Informatica Generale — III Compitino

27 gennaio 2004

Cognome ..... Nome .....
Matricola ..... (Versione A)

1. Scrivete un (frammento di) programma Java che operi come segue: legge 100 interi da tastiera e chiede all'utente di inserire due estremi i e j (che assumiamo essere compresi fra 0 e 99). Quindi, il programma stampa *i numeri inseriti il cui indice è compreso fra i e j* (è inteso che l'indice del primo intero inserito è 0, l'indice del secondo è 1, ..., l'indice dell'ultimo è 99).

   Ecco un esempio di esecuzione (le parti in grassetto sono state inserite dall'utente):

```
Numero 0:  12
Numero 1:  1288
Numero 2:  342
Numero 3:  342
Numero 4:  -23413
...
Numero 99:  23

Primo estremo:  1
Secondo estremo:  3

1288
342
342
```

Soluzione

```
int x[] = new int[ 100 ];
for ( int i = 0; i < 100; i++ )
 x[ i ] = in.readInt( "Numero " + i + ": " );
int a = in.readInt( "Primo estremo: " );
int b = in.readInt( "Primo estremo: " );
for ( int i = a; i <= b; i++ )
 out.println( x[ i ] );
```

2. Supponete che esista una classe Punto i cui oggetti corrispondono a punti nel piano cartesiano; questa classe ha un costruttore pubblico

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

che costruisce un punto di coordinate x e y, e un metodo pubblico

```
public double mod()
```

che fornisce il modulo (cioè, la distanza dall'origine) del punto.

Scrivete un programma che operi come segue:

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

(b) chieda all'utente di inserire un valore double (la soglia), diciamo r;

(c) dichiari un array di N punti e lo riempia con punti inseriti dall'utente (specificandone le coordinate);

(d) stampi i soli punti *il cui modulo è maggiore di* r (potete assumere che la classe Punto disponga di un opportuno metodo toString()).

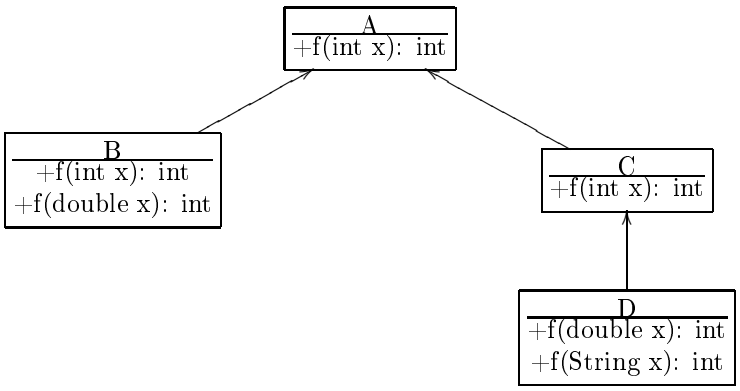
Ecco un esempio di esecuzione (le parti in grassetto sono state inserite dall'utente):

```
Inserisci un intero:  4
Inserisci una soglia:  1

Coordinata X: 0.5
Coordinata Y: 0.8
Coordinata X: 1
Coordinata Y: 1
Coordinata X: 0.7
Coordinata Y: 0.2
Coordinata X: 0.8
Coordinata Y: 1.2

(1.0,1.0)
(0.8,1.2)
```

Soluzione

```
int N = in.readInt( "Inserisci un intero: " );
double r = in.readDouble( "Inserisci una soglia: " );
Punto a[] = new Punto[N];
for ( int i = 0; i < N; i++ ) {
 double x, y;
 x = in.readDouble( "Coordinata X: " );
 y = in.readDouble( "Coordinata Y: " );
 a[i] = new Punto( x, y );
}
for ( int i = 0; i < N; i++ )
 if ( a[i].mod() > r )
  out.println( a[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, a4;
B b;
C c1, c2;
D d;

a1 = new A(); a2 = new B(); a3 = new C(); a4 = new D();
b = new B();
c1 = new C(); c2 = new D();
D d = new D();
```

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 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):

- `a1=b`: **Ok**
- `b=a1`: **Non corretto**
- `a1=d`: **Ok**
- `c1=d`: **Ok**
- `d=c2`: **Richiede cast:** `d=(D)c2`

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.f(3.4)`: `B: f(double)`
   - `b.f(3)`: `B: f(int)`
   - `a3.f(5)`: `C: f(int)`
   - `a4.f("pippo")`: **Non corretto**
   - `d.f("pippo")`: `D: f(String)`

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

$$f(n) = \begin{cases} 0 & \text{se } n = 1 \\ 1 + f(n-1) & \text{altrimenti.} \end{cases}$$

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;
 else return 1 + f( x-1 );
}
```

6. Definite una classe, di nome Scarpa, le cui istanze rappresentano delle scarpe da uomo. Ogni scarpa è caratterizzata da un numero *intero* (la misura), una stringa che corrisponde al modello e una stringa che corrisponde al colore. La classe deve avere:

   - un costruttore con 3 argomenti (modello, misura, colore);
   - un costruttore con 2 argomenti (e assume che il colore sia `"nero"`);
   - tre metodi getMisura(), getModello() e getColore() che restituiscono la misura, il modello e il colore, rispettivamente;
   - un metodo **public boolean** stessaMisura(Scarpa s) che restituisce `true` se la scarpa su cui è invocato ha la stessa misura della scarpa che viene passata come argomento, `false` altrimenti;
   - un metodo **public boolean** isLunga() che restituisce `true` se la scarpa ha misura maggiore di 45, `false` altrimenti.

Soluzione

```
class Scarpa {
 private String modello;
 private int misura;
 private String colore;

 public Scarpa(String modello, int misura, String colore) {
  this.modello = modello;
  this.misura = misura;
  this.colore = colore;
 }
 public Scarpa(String modello, int misura) {
  this(modello,misura,"nero");
 }
 public String getModello() {
  return modello;
 }
 public String getColore() {
  return colore;
 }
 public int getMisura() {
  return misura;
 }
 public boolean stessaMisura( Scarpa x ) {
  return misura == x.misura;
 }
 public boolean isLunga() {
  return misura > 45;
 }
}
```