Ingegneria del Software
Esercitazione

23 Ottobre 2023

Davide Yi Xian Hu

Email: davideyi.hu@polimi.it

POLITECNICO
MILANO 1863

Esercizio 1A / Access Modifiers

Se si invoca C2.m3(), e' corretto il seguente codice?

```
package A;

public class C1 {
  public void m1() {
    System.out.print(“1”);
  }
  protected void m2() {
    System.out.print(“2”);
  }
  private void m3() {
    System.out.print(“3”);
  }
}
```

```
package B;

import A.*;
public class C2 extends C1 {
  public void m1() {
    System.out.print("4");
    m2();
    m3();
  }
  protected void m2() {
    System.out.print(“5”);
  }
}
```

Esercizio 1B / Access Modifiers

Se si invoca C2.m1(), cosa stampa il programma?

```
package A;

public class C1 {
  public void m1() {
    System.out.print(“1”);
  }
  protected void m2() {
    System.out.print(“2”);
  }
  private void m3() {
    System.out.print(“3”);
  }
}
```

```
package B;

import A.*;
public class C2 extends C1 {
  public void m1() {
    System.out.print("4");
    m2();
    m3();
  }
  protected void m2() {
    System.out.print(“5”);
  }
  private void m3() {
    System.out.print(“6");
  }
}
```

Esercizio 2 / Static and Dynamic Typing

Per ogni riga, dire se il codice è' corretto e cosa stampa.

```
package C;

public class C3 {
  public static void main(String[] s) {
    C1 c1; C2 c2; Object o;
    c1 = new C1();            /* 1*/
    c1.m1();                  /* 2*/
    c2 = new C2();            /* 3*/
    c2.m2();                  /* 4*/
    c1 = c2;                  /* 5*/
    c1.m1();                  /* 6*/
    c2 = new C1();            /* 7*/
    o = new C1();             /* 8*/
    c2 = (C2) o;              /* 9*/
    o = new C2();             /*10*/
    c1 = (C1) o;              /*11*/
    c1.m1();                  /*12*/
  }
}
```

Esercizio 3 / Comparable Interface

Implementare la funzione min(List list).

- 👉 Implementare un metodo statico min che trova il minimo di una lista di oggetti che implementano l'interfaccia java.lang.Comparable

- 👉 Person implementa Comparable controllando l'ordine del cognome e poi, in caso di omonimia, il nome. Student aggiunge a questo comportamento in caso di omonimia sia sul nome che sul cognome il controllo sull'id.
  (Modificare esercizio 5, esercitazione 2)

- 👉 Ordinare una lista di studenti sia con java.util.Collections.sort che con il Comparator (invocando direttamente studentList.sort( ))

Esercizio 4 / Default Constructor

Cosa stampa il programma?

```
class Padre {
  Padre() { System.out.println(“Padre!"); }
}

class Figlio extends Padre {
  Figlio() { System.out.println(“Figlio!"); }
}

class Example {
  public static void main(String[] args){
    Figlio p = new Figlio();
  }
}
```

Esercizio 5 / Static and Dynamic Typing

Per ogni riga, dire se il codice è' corretto e cosa stampa.

```
class Person {
  void greet() {
    System.out.println("Arrivederci");
  }
}
class EasyPerson extends Person {
  void greet() {
    System.out.println("Ciao");
  }
}
class FormalPerson extends Person {
  void greet() {
    System.out.println("Saluti");
  }
}
class VeryFormalPerson extends FormalPerson {
  void greet() {
    System.out.println("Distinti saluti");
  }
}
```

```
class Example {
  public static void main(String[] args) {
    Person p = new Person();
    EasyPerson ep = new EasyPerson();
    FormalPerson fp = new FormalPerson();
    VeryFormalPerson vfp = new VeryFormalPerson();
    p.greet();                          //  1
    ep = p;                             //  2
    p = ep;                             //  3
    p.greet();                          //  4
    ep = fp;                            //  5
    ep.greet();                         //  6
    fp.greet();                         //  7
    p = new FormalPerson();             //  8
    p.greet();                          //  9
    fp = p;                             // 10
    vfp = (VeryFormalPerson) fp;        // 11
    vfp.greet();                        // 12
  }
}
```

Esercizio 6 / Hierarchical Polygons

Definire una gerarchia di poligoni e sfruttare il poliformismo.

- 👉 **Polygon** è una classe astratta che definisce il metodo astratto **getPerimeter( )**.
- 👉 **Polygon** implementa una funzione **printPerimeters( )** che stampi il perimetro di un array di poligoni.
- 👉 Implementare le sotto-classi di **Polygon Square**, **Rectangle** e **Triangle**, ognuna con la propria implementazione di **getPerimeter( )**

Esercizio 7 / Runtime Type Checking

Cosa stampa il programma?

```
class Father { }

class Son extends Father { }

class Test {
  public static void main(String[] s) {
  Father f = new Son();
  Father f2 = new Father();
  if (f instanceof Father)
    System.out.println("True");
  else
    System.out.println("False");
  if (f.getClass() = = f2.getClass())
    System.out.println("True");
  else
    System.out.println("False");
  }
}
```