Informatica Generale — Preparazione al III Compitino

1. Scrivete un (frammento di) programma Java che operi come segue: legge 10 double non negativi da tastiera, e chiede all'utente di inserire un altro valore double α compreso fra 0 e 1. Quindi, il programma stampa i numeri inseriti la cui parte frazionaria sia minore di α. Ad esempio (le parti in grassetto sono state inserite dall'utente):

   Numero 0: **3.12**
   Numero 1: **12.88**
   Numero 2: **0.342**
   Numero 3: **3.42**
   Numero 4: **2**
   ...
   Numero 9: **2.5**
   Soglia: **.5**
   3.12
   0.342
   3.42

   ## Soluzione

   ```
   double a[] = new double[ 10 ];
   for ( int i = 0; i < 10; i++ )
       a[ i ] = in.readDouble( "Numero " + i + ": " );
   double s = in.readDouble( "Soglia: " );
   for ( int i = 0; i < 10; i++ )
       if ( a[ i ] - (int)a[ i ] < s )
           out.println( a[ i ] );
   ```

2. Supponete che esista una classe Triangolo le cui istanze corrispondono a dei triangoli rettangoli; questa classe ha un costruttore pubblico

```
public Triangolo( double x, double y )
```

che costruisce un triangolo con cateti x e y, e un metodo pubblico

```
public double ipotenusa()
```

che fornisce l'ipotenusa del triangolo.

Scrivete un programma che operi come segue:

(a) chieda all'utente di inserire un numero intero, diciamo N;

(b) chieda all'utente di inserire N triangoli;

(c) stampi i soli triangoli aventi ipotenusa maggiore di 2 (supponete che in Triangolo ci sia un opportuno metodo toString()).

Ecco un esempio di esecuzione:

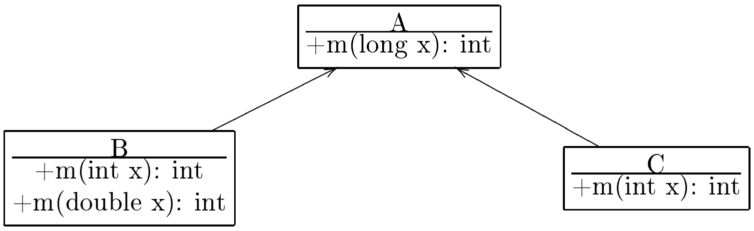
```
Inserisci un intero: 4
Primo cateto: 0.5
Secondo cateto: 0.8
Primo cateto: 4
Secondo cateto: 3.21
...
Secondo cateto: 0.2

Triangolo con cateti 4 e 3.21
...
```

Soluzione

```
int N = in.readInt( "Inserisci un intero: " );
Triangolo t[] = new Triangolo[ N ];
for ( int i = 0; i < N; i++ ) {
  double x = in.readDouble( "Primo cateto: " );
  double y = in.readDouble( "Secondo cateto: " );
  t[ i ] = new Triangolo( x, y );
}
for ( int i = 0; i < N; i++ )
  if ( t[ i ].ipotenusa() > 2 )
     out.println( t[ i ] );
```

3. Considerate la seguente gerarchia di classi (rappresentata mediante un diagramma UML):



e ipotizzate che tutte le classi abbiano un costruttore pubblico senza argomenti. Assumete le seguenti definizioni e inizializzazioni:

```
A a1, a2, a3;
B b;
C c;

a1 = new A(); a2 = new B(); a3 = new C();
b = new B();
c = new C();
```

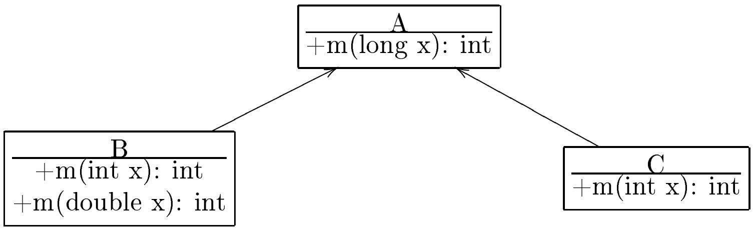
Per ciascuno dei seguenti assegnamenti, dite se l'assegnamento è consentito, oppure se richiede (per poter essere compilato) un operatore di cast nel secondo membro (e in tal caso indicate l'assegnamento corretto), oppure infine se l'assegnamento non è consentito nemmeno con un cast (nel senso che anche con un cast produrrebbe un errore di compilazione o un'eccezione in esecuzione).

- `a2=b`: ....
- `b=a1`: ....
- `b=a2`: ....
- `c=b`: ....
- `a3=b`: ....

Soluzione

- `a2=b`: **Ok**
- `b=a1`: **No**
- `b=a2`: **Richiede cast:** `b=(B)a2`
- `c=b`: **No**
- `a3=b`: **Ok**

4. Facendo riferimento al precedente diagramma UML, e alle variabili definite e inizializzate come sopra, considerate le seguenti invocazioni di metodo. Per ciascuna di esse dovete dire se è consentita, e in caso affermativo dovete indicare il nome della classe che contiene il metodo che verrà effettivamente eseguito:



- `b.m(3.4)`: ....
- `b.m(3)`: ....
- `a1.m(5)`: ....
- `a2.m(3.3)`: ....
- `c.m(5)`: ....
- `a2.m(5)`: ....

Soluzione

- `b.m(3.4)`: **metodo** `m(double x)` **di B**
- `b.m(3)`: **metodo** `m(int x)` **di B**
- `a1.m(5)`: **metodo** `m(long x)` **di A**
- `a2.m(3.3)`: **Non consentita**
- `c.m(5)`: **metodo** `m(int x)` **di C**
- `a2.m(5)`: **metodo** `m(long x)` **di A**

5. Considerate la seguente funzione $f : \{1, 2, \ldots\} \to \{1, 2, \ldots\}$:

$$f(x) = \begin{cases} 0 & \text{se } x = 1 \\ 1 + f(\lfloor x/2 \rfloor) & \text{altrimenti.} \end{cases}$$

Scrivete un metodo statico che realizzi f (ignorate il problema di controllare che l'argomento passato al metodo sia maggiore di 0):

Soluzione

```
public static int f( int x ) {
    if ( x == 1 ) return 0;
    return 1 + f( x / 2 );
}
```

6. Definite una classe, di nome Complesso, le cui istanze rappresentano numeri complessi (della forma $a + ib$). La classe deve avere:

   - un costruttore che prende 2 argomenti (la parte reale e la parte immaginaria);
   - un costruttore che prende 1 argomento (la parte reale) e assume che la parte immaginaria sia 0;
   - due metodi getReale() e getImmaginaria() che restituiscono la parte reale e la parte immaginaria, rispettivamente;
   - un metodo **public double** getModulo() che restituisce il modulo del numero complesso (cioè, $\sqrt{a^2 + b^2}$);
   - un metodo **public** Complesso piu(Complesso c) che restituisce il numero complesso ottenuto sommando `c` a quello su cui è invocato.

Soluzione

```java
class Complesso {
  private double a, b;

  public Complesso( double a, double b ) {
    this.a = a;
    this.b = b;
  }
  public Complesso( double a ) {
    this( a, 0 );
  }

  public double getReale() {
    return a;
  }
  public double getImmaginaria() {
    return b;
  }
  public double getModulo() {
    return Math.sqrt( a*a + b*b );
  }
  public Complesso piu( Complesso c ) {
      Complesso ris = new Complesso( a + c.a, b + c.b );
      return ris;
  }
}
```