Cognome.....

Nome.....

Matricola.....

Programmazione

Prova scritta del 7 settembre 2015

TEMPO DISPONIBILE: 1 ora e 40 minuti

Negli esercizi proposti si utilizzano le seguenti classi:

- Classe astratta `Figura`.
  Ogni oggetto della classe rappresenta una figura geometrica nel piano. La classe possiede un costruttore privo di argomenti.
  Tra i metodi della classe `Figura` vi sono:

  ```
  public abstract double getArea()
  public abstract double getPerimetro()
  ```
  restituiscono, rispettivamente, l'area e il perimetro della figura che esegue il metodo.

- Classi `Rettangolo`, `Cerchio`, `Quadrato`.
  Gli oggetti rappresentano, rispettivamente, rettangoli, cerchi e quadrati. `Rettangolo` e `Cerchio` estendono direttamente `Figura`, mentre `Quadrato` estende direttamente `Rettangolo`.

  Tra i metodi forniti dalla classe `Cerchio` vi sono:

  ```
  public double getRaggio()
  ```
  restituisce la lunghezza del raggio del cerchio che esegue il metodo.

  L'unico costruttore fornito da `Cerchio` ha un argomento:

  ```
  public Cerchio(double r)
  ```
  costruisce un oggetto che rappresenta un cerchio il cui raggio ha la lunghezza ricevuta tramite il parametro.

Attenzione: nello svolgere gli esercizi fate riferimento a quanto indicato sopra e non all'implementazione delle classi, che è privata e dunque non accessibile al di fuori del codice di ciascuna delle classi stesse.

Negli esercizi 1, 2 e 3 considerate la dichiarazione di variabile `Figura[] ar`. Supponete di disporre di una porzione di codice alla fine della quale `ar` si riferisca a un array in cui ogni posizione contiene il riferimento a un oggetto di tipo `Figura` (quindi nessuna delle posizioni contiene `null`).

1. Scrivete una porzione di codice per calcolare e visualizzare sul monitor la media dei perimetri di tutte le figure contenute nell'array `ar`. Nel caso l'array sia vuoto, al posto della media il metodo deve visualizzare un messaggio opportuno.

2. Scrivete una porzione di codice per calcolare e visualizzare sul monitor la media dei perimetri di tutti i cerchi contenuti nell'array `ar`. Nel caso l'array non contenga cerchi, al posto della media il metodo deve visualizzare un messaggio opportuno.

3. Scrivete una porzione di codice per calcolare e visualizzare sul monitor la media dei raggi di tutti i cerchi contenuti nell'array `ar`. Nel caso l'array non contenga cerchi, al posto della media il metodo deve visualizzare un messaggio opportuno.

4. Considerate le seguenti classi:

```
public class Strana extends Cerchio {
  private String t;
  private static int k = 2;

  public Strana(String s) {
    super(1);
    t = s;
    k = k + s.length() + 3;
  }

  public Strana() {
    this("");
  }

  public double getArea() {
    return super.getArea() + t.length();
  }

  public static int getStatico() {
    return k;
  }
}
```

```
class Prova {
  public static void main(String[] args) {
    System.out.println(Strana.getStatico());  //1
    Strana s = new Strana("gatto");
    System.out.println((int)s.getArea());     //2
    s = new Strana();
    System.out.println((int)s.getArea());     //3
    System.out.println(Strana.getStatico());  //4
  }
}
```

Scrivete in ogni riquadro l'output prodotto dall'istruzione di stampa seguita dal commento indicato:

//1	//2	//3	//4

5. Oltre alle classi utilizzate negli esercizi precedenti, considerate una classe concreta `Gamma` che estende `Figura` e una classe concreta `Delta` che estende `Cerchio` e implementa un'interfaccia `In`.

a. Nel riquadro che precede ciascuna affermazione, scrivete V se l'affermazione è vera, F se è falsa:

- ☐ Ogni istanza di `Strana` contiene esattamente un campo (oltre a quelli ereditati dalla superclasse)
- ☐ Ogni istanza di `Strana` contiene esattamente 2 campi (oltre a quelli ereditati dalla superclasse)
- ☐ `Gamma` deve fornire l'implementazione dei metodi di `In`
- ☐ `Gamma` deve fornire l'implementazione del metodo `getArea`
- ☐ `Gamma` deve fornire l'implementazione del metodo `getRaggio`
- ☐ Se il codice sorgente della classe `Gamma` non contiene un costruttore allora il compilatore segnala un errore
- ☐ Se il codice sorgente della classe `Delta` non contiene un costruttore allora il compilatore segnala un errore
- ☐ `Delta` deve fornire l'implementazione del metodo `getArea`
- ☐ `Delta` deve fornire l'implementazione del metodo `getRaggio`
- ☐ `Delta` deve fornire l'implementazione dei metodi di `In`

b. Considerate le seguenti dichiarazioni di variabile:

```
Figura f, Gamma g, Cerchio c, Delta d, Strana s, In i;
```

Nel riquadro accanto a ciascun assegnamento scrivete SI se l'assegnamento è compilato correttamente, NO se non è compilato correttamente (supponete che al posto di `...` vi siano gli argomenti opportuni):

- ☐ `f = new Cerchio(2f)`
- ☐ `f = s`
- ☐ `s = (Strana) f`
- ☐ `f = new Figura()`
- ☐ `g = (Gamma) i`
- ☐ `d = (Delta) i`
- ☐ `f = new Cerchio(f)`
- ☐ `f = new Cerchio(f.getArea())`
- ☐ `c = (Cerchio) i`
- ☐ `f = new Cerchio(2 * f)`

6. Considerate la dichiarazione di variabile `Figura[] figure` e il seguente frammento di codice:

```
int x = 0;
try {
  for (Figura f: figure)
    x = x + (f instanceof Rettangolo ? (int) f.getArea() : 2 * x - 1);
  Rettangolo r = (Rettangolo) figure[x / x];
  x = (int) r.getArea();
} catch (ArithmeticException e) {
  x = x + 22;
} catch (ArrayIndexOutOfBoundsException e) {
  x = x + 32;
} catch (NullPointerException e) {
  x = x + 42;
} catch (ClassCastException e) {
  x = x + 52;
}
```

Ricordando che:

- `ArithmeticException` viene sollevata in caso di anomalie nel calcolo di operazioni aritmetiche,
- `ArrayIndexOutOfBoundsException` viene sollevata quando si tenti di accedere a una posizione inesistente in un array,
- `NullPointerException` viene sollevata quando si tenti di accedere a un oggetto tramite un riferimento `null`,
- `ClassCastException` viene sollevata quando si tenti di forzare un oggetto a una sottoclasse di cui l'oggetto non è istanza,

indicate nel riquadro corrispondente, in ciascuno dei seguenti casi, il valore della variabile `x` dopo l'esecuzione:

(a) l'array riferito da `figure` contiene (nell'ordine indicato) riferimenti a oggetti che rappresentano un quadrato di lato `1`, un cerchio di raggio `3`, un quadrato di lato `2`.

(b) l'array riferito da `figure` contiene (nell'ordine indicato) riferimenti a oggetti che rappresentano un cerchio di raggio `4`, un quadrato di lato `1`.

(c) `figure` contiene `null`.

(d) l'array riferito da `figure` contiene (nell'ordine indicato) riferimenti a oggetti che rappresentano un quadrato di lato `1`, un quadrato di lato `2`, un cerchio di raggio `3`.

7. Considerate il seguente metodo ricorsivo. Scrivete il risultato restituito da ciascuna delle chiamate indicate nei due riquadri:

```
... double f(double x) {
  if (x != (int) x)
    return x;
  else
    return x + 2 * f (x / 2);
}
```

f(1)	f(4)

Cognome.....

Nome.....

Matricola.....

Programmazione

Prova scritta del 7 settembre 2015

TEMPO DISPONIBILE: 1 ora e 40 minuti

Negli esercizi proposti si utilizzano le seguenti classi:

- Classe astratta `Figura`.
  Ogni oggetto della classe rappresenta una figura geometrica nel piano. La classe possiede un costruttore privo di argomenti.
  Tra i metodi della classe `Figura` vi sono:

  ```
  public abstract double getArea()
  public abstract double getPerimetro()
  ```
  restituiscono, rispettivamente, l'area e il perimetro della figura che esegue il metodo.

- Classi `Rettangolo`, `Cerchio`, `Quadrato`.
  Gli oggetti rappresentano, rispettivamente, rettangoli, cerchi e quadrati. `Rettangolo` e `Cerchio` estendono direttamente `Figura`, mentre `Quadrato` estende direttamente `Rettangolo`.

  Tra i metodi forniti dalla classe `Cerchio` vi sono:

  ```
  public double getRaggio()
  ```
  restituisce la lunghezza del raggio del cerchio che esegue il metodo.

  L'unico costruttore fornito da `Cerchio` ha un argomento:

  ```
  public Cerchio(double r)
  ```
  costruisce un oggetto che rappresenta un cerchio il cui raggio ha la lunghezza ricevuta tramite il parametro.

Attenzione: nello svolgere gli esercizi fate riferimento a quanto indicato sopra e non all'implementazione delle classi, che è privata e dunque non accessibile al di fuori del codice di ciascuna delle classi stesse.

Negli esercizi 1, 2 e 3 considerate la dichiarazione di variabile `Figura[] vet`. Supponete di disporre di una porzione di codice alla fine della quale `vet` si riferisca a un array in cui ogni posizione contiene il riferimento a un oggetto di tipo `Figura` (quindi nessuna delle posizioni contiene `null`).

1. Scrivete una porzione di codice per calcolare e visualizzare sul monitor la media dei perimetri di tutte le figure contenute nell'array `vet`. Nel caso l'array sia vuoto, al posto della media il metodo deve visualizzare un messaggio opportuno.

2. Scrivete una porzione di codice per calcolare e visualizzare sul monitor la media dei perimetri di tutti i cerchi contenuti nell'array `vet`. Nel caso l'array non contenga cerchi, al posto della media il metodo deve visualizzare un messaggio opportuno.

3. Scrivete una porzione di codice per calcolare e visualizzare sul monitor la media dei raggi di tutti i cerchi contenuti nell'array `vet`. Nel caso l'array non contenga cerchi, al posto della media il metodo deve visualizzare un messaggio opportuno.

4. Considerate le seguenti classi:

```
public class Strana extends Cerchio {
  private String t;
  private static int k = 4;

  public Strana(String s) {
    super(1);
    t = s;
    k = k + s.length() + 2;
  }

  public Strana() {
    this("");
  }

  public double getArea() {
    return super.getArea() + t.length();
  }

  public static int getStatico() {
    return k;
  }
}
```

```
class Prova {
  public static void main(String[] args) {
    System.out.println(Strana.getStatico());  //1
    Strana s = new Strana("ape");
    System.out.println((int)s.getArea());     //2
    s = new Strana();
    System.out.println((int)s.getArea());     //3
    System.out.println(Strana.getStatico());  //4
  }
}
```

Scrivete in ogni riquadro l'output prodotto dall'istruzione di stampa seguita dal commento indicato:

//1	//2	//3	//4

5. Oltre alle classi utilizzate negli esercizi precedenti, considerate una classe concreta `Alfa` che estende `Figura` e una classe concreta `Beta` che estende `Cerchio` e implementa un'interfaccia `In`.

a. Nel riquadro che precede ciascuna affermazione, scrivete V se l'affermazione è vera, F se è falsa:

- ☐ Se il codice sorgente della classe `Alfa` non contiene un costruttore allora il compilatore segnala un errore
- ☐ Se il codice sorgente della classe `Beta` non contiene un costruttore allora il compilatore segnala un errore
- ☐ `Beta` deve fornire l'implementazione del metodo `getArea`
- ☐ `Beta` deve fornire l'implementazione del metodo `getRaggio`
- ☐ `Beta` deve fornire l'implementazione dei metodi di `In`
- ☐ `Alfa` deve fornire l'implementazione dei metodi di `In`
- ☐ `Alfa` deve fornire l'implementazione del metodo `getArea`
- ☐ `Alfa` deve fornire l'implementazione del metodo `getRaggio`
- ☐ Ogni istanza di `Strana` contiene esattamente un campo (oltre a quelli ereditati dalla superclasse)
- ☐ Ogni istanza di `Strana` contiene esattamente 2 campi (oltre a quelli ereditati dalla superclasse)

b. Considerate le seguenti dichiarazioni di variabile:

```
Figura f, Alfa a, Cerchio c, Beta b, Strana s, In i;
```

Nel riquadro accanto a ciascun assegnamento scrivete SI se l'assegnamento è compilato correttamente, NO se non è compilato correttamente (supponete che al posto di `...` vi siano gli argomenti opportuni):

- ☐ `f = s`
- ☐ `f = new Cerchio(f)`
- ☐ `f = new Cerchio(f.getArea())`
- ☐ `c = (Cerchio) i`
- ☐ `s = (Strana) f`
- ☐ `f = new Figura()`
- ☐ `a = (Alfa) i`
- ☐ `b = (Beta) i`
- ☐ `f = new Cerchio(2f)`
- ☐ `f = new Cerchio(2 * f)`

6. Considerate la dichiarazione di variabile `Figura[] figure` e il seguente frammento di codice:

```
int x = 0;
try {
  for (Figura f: figure)
    x = x + (f instanceof Rettangolo ? (int) f.getArea() : 2 * x - 1);
  Rettangolo r = (Rettangolo) figure[x / x];
  x = (int) r.getArea();
} catch (ArithmeticException e) {
  x = x + 24;
} catch (ArrayIndexOutOfBoundsException e) {
  x = x + 34;
} catch (NullPointerException e) {
  x = x + 44;
} catch (ClassCastException e) {
  x = x + 54;
}
```

Ricordando che:

- `ArithmeticException` viene sollevata in caso di anomalie nel calcolo di operazioni aritmetiche,
- `ArrayIndexOutOfBoundsException` viene sollevata quando si tenti di accedere a una posizione inesistente in un array,
- `NullPointerException` viene sollevata quando si tenti di accedere a un oggetto tramite un riferimento `null`,
- `ClassCastException` viene sollevata quando si tenti di forzare un oggetto a una sottoclasse di cui l'oggetto non è istanza,

indicate nel riquadro corrispondente, in ciascuno dei seguenti casi, il valore della variabile `x` dopo l'esecuzione:

(a) `figure` contiene `null`. ☐

(b) l'array riferito da `figure` contiene (nell'ordine indicato) riferimenti a oggetti che rappresentano un quadrato di lato `1`, un quadrato di lato `2`, un cerchio di raggio `3`. ☐

(c) l'array riferito da `figure` contiene (nell'ordine indicato) riferimenti a oggetti che rappresentano un quadrato di lato `1`, un cerchio di raggio `3`, un quadrato di lato `2`. ☐

(d) l'array riferito da `figure` contiene (nell'ordine indicato) riferimenti a oggetti che rappresentano un cerchio di raggio `4`, un quadrato di lato `1`. ☐

7. Considerate il seguente metodo ricorsivo. Scrivete il risultato restituito da ciascuna delle chiamate indicate nei due riquadri:

```
... double f(double x) {
  if (x != (int) x)
    return x;
  else
    return x + 2 * f (x / 2);
}
```

f(1)	f(4)