Esercizi Java

Esercizio 1

Creare la classe *Rettangolo* con gli attributi larghezza e altezza, ciascuno con valore di default pari a 1. La classe include metodi che calcolano il perimetro e l'area del rettangolo. Scrivere i metodi *set* e *get* sia per larghezza che per altezza.
I metodi set devono verificare che larghezza e altezza siano double maggiori di 0.0 e minori di 20.0. Scrivete un programma che verifichi il funzionamento della classe *Rettangolo*.

Esercizio 2

Realizzare una gerarchia di classi per rappresentare figure geometriche posizionate in un piano cartesiano: rettangolo, ellisse, cerchio. Ogni figura geometrica è caratterizzata dalle coordinate cartesiano del suo baricentro e dalle misure caratteristiche della figura e deve realizzare i metodi per calcolare perimetro e area. Si realizzi una classe FigureHandler contenente dei metodi statici che svolgano le seguenti operazioni su un array di figure geometriche: - somma dei perimetri - stampa delle figure il cui baricentro si trova nel primo quadrante.

Esercizio 3

Realizzare una gerarchia di classi per rappresentare strumenti musicali di un'orchestra. Realizzare le classi astratte StrumentoMusicale e StrumentoCorde. Gli strumenti musicali sono caratterizzati dal nome del modello e dal prezzo. Gli strumenti a corde sono caratterizzati da un attributo «numero corde» e da un valore intero compreso tra 1 e 12 indicante l'accordatura. Creare le classi Sax e Chitarra. Definire i costruttori delle classi Sax e Chitarra sapendo che, di default, la chitarra ha 6 corde e la sua accordatura ha valore 1.

Realizzare poi la gerarchia di classi per rappresentare un'orchestra.
Un' orchestra è costituita da un insieme di musicisti ed è definita da un nome e da un genere. Un musicista è caratterizzato da un nome, cognome e da uno strumento musicale.
Scrivere un programma in cui si istanziano 4 oggetti associati ai membri dell' orchestra (il chitarrista «Jimy», il sassofonista «Charlie», il chitarrista «Jimmy» e il chitarrista «Eric») e si stampi a video il numero di chitarre presenti nell'orchestra.

Esercizio 4

Si considerino le seguenti dichiarazioni:

```
public abstract class Point {
   public abstract double distance (Point p);
}

public class Point1D extends Point {
   private double c1;

   public Point1D (double c1) {
      this.c1 = c1;
   }

   public double getC1() {
      return c1;
   }

   public double distance (Point p) {
      return Math.abs(((Point1D) p).getC1()-c1);
   }
}
```

```
public class Point2D extends Point1D {
   private double c2;

   public Point2D (double c1, double c2) {
      super(c1);
      this.c2 = c2;
   }

   public double getC2() {
      return c2;
   }

   public double distance (Point p) {
      return Math.sqrt(Math.pow(((Point2D) p).getC1()-this.getC1(), 2) +
                       Math.pow(((Point2D) p).getC2()-this.getC2(), 2));
   }

   public double distance (Point2D p) {
      return Math.sqrt(Math.pow(p.getC1()-this.getC1(), 2) +
                       Math.pow(p.getC2()-this.getC2(), 2));
   }
}
```

Domanda 1

Quali dei due metodi distance della classe Point2D rappresenta un caso di overloading e quale invece rappresenta un caso di overriding?

```
public abstract class Point {
   public abstract double distance (Point p);
}

public class Point1D extends Point {
   private double c1;

   public Point1D (double c1) {
      this.c1 = c1;
   }

   public double getC1() {
      return c1;
   }

   public double distance (Point p) {
      return Math.abs(((Point1D) p).getC1()-c1);
   }
}
```

```
public class Point2D extends Point1D {
   private double c2;

   public Point2D (double c1, double c2)
      super(c1);
      this.c2 = c2;
   }

   public double getC2() {
      return c2;
   }

  public double distance (Point p) {
     return Math.sqrt(Math.pow(((Point2D) p).getC1()-this.getC1(), 2) +
                      Math.pow(((Point2D) p).getC2()-this.getC2(), 2));
  }

  public double distance (Point2D p) {
     return Math.sqrt(Math.pow(p.getC1()-this.getC1(), 2) +
                      Math.pow(p.getC2()-this.getC2(), 2));
  }
```

Soluzione:

Il primo metodo distance in ordine di dichiarazione rappresenta un caso di overriding rispetto al metodo con la stessa signature della classe Point1D. Il secondo, invece, rappresenta un caso di overloading.

Dopo la seguente sequenza di dichiarazioni:

```
Point p;
Point1D p1;
Point2D p2;
```

segnare a fianco di ciascuno dei seguenti assegnamenti se essi sono considerati corretti dal compilatore Java in base alle regole di tipo:

```
p = p1;

p = p2;

p1 = p;

p1 = p2;

p2 = p1;
```

Dopo la seguente sequenza di dichiarazioni:

```
Point p;
Point1D p1;
Point2D p2;
```

segnare a fianco di ciascuno dei seguenti assegnamenti se essi sono considerati corretti dal compilatore Java in base alle regole di tipo:

```
p = p1;    /* corretta */

p = p2;    /* corretta */

p1 = p;    /* non corretta */

p1 = p2;    /* corretta */

p2 = p1;    /* non corretta */
```

Si consideri il seguente frammento e si pieghi sinteticamente a parole l'effetto di ciascun assegnamento, indicando quali metodi vengono invocati in ciascun caso e scrivendo il risultato calcolato (valore stampato a video).

```
Point p1 = new Point1D(0.0);
Point p2 = new Point1D(1.0);
Point p3 = new Point2D(0.0, 1.0);
Point p4 = new Point2D(1.0, 0.0);
double x;

x = p1.distance(p2); /* 1 */
System.out.println(x);

x = p3.distance(p4); /* 2 */
System.out.println(x);

x = p1.distance(p3); /* 3 */
System.out.println(x);

x = p3.distance(p1); /* 4 */
System.out.println(x);

p1=p3; /* 5 */
x = p2.distance(p1);


System.out.println(x);

x = p4.distance(p1); /* 6 */
System.out.println(x);
```

Esercizio 5

Realizzare una gerarchia di classi per rappresentare le persone coinvolte nello svolgimento di un esame.
Definire una classe *Persona* e le sue sottoclassi *Professore* e *Studente*. Tutte le persone sono caratterizzate da un nome, da un cognome e da un codice fiscale.
Un professore è caratterizzato da un'area di ricerca e dall'università di appartenenza.
Uno Studente può essere di tipo *studente standard* o *studente erasmus.*Tutti gli studenti sono definiti da un codice persona e dall'università di appartenenza. Per gli *studenti standard* si deve tenere traccia della loro media esami e del numero esami sostenuti; per gli *studenti erasmus* è necessario definire soltanto il numero di mesi del periodo di scambio.

Definire una classe *Esame* sapendo che un esame è composto da un insieme di prove d'esame ed è associato ad un corso e a una data. Ad ogni prova d'esame è associato uno studente e il relativo voto.

Si realizzi infine un programma che istanzia 4 oggetti di tipo studente e uno di tipo esame (e i suoi relativi campi). Il programma dovrà stampare a video la media di tutte le prove d'esame e aggiornare la media e il numero di esami sostenuti di ciascuno studente standard.