch(ArraylndexOutOfBoundsException ae){
System.err.println("Usage: Server port");
System.exit(0);
}catch(IOException e){
System.out.println(e);
}
}
}
class HttpConnect extends Thread{
  private Socket sock;
  HttpConnect(Socket s) {
    sock = s;
    setPriority(NORM_PRIORITY - 1);
    start {) ;
  }
  public void run(){
    try{
      PrintWriter pw = new PrintWriter(new OutputStreamWriter(sock.getOutputStream()}, true);
      BufferedReader br = new BufferedReader(new InputStreamReader(sock.getlnputStream() ) ) ;
      String req = br.readLine();
      System.out.println("Request: " + req);
      StringTokenizer st = new StringTokenizer(req);
      if ((st.countTokens() >= 2) && st.nextToken().equals("POST")){
        if ((req = st.nextToken()).endsWith("/") II req.equals("")) req += "index.html";
        try{
          File f = new File(req);
          BufferedReader bfr = new BufferedReader(new FileReader(f));
          char[] data = new char[(int)f.length()];
          bfr.read(data);
          pw.println("HTTP/1.1 200 OKn");
          pw.write(data);
          pw.flush();
```

```
        }catch(FileNotFoundException fe){
        pw.println("HTTP/1.1 404 Not FoundX\n");
    }catch(lOException ioe){
    System.out.println(ioe);
    }
}else pw.println("HTTP/l.l 400 Bad RequestW);
sock.close();
}catch(IOException e){
System.out.println(e);
}
}
}
```

Варіант 2

Наведений нижче клієнт автоматично відправляє серверу, який запущено на локальному комп'ютері, кілька рядків. Змініть код клієнта так, щоб він спочатку читав зі стандартного вводу IP-адресу сервера, а потім рядки, які треба відправити.

1. **Сервер**

```
//: c15:JabberServer.java
// Дуже простий сервер, який тільки є
// відображає то, що посилає клієнт.
import java.io.*;
import java.net.*;


public class JabberServer {
  // Вибираєм номер порта за межами 1-1024:
  public static final int PORT = 8080;
  public static void main(String[] args)
      throws IOException {
    ServerSocket s = new ServerSocket(PORT);
    System.out.println("Started: " + s);
    try {
      // Блокуємо поки не проойде з'єднання:
      Socket socket = s.accept();
      try {
        System.out.println(
          "Connection accepted: "+ socket);
        BufferedReader in =
          new BufferedReader(
            new InputStreamReader(
              socket.getInputStream()));
        // Вивід автоматично обновлюється
        // класом PrintWriter:
        PrintWriter out =
```

```
        new PrintWriter(
          new BufferedWriter(
            new OutputStreamWriter(
              socket.getOutputStream())),true);
        while (true) {
          String str = in.readLine();
          if (str.equals("END")) break;
          System.out.println("Echoing: " + str);
          out.println(str);
        }
      // завжди закриваєм обидва сокети...
      } finally {
        System.out.println("closing...");
        socket.close();
      }
    } finally {
      s.close();
    }
  }
} ///:~
```

2. Клієнт

```
//: c15:JabberClient.java
// дуже простий клієнт, який просто відсилає рядки серверу
// і читає рядки, які посилає сервер
import java.net.*;
import java.io.*;

public class JabberClient {
  public static void main(String[] args)
      throws IOException {
    // Установка параметра в null в getByName()
    // повертає спеціальний IP address - "Локальну петлю",
    // для тестування на одній машині без присутності сітки
    InetAddress addr =
      InetAddress.getByName(null);
    // Альтернативно Ви можете використати
    // адресу або імя:
    // InetAddress addr =
    //    InetAddress.getByName("127.0.0.1");
    // InetAddress addr =
    //    InetAddress.getByName("localhost");
    System.out.println("addr = " + addr);
    Socket socket =
      new Socket(addr, JabberServer.PORT);
    // Окружаем все блоками try-finally to make
    // щоби переконатися що сокет закривається:
```

```
    try {
      System.out.println("socket = " + socket);
      BufferedReader in =
        new BufferedReader(
          new InputStreamReader(
            socket.getInputStream()));
      // Вивід автоматично скидається
      // за допомогою PrintWriter:
      PrintWriter out =
        new PrintWriter(
          new BufferedWriter(
            new OutputStreamWriter(
              socket.getOutputStream())),true);
      for(int i = 0; i < 10; i ++) {
        out.println("howdy " + i);
        String str = in.readLine();
        System.out.println(str);
      }
      out.println("END");
    } finally {
      System.out.println("closing...");
      socket.close();
    }
  }
} ///:~
```

ЛІТЕРАТУРА

1. Брюс Эккель Философия Java. 4-е полное изд. // Питер Пресс 1168с. – 2018
2. Кей С. Хорстманн, Гари Корнелл Кей С. Хорстманн Java SE 9. Базовый курс, 2-е издание // Вильямс, 576с. – 2018
3. Егор Бугаенко Элегантные объекты. Java Edition // Питер Пресс, 240с. – 2018
4. Себастьян Дашнер Изучаем Java EE. Современное программирование для больших предприятий // Питер Пресс, 384с. – 2018
5. Герберт Шилдт Java. Полное руководство. 10-е издание // Диалектика, 1488с. – 2018
6. Барри Берд Java для чайников, 7-е издание // Диалектика 624с. – 2018
7. Коузен К. Современный Java. Рецепты программирования// ДМК Пресс, 274с. – 2018
8. Прохоренок Н.А. Основы Java// БХВ-Петербург, 704с. – 2017
9. Роберт Седжвик, Кевин Уэйн Алгоритмы на Java, 4-е издание // Вильямс, 848с. – 2016
10. Яшин А.С., Сеттер Р.В. JAVA НА ПРИМЕРАХ. Практика, практика и только практика // Наука и Техника, 256с. -2018