Cognome.....

Nome.....

Matricola.....

Programmazione

Prova scritta del 1° febbraio 2016

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()`
  restituisce l'area della figura che esegue il metodo;

  `public boolean haPerimetroMinore(Figura altra)`
  restituisce `true` se e solo se il perimetro della figura che esegue il metodo è minore di quello della figura di cui viene fornito il riferimento tramite l'argomento.

- 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 `Rettangolo` vi sono:

  `public double getBase()`
  `public double getAltezza()`
  restituiscono, rispettivamente, la base e l'altezza del rettangolo che esegue il metodo.

  L'unico costruttore fornito da `Rettangolo` ha due argomenti:

  `public Rettangolo(double b, double a)`
  costruisce un oggetto che rappresenta un rettangolo, la cui base e la cui altezza hanno le lunghezze fornite, rispettivamente, tramite il primo e il secondo 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.
Per le classi elencate in precedenza, potete servirvi esclusivamente dei metodi indicati sopra.

Negli esercizi 1, 2 e 3 considerate le dichiarazioni di variabile `Figura[] ar` e `Rettangolo r`. 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`) e `r` si riferisca a un oggetto di tipo `Rettangolo` costruito in precedenza.

1. Scrivete una porzione di codice per calcolare e visualizzare sul monitor la somma delle *aree* delle figure contenute nell'array `ar` il cui *perimetro* sia minore di quello del rettangolo riferito da `r`.

2. Scrivete una porzione di codice per calcolare e visualizzare sul monitor la somma delle *aree* dei rettangoli (inclusi i quadrati) contenuti nell'array `ar` il cui *perimetro* sia minore di quello del rettangolo riferito da `r`.

3. Scrivete una porzione di codice per calcolare e visualizzare sul monitor il numero di rettangoli (inclusi i quadrati) contenuti nell'array `ar` che hanno base minore della base del rettangolo riferito da `r`.

4. Considerate le seguenti classi:

```
public class Strana extends Rettangolo {
  private String s;
  private static int k = 2;

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

  public double getArea() {
    return super.getArea() - s.length();
  }

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

```
class Prova {
  public static void main(String[] args) {
    System.out.println(Strana.getStatico());  //1
    Rettangolo r = new Rettangolo(3, 5);
    System.out.println(r.getArea());          //2
    r = new Strana("gatto");
    System.out.println(r.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 `Strana` 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 del metodo `haPerimetroMinore`
- ☐ `Gamma` deve fornire l'implementazione del metodo `getArea`
- ☐ `Gamma` deve fornire l'implementazione del metodo `getBase`
- ☐ 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
- ☐ Nel codice di `Delta` è possibile accedere al campo `s` dichiarato in `Strana`
- ☐ `Delta` deve fornire l'implementazione del metodo `getBase`
- ☐ `Delta` deve fornire l'implementazione dei metodi di `In`

b. Considerate le seguenti dichiarazioni di variabile:

```
Figura f, Rettangolo r, Gamma g, 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:

- ☐ `f = new Rettangolo(s.getAltezza(), s.getBase())`
- ☐ `r = new Rettangolo(r.getAltezza(), r.getBase())`
- ☐ `r = (Rettangolo) f`
- ☐ `f = new Figura()`
- ☐ `g = (Gamma) i`
- ☐ `d = (Delta) i`
- ☐ `d = s`
- ☐ `s = d`
- ☐ `r = (Delta) i`
- ☐ `f = (Gamma) i`

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

```
int x = 4;
try {
  for (Figura f: figure)
    x = f instanceof Rettangolo ? (int)((Rettangolo) f).getBase() : x / 2;
  Quadrato r = (Quadrato) figure[x / x];
  x = (int) r.getArea();
} catch (ArithmeticException e) {
  x = 22 + x;
} catch (ArrayIndexOutOfBoundsException e) {
  x = 42 + x;
} catch (NullPointerException e) {
  x = 52 + x;
} catch (ClassCastException e) {
  x = 32 + x;
}
```

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 `2`, un quadrato di lato `1`, un cerchio di raggio `3`. ☐

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

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

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

```
... int f(int x) {
  if (x <= 1)
    return 2;
  else if (x % 2 == 0)
    return 2 * f(x / 2);
  else
    return f(x + 1) + 1;
}
```

f(4)	f(6)

Cognome.....

Nome.....

Matricola.....

Programmazione

Prova scritta del 1° febbraio 2016

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()`
  restituisce l'area della figura che esegue il metodo;

  `public boolean haPerimetroMaggiore(Figura altra)`
  restituisce `true` se e solo se il perimetro della figura che esegue il metodo è maggiore di quello della figura di cui viene fornito il riferimento tramite l'argomento.

- 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 `Rettangolo` vi sono:

  ```
  public double getBase()
  public double getAltezza()
  ```
  restituiscono, rispettivamente, la base e l'altezza del rettangolo che esegue il metodo.

  L'unico costruttore fornito da `Rettangolo` ha due argomenti:

  `public Rettangolo(double b, double a)`
  costruisce un oggetto che rappresenta un rettangolo, la cui base e la cui altezza hanno le lunghezze fornite, rispettivamente, tramite il primo e il secondo 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.
Per le classi elencate in precedenza, potete servirvi esclusivamente dei metodi indicati sopra.

Negli esercizi 1, 2 e 3 considerate le dichiarazioni di variabile `Figura[] vet` e `Rettangolo r`. 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`) e `r` si riferisca a un oggetto di tipo `Rettangolo` costruito in precedenza.

1. Scrivete una porzione di codice per calcolare e visualizzare sul monitor la somma delle *aree* delle figure contenute nell'array `vet` il cui *perimetro* sia maggiore di quello del rettangolo riferito da `r`.

2. Scrivete una porzione di codice per calcolare e visualizzare sul monitor la somma delle *aree* dei rettangoli (inclusi i quadrati) contenuti nell'array `vet` il cui *perimetro* sia maggiore di quello del rettangolo riferito da `r`.

3. Scrivete una porzione di codice per calcolare e visualizzare sul monitor il numero di rettangoli (inclusi i quadrati) contenuti nell'array `vet` che hanno base maggiore della base del rettangolo riferito da `r`.

4. Considerate le seguenti classi:

```
public class Strana extends Rettangolo {
  private String s;
  private static int k = 4;

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

  public double getArea() {
    return super.getArea() - s.length();
  }

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

```
class Prova {
  public static void main(String[] args) {
    System.out.println(Strana.getStatico());  //1
    Rettangolo r = new Rettangolo(2, 7);
    System.out.println(r.getArea());          //2
    r = new Strana("ape");
    System.out.println(r.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 `Strana` 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
- ☐ Nel codice di `Beta` è possibile accedere al campo `s` dichiarato in `Strana`
- ☐ `Beta` deve fornire l'implementazione del metodo `getBase`
- ☐ `Beta` deve fornire l'implementazione dei metodi di `In`
- ☐ `Alfa` deve fornire l'implementazione del metodo `haPerimetroMaggiore`
- ☐ `Alfa` deve fornire l'implementazione del metodo `getArea`
- ☐ `Alfa` deve fornire l'implementazione del metodo `getBase`
- ☐ 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, Rettangolo r, Alfa a, 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:

- ☐ `r = new Rettangolo(r.getAltezza(), r.getBase())`
- ☐ `b = s`
- ☐ `s = b`
- ☐ `r = (Beta) i`
- ☐ `r = (Rettangolo) f`
- ☐ `f = new Figura()`
- ☐ `a = (Alfa) i`
- ☐ `b = (Beta) i`
- ☐ `f = new Rettangolo(s.getAltezza(), s.getBase())`
- ☐ `f = (Alfa) i`

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

```
int x = 4;
try {
  for (Figura f: figure)
    x = f instanceof Rettangolo ? (int)((Rettangolo) f).getBase() : x / 2;
  Quadrato r = (Quadrato) figure[x / x];
  x = (int) r.getArea();
} catch (ArithmeticException e) {
  x = 42 + x;
} catch (ArrayIndexOutOfBoundsException e) {
  x = 52 + x;
} catch (NullPointerException e) {
  x = 32 + x;
} catch (ClassCastException e) {
  x = 22 + x;
}
```

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 `2`, un cerchio di raggio `3`, un quadrato di lato `1`.

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

(c) `figure` contiene `null`.

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

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

```
... int f(int x) {
  if (x <= 1)
    return 3;
  else if (x % 2 == 0)
    return 2 * f(x / 2);
  else
    return f(x + 1) - 1;
}
```

f(4)	f(6)

Cognome.....

Nome.....

Matricola.....

Programmazione

Prova scritta del 1° febbraio 2016

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 getPerimetro()`
  restituisce il perimetro della figura che esegue il metodo;

  `public boolean haAreaMinore(Figura altra)`
  restituisce `true` se e solo se l'area della figura che esegue il metodo è minore di quella della figura di cui viene fornito il riferimento tramite l'argomento.

- 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 `Rettangolo` vi sono:

  `public double getBase()`
  `public double getAltezza()`
  restituiscono, rispettivamente, la base e l'altezza del rettangolo che esegue il metodo.

  L'unico costruttore fornito da `Rettangolo` ha due argomenti:

  `public Rettangolo(double b, double a)`
  costruisce un oggetto che rappresenta un rettangolo, la cui base e la cui altezza hanno le lunghezze fornite, rispettivamente, tramite il primo e il secondo 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.
Per le classi elencate in precedenza, potete servirvi esclusivamente dei metodi indicati sopra.

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

1. Scrivete una porzione di codice per calcolare e visualizzare sul monitor la somma dei *perimetri* delle figure contenute nell'array `seq` la cui *area* sia minore di quella del rettangolo riferito da `r`.

2. Scrivete una porzione di codice per calcolare e visualizzare sul monitor la somma dei *perimetri* dei rettangoli (inclusi i quadrati) contenuti nell'array `seq` la cui *area* sia minore di quella del rettangolo riferito da `r`.

3. Scrivete una porzione di codice per calcolare e visualizzare sul monitor il numero di rettangoli (inclusi i quadrati) contenuti nell'array `seq` che hanno base minore della base del rettangolo riferito da `r`.

4. Considerate le seguenti classi:

```
public class Strana extends Rettangolo {
  private String s;
  private static int k = 3;

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

  public double getPerimetro() {
    return super.getPerimetro() - s.length();
  }

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

```
class Prova {
  public static void main(String[] args) {
    System.out.println(Strana.getStatico());  //1
    Rettangolo r = new Rettangolo(4, 6);
    System.out.println(r.getPerimetro());     //2
    r = new Strana("cicala");
    System.out.println(r.getPerimetro());     //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 `Delta` che estende `Figura` e una classe concreta `Alfa` che estende `Strana` 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 2 campi (oltre a quelli ereditati dalla superclasse)
- ☐ `Alfa` deve fornire l'implementazione dei metodi di `In`
- ☐ `Delta` deve fornire l'implementazione del metodo `haAreaMinore`
- ☐ Ogni istanza di `Strana` contiene esattamente un campo (oltre a quelli ereditati dalla superclasse)
- ☐ Se il codice sorgente della classe `Delta` non contiene un costruttore allora il compilatore segnala un errore
- ☐ Se il codice sorgente della classe `Alfa` non contiene un costruttore allora il compilatore segnala un errore
- ☐ Nel codice di `Alfa` è possibile accedere al campo `s` dichiarato in `Strana`
- ☐ `Alfa` deve fornire l'implementazione del metodo `getBase`
- ☐ `Delta` deve fornire l'implementazione del metodo `getPerimetro`
- ☐ `Delta` deve fornire l'implementazione del metodo `getBase`

b. Considerate le seguenti dichiarazioni di variabile:

```
Figura f, Rettangolo r, Delta d, Alfa a, Strana s, In i;
```

Nel riquadro accanto a ciascun assegnamento scrivete SI se l'assegnamento è compilato correttamente, NO se non è compilato correttamente:

- ☐ `r = (Alfa) i`
- ☐ `f = (Delta) i`
- ☐ `d = (Delta) i`
- ☐ `a = (Alfa) i`
- ☐ `a = s`
- ☐ `f = new Rettangolo(s.getAltezza(), s.getBase())`
- ☐ `r = new Rettangolo(r.getAltezza(), r.getBase())`
- ☐ `r = (Rettangolo) f`
- ☐ `f = new Figura()`
- ☐ `s = a`

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

```
int x = 4;
try {
  for (Figura f: figure)
    x = f instanceof Rettangolo ? (int)((Rettangolo) f).getBase() : x / 2;
  Quadrato r = (Quadrato) figure[x / x];
  x = (int) r.getArea();
} catch (ArithmeticException e) {
  x = 32 + x;
} catch (ArrayIndexOutOfBoundsException e) {
  x = 22 + x;
} catch (NullPointerException e) {
  x = 52 + x;
} catch (ClassCastException e) {
  x = 42 + x;
}
```

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 `2`, un quadrato di lato `1`, un cerchio di raggio `3`.

(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 `2`, un cerchio di raggio `3`, un quadrato di lato `1`.

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

```
... int f(int x) {
  if (x <= 1)
    return 3;
  else if (x % 2 == 0)
    return 2 * f(x / 2);
  else
    return f(x + 1) - 2;
}
```

f(4)	f(6)

Cognome.....

Nome.....

Matricola.....

Programmazione

Prova scritta del 1° febbraio 2016

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 getPerimetro()`
  restituisce il perimetro della figura che esegue il metodo;

  `public boolean haAreaMaggiore(Figura altra)`
  restituisce `true` se e solo se l'area della figura che esegue il metodo è maggiore di quella della figura di cui viene fornito il riferimento tramite l'argomento.

- 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 `Rettangolo` vi sono:

  ```
  public double getBase()
  public double getAltezza()
  ```
  restituiscono, rispettivamente, la base e l'altezza del rettangolo che esegue il metodo.

  L'unico costruttore fornito da `Rettangolo` ha due argomenti:

  `public Rettangolo(double b, double a)`
  costruisce un oggetto che rappresenta un rettangolo, la cui base e la cui altezza hanno le lunghezze fornite, rispettivamente, tramite il primo e il secondo 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.
Per le classi elencate in precedenza, potete servirvi esclusivamente dei metodi indicati sopra.

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

1. Scrivete una porzione di codice per calcolare e visualizzare sul monitor la somma dei *perimetri* delle figure contenute nell'array `ele` la cui *area* sia maggiore di quella del rettangolo riferito da `r`.

2. Scrivete una porzione di codice per calcolare e visualizzare sul monitor la somma dei *perimetri* dei rettangoli (inclusi i quadrati) contenuti nell'array `ele` la cui *area* sia maggiore di quella del rettangolo riferito da `r`.

3. Scrivete una porzione di codice per calcolare e visualizzare sul monitor il numero di rettangoli (inclusi i quadrati) contenuti nell'array `ele` che hanno base maggiore della base del rettangolo riferito da `r`.

4. Considerate le seguenti classi:

```
public class Strana extends Rettangolo {
  private String s;
  private static int k = 5;

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

  public double getPerimetro() {
    return super.getPerimetro() - s.length();
  }

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

```
class Prova {
  public static void main(String[] args) {
    System.out.println(Strana.getStatico());  //1
    Rettangolo r = new Rettangolo(6, 3);
    System.out.println(r.getPerimetro());     //2
    r = new Strana("topo");
    System.out.println(r.getPerimetro());     //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 `Beta` che estende `Figura` e una classe concreta `Gamma` che estende `Strana` e implementa un'interfaccia `In`.

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

- ☐ `Gamma` deve fornire l'implementazione del metodo `getBase`
- ☐ `Gamma` 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)
- ☐ Se il codice sorgente della classe `Beta` non contiene un costruttore allora il compilatore segnala un errore
- ☐ Se il codice sorgente della classe `Gamma` non contiene un costruttore allora il compilatore segnala un errore
- ☐ Nel codice di `Gamma` è possibile accedere al campo `s` dichiarato in `Strana`
- ☐ `Beta` deve fornire l'implementazione del metodo `haAreaMaggiore`
- ☐ `Beta` deve fornire l'implementazione del metodo `getPerimetro`
- ☐ `Beta` deve fornire l'implementazione del metodo `getBase`

b. Considerate le seguenti dichiarazioni di variabile:

```
Figura f, Rettangolo r, Beta b, Gamma g, Strana s, In i;
```

Nel riquadro accanto a ciascun assegnamento scrivete SI se l'assegnamento è compilato correttamente, NO se non è compilato correttamente:

- ☐ `g = s`
- ☐ `b = (Beta) i`
- ☐ `g = (Gamma) i`
- ☐ `s = g`
- ☐ `r = new Rettangolo(r.getAltezza(), r.getBase())`
- ☐ `r = (Rettangolo) f`
- ☐ `f = new Figura()`
- ☐ `r = (Gamma) i`
- ☐ `f = (Beta) i`
- ☐ `f = new Rettangolo(s.getAltezza(), s.getBase())`

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

```
int x = 4;
try {
  for (Figura f: figure)
    x = f instanceof Rettangolo ? (int)((Rettangolo) f).getBase() : x / 2;
  Quadrato r = (Quadrato) figure[x / x];
  x = (int) r.getArea();
} catch (ArithmeticException e) {
  x = 52 + x;
} catch (ArrayIndexOutOfBoundsException e) {
  x = 42 + x;
} catch (NullPointerException e) {
  x = 22 + x;
} catch (ClassCastException e) {
  x = 32 + x;
}
```

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 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 `2`, un cerchio di raggio `3`, un quadrato di lato `1`.

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

(d) `figure` contiene `null`.

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

```
... int f(int x) {
  if (x <= 1)
    return 2;
  else if (x % 2 == 0)
    return 2 * f(x / 2);
  else
    return f(x + 1) - 2;
}
```

f(4)	f(6)