Programmazione
II Compitino (Vers. A)

16 gennaio 2009

Cognome **Jekyll** Nome **Doctor**
Matricola **112233**

- 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 un numero n, seguito da una sequenza di esattamente n rettangoli; al termine, deve stampare i rettangoli di area maggiore del primo inserito e di perimetro minore dell'ultimo.

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

   ```
   Quanti rettangoli:  6
   Base:  3
   Altezza:  3
   Base:  7
   Altezza:  2
   Base:  7
   Altezza:  7
   Base:  3
   Altezza:  5
   Base:  1
   Altezza:  1
   Base:  5
   Altezza:  6
   base=7.0, altezza=2.0
   base=3.0, altezza=5.0
   ```

(Svolgimento sul retro)

```
 int n = in.readInt( "Quanti rettangoli: " );
 Rettangolo[] r = new Rettangolo[ n ];
 for ( int i = 0; i < n; i++ ) {
  double b = in.readDouble( "Base: " );
  double h = in.readDouble( "Altezza: " );
  r[ i ] = new Rettangolo( b, h );
 }
 for ( int i = 0; i < n; i++ )
  if ( r[i].haAreaMaggiore( r[0] ) && r[n-1].haPerimetroMaggiore( r[i] ) )
   out.println( r[ i ] );
}
```

2. Scrivete un programma che legga un numero n, seguito da una sequenza di esattamente n stringhe; al termine, deve stampare le stringhe che sono prefisso[1] di almeno un'altra stringa dell'elenco.

   ATTENZIONE. Notate che ogni stringa è un prefisso di sé stessa, per definizione. Noi vogliamo stampare solo le stringhe x che sono prefissi *propri* di qualche altra stringa y (cioè, x deve essere prefisso di y, e x e y devono essere *diverse*).

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

```
Quante stringhe:  6
Stringa 1:  canestro
Stringa 2:  vale
Stringa 3:  carnevale
Stringa 4:  genuflesso
Stringa 5:  cane
Stringa 6:  carnevalesco
carnevale
cane
```

(Svolgimento sul retro)

[1] Una stringa x è prefisso di y se e solo se y è ottenuta aggiungendo zero o più caratteri in fondo a x. Nella classe String c'è un metodo startsWith(String x) che restituisce true se e solo se la stringa su cui è invocato ha x come prefisso.

```
int n = in.readInt( "Quante␣stringhe:␣" );
String[] s = new String[ n ];
for ( int i = 0; i < n; i++ )
 s[ i ] = in.readLine( "Stringa␣" + ( i + 1 ) + ":␣" );
for ( int i = 0; i < n; i++ ) {
 int j;
 for ( j = 0; j < n; j++ )
  if ( s[ j ].startsWith( s[ i ] ) && !s[ j ].equals( s[ i ] ) )
   break;
 if ( j < n )
  out.println( s[ i ] );
}
```

3. Considerate la classe Telefonata i cui oggetti corrispondono a delle *telefonate*; questa classe ha un costruttore pubblico

   - public Telefonata(String chiamante, String chiamato, int durataSecondi, double costoAlSecondo)

   che costruisce una telefonata effettuata dal numero chiamante al numero chiamato, avente una durata di un numero specificato di secondi e con un dato costo al secondo (in euro). La classe possiede, fra gli altri, i metodi

   - public String getChiamante()
   - public String getChiamato()
   - public double costoTotale()
   - public boolean durataPiuDi(Telefonata t)

   che, rispettivamente, forniscono il numero chiamante, il numero chiamato, il costo totale della chiamata, e dicono se la chiamata è durata di più di una certa altra telefonata. La classe ha anche un metodo toString().

   Scrivete un *metodo statico* con la seguente intestazione:

   - public static Telefonata piuCostosa(Telefonata t[], String chiamante)

   che cerca nell'array t la telefonata più costosa[2] fra quelle effettuate dal numero chiamante specificato, e la restituisca. Se l'array non contiene alcuna telefonata effettuata dal numero specificato, il metodo deve restituire **null**.

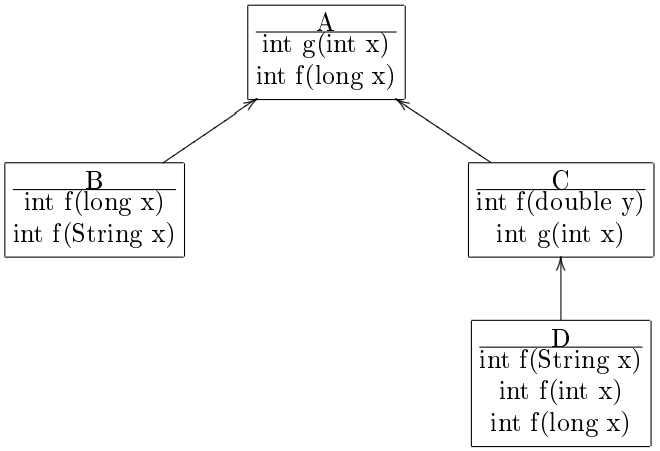
   ATTENZIONE! Il metodo non deve effettuare nessun input, né produrre alcun output.

(Svolgimento in questa pagina)

[2] Nel caso ci siano più telefonate con lo stesso costo massimo, il metodo può restituire una qualunque di esse.

```
public static Telefonata piuCostosa( Telefonata t[], String chiamante ) {
 Telefonata cand = null;
 double costoMax = -1;
 for ( int i = 0; i < t.length; i++ )
  if (t[i].getChiamante().equals(chiamante) && t[i].costoTotale()>costoMax)
   cand = t[ i ];
 return cand;
}
```

4. 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 = 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):

- a4=d: **Ok**
- c1=a3: **Non compila**
- a1=b: **Ok**
- a2=b: **Ok**
- b=a2: **Non compila**
- c2=d: **Ok**
- c1=(C)a2: **Eccezione in esecuzione**

- c1=(C)a3: **Ok**
- c2=b: **Non compila**

5. 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:

   - d.f(3): D: f(int)
   - d.f(4.2): C: f(double)
   - a1.f(3): A: f(long)
   - a3.g(5): C: g(int)
   - b.f("pippo"): B: f(String)
   - b.f(3): B: f(long)
   - d.f(3.4): C: f(double)
   - a4.f(4L): D: f(long)

6. Considerate la seguente funzione definita ricorsivamente sugli interi:

$$f(x) = \begin{cases} f(x-1) * f(x-2) & \text{se } x \text{ è pari e } x > 1 \\ f(x-1) * f(x-3) & \text{se } x \text{ è dispari e } x > 1 \\ x & \text{altrimenti} \end{cases}$$

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 % 2 == 0 && x > 1 )
  return f( x - 1 ) * f( x - 2 );
 else if ( x % 2 == 1 && x > 1 )
  return f( x - 1 ) * f( x - 3 );
 else return x;
}
```

7. Scrivete la classe Telefonata con i seguenti metodi e costruttori:

   - **public** Telefonata(String chiamante, String chiamato, **int** durataSecondi, **double** costoAlSecondo): costruisce una telefonata fra i numeri specificati, con la durata indicata e il costo al secondo indicato; questo costruttore deve sollevare una IllegalArgumentException se durataSecondi è minore o uguale a zero;
   - **public** String getChiamante(): restituisce il numero del chiamante;
   - **public** String getChiamato(): restituisce il numero chiamato;
   - **public double** costoTotale(): restituisce il costo totale;
   - **public boolean** durataPiuDi(Telefonata t): restituisce true se e solo se la telefonata su cui è invocato dura di più di quella passata come argomento;
   - **public** String toString(): restituisce una stringa descrittiva della chiamata opportuna.

(Svolgimento sulla facciata seguente)

```
class Telefonata {
 private String chiamante;
 private String chiamato;
 private int durataSecondi;
 private double costoAlSecondo;

 public Telefonata( String chiamante, String chiamato, int durataSecondi,
   double costoAlSecondo ) throws IllegalArgumentException {
  if ( durataSecondi <= 0 )
   throw new IllegalArgumentException();
  this.chiamante = chiamante;
  this.chiamato = chiamato;
  this.durataSecondi = durataSecondi;
  this.costoAlSecondo = costoAlSecondo;
 }

 public String getChiamante() {
  return chiamante;
 }

 public String getChiamato() {
  return chiamato;
 }

 public double costoTotale() {
  return costoAlSecondo * durataSecondi;
 }

 public boolean duraPiuDi( Telefonata t ) {
  return durataSecondi > t.durataSecondi;
 }

 public String toString() {
  return chiamante + "␣->␣" + chiamato + "␣(" + durataSecondi
   + "s.,␣" + costoAlSecondo + "␣EUR/s.)";
 }
}
```

8. Definite una classe di nome TelefonataAdd che estende Telefonata ma in cui viene anche applicato un addebito alla risposta. Questa classe deve avere un costruttore

   - public TelefonataAdd(String chiamante, String chiamato, int durataSecondi, double costoAlSecondo, double addebito)

   che costruisce una telefonata fra i numeri specificati, con la durata indicata e il costo al secondo indicato, e con l'indicato addebito fisso alla risposta; anche questo costruttore deve sollevare una IllegalArgumentException se durataSecondi è minore o uguale a zero. Inoltre:

   - il metodo ereditato costoTotale() deve essere opportunamente modificato;
   - deve essere fornito un nuovo metodo **public double** add() che fornisce l'addebito applicato alla risposta.

```
class TelefonataAdd extends Telefonata {
 private double add;

 public TelefonataAdd( String chiamante, String chiamato, int durataSecondi,
   double costoAlSecondo, double addebito )
   throws IllegalArgumentException {
  super( chiamante, chiamato, durataSecondi, costoAlSecondo );
  this.add = addebito;
 }

 public double add() {
  return add;
 }

 public double costoTotale() {
  return add + super.costoTotale();
 }

 public String toString() {
  return super.toString() + " + addebito iniziale " + add + " EUR";
 }

}
```