Informatica Generale — Preparazione al II Compitino

26 gennaio 2005

Nota. Ove necessario, nel seguito, assumeremo che `tastiera` e `video` siano due variabili di classe `ConsoleInputManager` e `ConsoleOutputManager` (rispettivamente), già dichiarate e inizializzate.

Rispondete alle seguenti domande.

1. Scrivete un (frammento di) programma Java che operi come segue:

   - legga un intero n da tastiera;
   - legga n stringhe da tastiera, memorizzandole in un array a;
   - stampi le stringhe che contengono la lettera 'a'.

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

   ```
   Quante stringhe?  6

   Stringa 0:  senza
   Stringa 1:  dubbio
   Stringa 2:  il
   Stringa 3:  docente
   Stringa 4:  è
   Stringa 3:  pazzo

   senza
   pazzo
   ```

(Svolgimento sul retro)

2. Considerate la 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 ascissa `x` e ordinata `y`. La classe possiede i metodi

   - public double distanza()
   - public double getX()
   - public double getY()

   che forniscono, rispettivamente, distanza dall'origine, ascissa ed ordinata.

   Scrivete un programma che operi come segue:

   (a) definisca un array di punti di dimensione inserita dall'utente;

   (b) lo riempia con punti inseriti dall'utente;

   (c) stampi i soli punti con entrambe le coordinate positive che distano più di 30 dall'origine (la classe `Punto` dispone di un opportuno metodo `toString()`).

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

   Quanti punti: **5**

   Ascissa: **4**
   Ordinata: **3**

   Ascissa: **30**
   Ordinata: **20**

   Ascissa: **40**
   Ordinata: **-10**

   Ascissa: **50**
   Ordinata: **0**

   Ascissa: **27**
   Ordinata: **28**

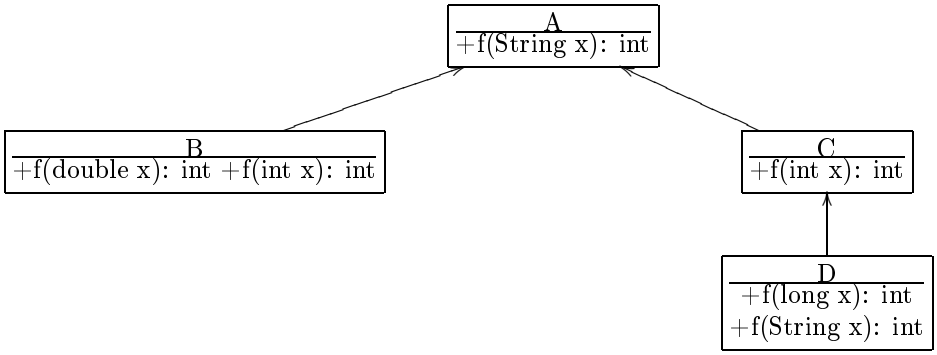
   Ascissa = 30; Ordinata = 20
   Ascissa = 27; Ordinata = 28

(Svolgimento sul retro)

3. Una retta sul piano cartesiano si può rappresentare usando l'equazione $y = mx + q$, mediante m e q. Completatene il codice riempiendo gli spazi.

```
class Retta {
  private double m, q;

  public Retta(double m, double q) {
    [                                    ]
    [                                    ]
    [                                    ]
    [                                    ]
    [                                    ]
  }

  // Restituisce true sse la retta passa per (x,y)
  public boolean passaPer(double x, double y) {
    [                                    ]
    [                                    ]
    [                                    ]
    [                                    ]
    [                                    ]
  }

  // Restituisce il coefficiente angolare
  public double getCoeffAng() {
    [                                    ]
    [                                    ]
    [                                    ]
    [                                    ]
    [                                    ]
  }

  // Restituisce true sse questa retta è parallela a r
  public boolean isParallela(Retta r) {
    [                                    ]
    [                                    ]
    [                                    ]
    [                                    ]
    [                                    ]
  }
}
```

4. 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 aa = new A(), ab = new B(), ac = new C(), ad = new D();
B bb = new B();
C cc = new C(), cd = new D();
D dd = 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):

- `aa=ab`: .....
- `cc=ad`: .....
- `ad=cc`: .....
- `bb=aa`: .....
- `ab=bb`: .....

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:

- `bb.f(3.4)`: .....
- `ab.f(3)`: .....

- `ad.f(5)`: .....
- `ad.f("pippo")`: .....
- `dd.f("pippo")`: .....
- `cd.f(2)`: .....
- `dd.f(3/4)`: .....

6. Scrivete una classe di nome `Telefonata` le cui istanze rappresentano telefonate effettuate da abbonati di una compagnia telefonica. Assumete di avere già scritto delle classi di nome `Abbonato` (le cui istanze rappresentano abbonati) e `Orario` (le cui istanze rappresentano orari). Ogni telefonata è caratterizzata da due abbonati (colui che fa la telefonata e colui che la riceve), un orario (l'orario di inizio della telefonata) e un numero interi di minuti (la durata).

   La classe deve contenere i seguenti metodi e costruttori:

   - `Telefonata(Abbonato chiamante, Abbonato chiamato, Orario inizio, int durata)`: crea una nuova telefonata
   - `Telefonata(Abbonato chiamante, Abbonato chiamato, Orario inizio)`: crea una telefonata, assumendo che la durata sia di 1 minuto
   - `int getDurata()`: restituisce la durata in minuti
   - `Orario inizio()`: restituisce l'orario di inizio
   - `Orario fine()`: restituisce l'orario di fine della telefonata; potete assumere che la classe orario abbia un metodo con segnatura `Orario dopo(int minuti)` che restituisce l'orario ottenuto aggiungendo all'orario target il numero di minuti indicato;
   - `String toString()`: restituisce una stringa descrittiva.

(Svolgimento sul retro)