In blu sono indicate le risposte agli esercizi. Naturalmente per gli esercizi 1, 2, 3, in cui è richiesta la scrittura di codice, sono possibili differenti soluzioni.

In rosso sono riportati alcuni commenti relativi all'esercizio e alla soluzione.

Cognome.....

Nome.....

Matricola.....

Programmazione

Prova scritta del 3 febbraio 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 `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.

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 somma dei perimetri di tutte le figure contenute nell'array `ar`.

```
double sommaPerimetri = 0;
for (Figura f: ar)
  sommaPerimetri = sommaPerimetri + f.getPerimetro();
System.out.println(sommaPerimetri);
```

Il metodo getPerimetro è disponibile già nella classe Figura e dunque può essere richiamato utilizzando il riferimento f, di tipo Figura.
È inutile e sbagliato controllare il tipo della figura e poi eseguire una forzatura per richiamare getPerimetro:
- inutile, perché come indicato il metodo è già disponibile a livello di Figura
- sbagliato, perché la classe Figura potrebbe avere altre sottoclassi oltre a quelle indicate, che dunque non verrebbero considerate.

Si poteva utilizzare un normale ciclo for al posto del ciclo for-each. Tuttavia è preferibile quest'ultimo in quanto più compatto.

2. Scrivete una porzione di codice per calcolare e visualizzare sul monitor la somma dei perimetri di tutti i rettangoli (inclusi i quadrati) contenuti nell'array `ar`.

```
double sommaPerimetriRett = 0;
for (Figura f: ar)
  if (f instanceof Rettangolo)
    sommaPerimetriRett = sommaPerimetriRett + f.getPerimetro();
System.out.println(sommaPerimetriRett);
```

L'operatore instanceof permette di controllare se l'oggetto riferito da f è un'istanza di Rettangolo. Poiché Quadrato è una sottoclasse di Rettangolo, le sue istanze sono anche istanze di Rettangolo. Pertanto, dovendo includere anche i quadrati, non è necessario aggiungere ulteriori controlli. Anche in questo caso, il metodo getPerimetro può essere richiamato tramite il riferimento f, essendo già disponibile a livello della classe Figura.

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 dell'altezza.

```
int nRett = 0;
for (Figura f: ar)
  if (f instanceof Rettangolo) {
    Rettangolo r = (Rettangolo) f;
    if (r.getBase() < r.getAltezza())
      nRett++;
  }
System.out.println(nRett);
```

A differenza dell'esercizio precedente, qui occorre eseguire i metodi getBase e getAltezza che sono disponibili nella classe Rettangolo, ma non in Figura. Dopo avere verificato che l'oggetto riferito da f è effettivamente un Rettangolo, è dunque necessaria la forzatura per disporre di un riferimento tramite il quale richiamare tali metodi.

4. Considerate le seguenti classi:

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

  public Strana(double s, int z) {
    super(s, s);
    t = z;
    k = k + (int) s;
  }

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

  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, 8);
    System.out.println(r.getArea());          //2
    r = new Strana(3, 8);
    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	24	17	8

Attenzione alla differenza tra campi e campi statici!

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

Per risolvere questo esercizio è utile disegnare la gerarchia dei tipi coinvolti!

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

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

Attenzione alla differenza tra campi e campi statici!

Se non c'è un costruttore, il compilatore tenta di aggiungere un costruttore senza argomenti che richiama il costruttore senza argomenti della superclasse. Per rispondere correttamente occorre controllare dunque i costruttori della superclasse!

b. Considerate le seguenti dichiarazioni di variabile:

```
Figura f, Rettangolo r, Gamma g, Cerchio c, Delta d, 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):

- SI — `f = new Rettangolo(1, 2)`
- NO — `r = f`
- SI — `c = (Cerchio) f`
- NO — `f = new Figura()` Figura è astratta!
- SI — `g = (Gamma) i`
- NO — `d = (Delta) i`
- NO — `d = g`
- NO — `g = d`
- NO — `c = (Cerchio) i`
- SI — `f = (Gamma) i`

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

```
int x = 0;
try {
  for (Rettangolo r: rettangoli)
    x = x + (int) r.getBase();
  x = (int) rettangoli[x / x].getBase();
} catch (ArithmeticException e) {
  x = x + 33;
} catch (ArrayIndexOutOfBoundsException e) {
  x = x + 15;
} catch (NullPointerException e) {
  x = x + 27;
}
```

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`,

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

(a) l'array riferito da `rettangoli` contiene (nell'ordine indicato) riferimenti a oggetti che rappresentano un rettangolo di base `1` e altezza `10`, un rettangolo di base `2` e altezza `10`, un rettangolo di base `3` e altezza `10`. — 2

(b) `rettangoli` contiene `null`. — 27

(c) l'array riferito da `rettangoli` contiene, come unico elemento, un riferimento a un oggetto che rappresenta un rettangolo di base `10` e altezza `1`. — 25

(d) l'array riferito da `rettangoli` è vuoto. — 33

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
    return x + 3 * f(x - 2);
}
```

f(3)	f(5)
12	41