

Cognome.....

Nome.....

Matricola.....

Programmazione

Prova scritta del 21 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 **Quadrato** vi è:

```
public double getLato()  
restituisce la lunghezza del lato del quadrato che esegue il metodo.
```

L'unico costruttore fornito da **Quadrato** ha un argomento:

```
public Quadrato(double l)  
costruisce un oggetto che rappresenta un quadrato il cui lato 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 delle aree 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 delle aree di tutti i quadrati contenuti nell'array `ar`. Nel caso l'array non contenga quadrati, 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 lati di tutti i quadrati contenuti nell'array `ar`. Nel caso l'array non contenga quadrati, al posto della media il metodo deve visualizzare un messaggio opportuno.

4. Considerate le seguenti classi:

```
public class Strana extends Quadrato {
    private int t;
    private static int k = 10;

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

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

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

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

```
class Prova {
    public static void main(String[] args) {
        Strana s = new Strana("ape");
        System.out.println((int)s.getArea()); //1
        System.out.println(Strana.getStatico()); //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 **Quadrato** e una classe concreta **Delta** che estende **Figura** 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 **Delta** non contiene un costruttore allora il compilatore segnala un errore
- ☐ **Delta** deve fornire l'implementazione del metodo **getArea**
- ☐ **Gamma** deve fornire l'implementazione dei metodi di **In**
- ☐ **Gamma** deve fornire l'implementazione del metodo **getArea**
- ☐ **Gamma** deve fornire l'implementazione del metodo **getLato**
- ☐ Se il codice sorgente della classe **Gamma** non contiene un costruttore allora il compilatore segnala un errore
- ☐ **Delta** deve fornire l'implementazione del metodo **getLato**
- ☐ **Delta** deve fornire l'implementazione dei metodi di **In**
- ☐ 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, **Gamma g**, **Rettangolo r**, **Quadrato q**, **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 Quadrato(f.getArea())**
- ☐ **f = r**
- ☐ **r = (Strana) f**
- ☐ **f = new Figura()**
- ☐ **r = new Quadrato(2.0)**
- ☐ **d = (Delta) i**
- ☐ **q = r**
- ☐ **r = q**
- ☐ **g = s**
- ☐ **s = (Quadrato) r**

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

```
int x = 5;
try {
    for (Figura f: figure)
        x = x - (int)(f instanceof Quadrato ? ((Quadrato)f).getLato() : f.getArea());
    Quadrato r = (Quadrato) figure[x / x];
    x = (int) r.getArea();
} catch (ArithmeticException e) {
    x = 24 - x;
} catch (ArrayIndexOutOfBoundsException e) {
    x = 34 - x;
} catch (NullPointerException e) {
    x = 44 - x;
} catch (ClassCastException e) {
    x = 54 - x;
}
```

Ricordando che:

- `ArithmeticException` viene sollevata in caso di anomalie nel calcolo di operazioni aritmetiche,
- `ArrayIndexOutOfBoundsException` viene sollevata quando si tenta di accedere a una posizione inesistente in un array,
- `NullPointerException` viene sollevata quando si tenta di accedere a un oggetto tramite un riferimento `null`,
- `ClassCastException` viene sollevata quando si tenta 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 2, un quadrato di lato 3, un cerchio di raggio 1.
- (b) `figure` contiene `null`.
- (c) l'array riferito da `figure` contiene (nell'ordine indicato) riferimenti a oggetti che rappresentano un cerchio di raggio 1, un quadrato di lato 2.
- (d) l'array riferito da `figure` contiene (nell'ordine indicato) riferimenti a oggetti che rappresentano un quadrato di lato 3, un cerchio di raggio 1, un quadrato di lato 2.

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

```
... Quadrato f(int x) {
    if (x <= 2)
        return new Quadrato(x);
    else
        return new Quadrato(3 + 2 * f(x / 2).getLato());
}
```

<code>f(2).getArea()</code>	<code>f(5).getArea()</code>

Cognome.....

Nome.....

Matricola

Programmazione

Prova scritta del 21 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 **Quadrato** vi è:

```
public double getLato()  
restituisce la lunghezza del lato del quadrato che esegue il metodo.
```

L'unico costruttore fornito da **Quadrato** ha un argomento:

```
public Quadrato(double l)  
costruisce un oggetto che rappresenta un quadrato il cui lato 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 delle aree 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 delle aree di tutti i quadrati contenuti nell'array `vet`. Nel caso l'array non contenga quadrati, 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 lati di tutti i quadrati contenuti nell'array `vet`. Nel caso l'array non contenga quadrati, al posto della media il metodo deve visualizzare un messaggio opportuno.

4. Considerate le seguenti classi:

```
public class Strana extends Quadrato {
    private int t;
    private static int k = 7;

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

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

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

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

```
class Prova {
    public static void main(String[] args) {
        Strana s = new Strana("bue");
        System.out.println((int)s.getArea());    //1
        System.out.println(Strana.getStatico()); //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 **Quadrato** e una classe concreta **Beta** che estende **Figura** 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
- ☐ 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)
- ☐ **Beta** deve fornire l'implementazione del metodo **getLato**
- ☐ **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**
- ☐ 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**
- ☐ **Alfa** deve fornire l'implementazione del metodo **getLato**

b. Considerate le seguenti dichiarazioni di variabile:

Figura f, Alfa a, Rettangolo r, Quadrato q, 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 = r**
- ☐ **q = r**
- ☐ **r = q**
- ☐ **a = s**
- ☐ **r = (Strana) f**
- ☐ **f = new Figura()**
- ☐ **r = new Quadrato(2.0)**
- ☐ **b = (Beta) i**
- ☐ **f = new Quadrato(f.getArea())**
- ☐ **s = (Quadrato) r**

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

```
int x = 5;
try {
    for (Figura f: figure)
        x = x - (int)(f instanceof Quadrato ? ((Quadrato)f).getLato() : f.getArea());
    Quadrato r = (Quadrato) figure[x / x];
    x = (int) r.getArea();
} catch (ArithmeticException e) {
    x = 22 - x;
} catch (ArrayIndexOutOfBoundsException e) {
    x = 32 - x;
} catch (NullPointerException e) {
    x = 42 - x;
} catch (ClassCastException e) {
    x = 52 - x;
}
```

Ricordando che:

- `ArithmeticException` viene sollevata in caso di anomalie nel calcolo di operazioni aritmetiche,
- `ArrayIndexOutOfBoundsException` viene sollevata quando si tenta di accedere a una posizione inesistente in un array,
- `NullPointerException` viene sollevata quando si tenta di accedere a un oggetto tramite un riferimento `null`,
- `ClassCastException` viene sollevata quando si tenta 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 cerchio di raggio 1, un quadrato di lato 2.
- (b) l'array riferito da `figure` contiene (nell'ordine indicato) riferimenti a oggetti che rappresentano un quadrato di lato 3, un cerchio di raggio 1, un quadrato di lato 2.
- (c) l'array riferito da `figure` contiene (nell'ordine indicato) riferimenti a oggetti che rappresentano un quadrato di lato 2, un quadrato di lato 3, un cerchio di raggio 1.
- (d) `figure` contiene `null`.

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

```
... Quadrato f(int x) {
    if (x <= 2)
        return new Quadrato(x);
    else
        return new Quadrato(3 + 2 * f(x / 2).getLato());
}
```

<code>f(2).getArea()</code>	<code>f(5).getArea()</code>