Informatica Generale — III Compitino

27 gennaio 2004

Cognome .... Nome ....
Matricola .... (Versione B)

1. Scrivete un (frammento di) programma Java che operi come segue: legge 100 interi da tastiera e chiede all'utente di inserire due numeri (estremi) i e j. Quindi, dei 100 numeri inseriti, il programma stampa quelli *il cui valore è compreso fra* i *e* j (estremi inclusi).

   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:  12
Secondo estremo:  350

12
342
342
...
23
```

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 = 0; i <= 100; i++ )
  if ( a <= x[ i ] && x[ i ] <= b )
   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 minore di* r (potete assumere che la classe Punto disponga di un opportuno metodo toString()).

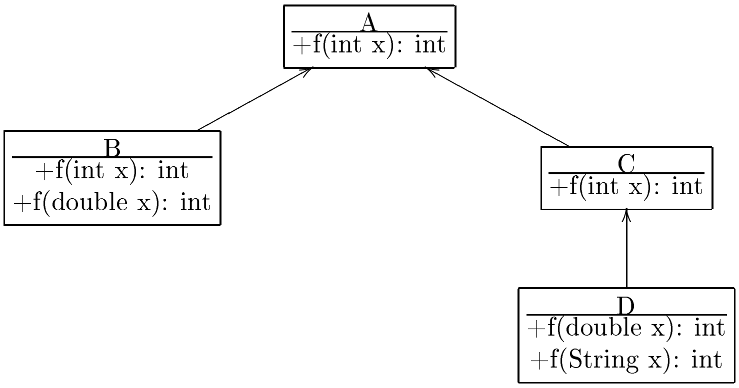
Ecco un esempio di esecuzione:

```
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

(0.5,0.8)
(0.7,0.2)
```

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=a2`: **Richiede cast:** `b=(B)a2`
- `a1=d`: **Ok**
- `c1=d`: **Ok**
- `d=c1`: **No**

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:

   - `a1.f(3.4)`: **Non corretto**
   - `b.f(3)`: `B: f(int)`
   - `a3.f(5)`: `C: f(int)`
   - `d.f(3)`: `C: f(int)`
   - `d.f("pippo")`: `D: f(String)`

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

$$f(n) = \begin{cases} 0 & \text{se } n < 3 \\ n + f(n-2) & \text{altrimenti.} \end{cases}$$

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 < 3 ) return 0;
 else return x + f( x-2 );
}
```

6. Definite una classe, di nome Persona, le cui istanze rappresentano delle persone. Ogni persona è caratterizzata da un numero *intero* (la sua età), una stringa che corrisponde al nome e una stringa che corrisponde al cognome. La classe deve avere:

   - un costruttore con 3 argomenti (età, nome, cognome);
   - un costruttore con 2 argomenti (e assume che l'età sia 18);
   - tre metodi getNome(), getCognome() e getEta() che restituiscono il nome, il cognome e l'età, rispettivamente;
   - un metodo **public boolean** stessaEta(Persona s) che restituisce `true` se la persona su cui è invocato ha la stessa età della persona che viene passata come argomento, `false` altrimenti;
   - un metodo **public boolean** isMaggiorenne() che restituisce `true` se la persona ha almeno 18 anni, `false` altrimenti.

Soluzione

```
class Persona {
 private int eta;
 private String nome;
 private String cognome;

 public Persona(int eta, String nome, String cognome) {
  this.eta = eta;
  this.nome = nome;
  this.cognome = cognome;
 }
 public Persona(String nome, String cognome) {
  this(18,nome,cognome);
 }
 public String getNome() {
  return nome;
 }
 public String getCognome() {
  return cognome;
 }
 public int getEta() {
  return eta;
 }
 public boolean stessaEta(Persona p) {
  return eta==p.eta;
 }
 public boolean isMaggiorenne() {
  return eta>=18;
 }
}
```