Fondamenti di Architetture e Programmazione
III Compitino (Vers. A)

27 gennaio 2006

- Usate il retro dei fogli per scrivere lo svolgimento degli esercizi di programmazione.
- Nei seguenti quesiti, quando vi è richiesto di scrivere un programma, potete limitarvi al *corpo* del metodo main, assumendo se necessario che in e out siano due variabili di classe ConsoleInputManager e ConsoleOutputManager (rispettivamente), già dichiarate e inizializzate.

1. Scrivete un programma che legga una sequenza di esattamente 10 stringhe, e stampi solo quelle che terminano con una cifra; ricordiamo che il metodo statico isDigit (**char** c) della classe Character restituisce true sse il carattere che gli viene passato come argomento è una cifra.

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

   ```
   Stringa 1:  mamma3
   Stringa 2:  pippo
   Stringa 3:  xy12
   ...

   mamma3
   xy12
   ...
   ```

(Svolgimento sul retro)

```java
String[] s = new String[ 10 ];
for ( int i = 0; i < 10; i++ )
 s[ i ] = in.readLine( "Stringa " + ( i + 1 ) + ": " );
for ( int i = 0; i < 10; i++ )
 if ( Character.isDigit( s[ i ].charAt( s[ i ].length() - 1 ) ) )
  out.println( s[ i ] );
```

2. Supponete che esista una classe Paziente i cui oggetti corrispondono a dei pazienti di uno studio di dietologia; questa classe ha un costruttore pubblico

```
public Paziente( String n, double p, double h )
```

che costruisce un paziente con dato nominativo n, peso p (in chilogrammi) e altezza h (in centimetri), e un metodo pubblico

```
public double coeffPeso()
```

che fornisce un coefficiente di peso, calcolato come $p/(h-100)^2$.

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 pazienti e lo riempia con dati inseriti dall'utente (che, per ciascun paziente, deve inserire il nominativo, il peso e l'altezza);

(d) stampi i soli pazienti *il cui coefficiente è maggiore di* r (potete assumere che la classe Paziente disponga di un opportuno metodo toString()), ciascuno accompagnato dal suo coefficiente.

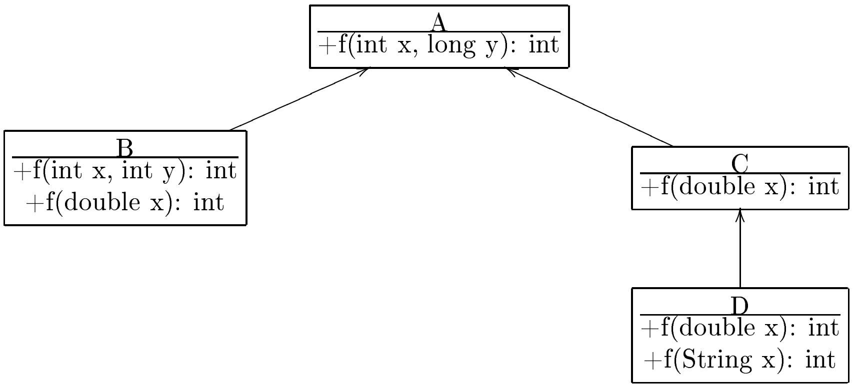
Ecco un esempio di esecuzione (le parti in grassetto sono state inserite dall'utente):

```
Inserisci un intero: 3
Inserisci una soglia: .02

Nominativo: Mario Rossi
Peso kg.: 75.4
Altezza cm.: 172
Nominativo: Giovanni Bianchi
Peso kg.: 77.2
Altezza cm.: 161
Nominativo: Maria Marroni
Peso kg.: 66
Altezza cm.: 141

Giovanni Bianchi .0207471109
Maria Marroni .0392623438
```

(Svolgimento sul retro)

```
int n = in.readInt( "Inserisci un intero: " );
double soglia = in.readDouble( "Inserisci una soglia: " );
Paziente[] a = new Paziente[ n ];
int i;
for ( i = 0; i < n; i++ ) {
 String nome = in.readLine( "Nominativo: " );
 double peso = in.readDouble( "Peso kg.: " );
 double alt = in.readDouble( "Altezza cm.: " );
 a[ i ] = new Paziente( nome, peso, alt );
}

for ( int i = 0; i < 10; i++ )
 if ( a[ i ].coeffPeso() > r )
  out.println( a[ i ] + " " + a[ i ].coeffPeso() );
```

3. Considerate la seguente gerarchia dei tipi (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=a3`: **No**
- `b=c1`: **No**
- `a1=c1`: **Ok**
- `c1=a3`: **c1=(C)a3**
- `d=c1`: **No**
- `c2=d`: **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:

    - `a1.f(3,4)`: **A: f(int,long)**
    - `a2.f(3,4)`: **A: f(int,long)**
    - `b.f(3)`: **B: f(double)**
    - `a3.f(5)`: **No**
    - `a4.f("pippo")`: **No**
    - `d.f("pippo")`: **D: f(String)**
    - `d.f(3,4)`: **A: f(int,long)**
    - `d.f(3.4)`: **D: f(double)**

5. La classe Integer contiene un metodo statico

   ```
   public static String toBinaryString( int x )
   ```

   che dato un intero `x` lo converte in una stringa che ne fornisce la rappresentazione binaria. Avete una variabile intera `i` e volete determinare se la lunghezza della sua rappresentazione binaria è pari, mettendo il risultato (true o false) nella variabile booleana `z`; scrivete l'istruzione di assegnamento a `z`:

   ```
   z=( Integer.toBinaryString(i) % 2 == 0 );
   ```

6. Considerate la seguente funzione definita ricorsivamente sugli interi:

   $$f(x) = \begin{cases} x & \text{se } x < 10 \\ x + f(x-10) & \text{altrimenti} \end{cases}$$

   Scrivete un metodo statico con intestazione

   ```
   public static int f( int x )
   ```

   per il calcolo di f.

   ```
   public static int f( int x ) {
    if ( x < 10 ) return x;
    return x + f( x - 10 );
   }
   ```

7. Definite una classe di nome Paziente i cui oggetti corrispondono a dei pazienti di uno studio di dietologia; questa classe deve avere:

   - un costruttore pubblico

     ```
     public Paziente( String n, double p, double h )
     ```

     che specifica il nominativo, il peso (in kg.) e l'altezza (in cm.);
   - un costruttore pubblico

     ```
     public Paziente( String n, double p )
     ```

     che assume che l'altezza sia pari a 1.72;
   - un metodo

     ```
     public double coeffPeso()
     ```

     che fornisce un coefficiente calcolato come $p/(h-100)^2$;
   - un metodo

     ```
     public boolean stessoPeso( Paziente p )
     ```

     che restituisce true se e solo se il paziente su cui è invocato e p hanno lo stesso peso.

(Svolgimento sulla facciata seguente)

```java
class Paziente {
 private String n;
 private double p;
 private double h;

 public Paziente( String n, double p, double h ) {
  this.n = n;
  this.p = p;
  this.h = h;
 }

 public Paziente( String n, double p ) {
  this( n, p, 172 );
 }

 public double coeffPeso() {
  return p / ( ( h - 100.0 ) * ( h - 100.0 ) );
 }

 public boolean stessoPeso( Paziente p ) {
  return p == p.p;
 }
}
```

8. Definite una classe di nome PazienteAmmalato che estende Paziente e i cui oggetti corrispondono a dei pazienti con una malattia. Questa classe deve avere:

   - un costruttore pubblico

     ```
     public PazienteAmmalato( String n, String m, double p, double h )
     ```

     che specifica il nominativo, la malattia, il peso (in kg.) e l'altezza (in cm.);

   - un metodo

     ```
     public boolean stessaMalattia( PazienteAmmalato p )
     ```

     che restituisce true se e solo se il paziente su cui è invocato e p hanno la stessa malattia.

```
class PazienteAmmalato extends Paziente {
 private String m;

 public PazienteAmmalato( String n, String m, double p, double h ) {
  super( n, p, h );
  this.m = m;
 }

 public boolean stessaMalattia( PazienteAmmalato p ) {
  return m.equals( p.m );
 }
}
```