Corso di laurea in Informatica AA 2007/08 Programmazione, M.A. Alberti

I Compitino - 5 novembre 2007

Cognome **Nome**

Matricola

---

1 (12 punti)

La classe Banco simula la gestione di bancarelle in un mercato. Ogni oggetto della classe ha una struttura con campi che definiscono il nome, l'entità del magazzino e i contanti di cassa. Ogni oggetto della classe risponde ad alcune funzionalità: come vendere e acquistare prodotti ad un dato prezzo. Ognuna di queste azioni modifica ovviamente il magazzino e la cassa. Quando si vende c'è un sovrapprezzo percentuale stabilito da una costante, che costituisce il ricavo. Quando la richiesta di acquistare prodotti è in quantità superiore al prodotto conservato in magazzino allora si vende tutto il magazzino con uno sconto e il sovrapprezzo è minore.

```
public class Banco {
   final double RICARICO = 0.2;
   final double RICARICO_SCT=0.1;
   private static int bancarelle;
   private int magazzino;
   private double cassa;
   private String nome;

   //avvio standard con finanziamento di base
   public Banco(String n) {
      nome = n; magazzino = 10;
      cassa = 100.0;
      bancarelle++;
   }

   //avvio specifico con finanziamento ad hoc
   public Banco(String n,
                int p, double c) {
      nome = n;
      magazzino = p; cassa = c;
      bancarelle++;
   }

   public String toString() {
      return "banco " + nome +
             " in cassa " + cassa +
             ", in magazzino " +
                       magazzino;
   }
```

```
public String acquista
        (int prodotto, double costo) {
   magazzino += prodotto;
   cassa -= prodotto*costo;
   return "banco " + nome +
          " acquista " + prodotto;
}

public String vende
        (int prodotto, double costo) {
   double ricavo;
   if (magazzino>prodotto) {
      magazzino -= prodotto;
      ricavo =
          costo*prodotto*RICARICO;
      cassa += costo*prodotto +
          ricavo;
      return "banco " + nome +
             " ricava " + ricavo;
   }
   else {
      int residuo = magazzino;
      magazzino = 0;
      ricavo = costo * residuo *
                      RICARICO_SCT;
      cassa += costo * residuo +
                           ricavo;
      return "banco " + nome +
             " ricava " + ricavo;
    }
}

}
```

Immaginando di voler effettuare un test della classe `Banco` e di voler istanziare oggetti ad esempio in un metodo main della classe, scrivere gli spezzoni di codice richiesti:

1. Dichiarare due oggetti della classe indicati con gli identificatori `frutta` e `verdura`

   `Banco frutta, verdura;`

2. Istanziare i due oggetti la bancarella `frutta` sarà creata con l'allestimento standard mentre la bancarella `verdura` avrà una dotazione iniziale di 100 (si sottintende chili di prodotto) e una cassa iniziale vuota (avendo speso tutto per l'acquisto del magazzino)

```
frutta = new Banco("frutta");
verdura = new Banco ("verdura", 100, 0);
```

3. Indicare l'istruzione con cui s'intende che la bancarella `frutta` acquisisce altri 30 (chili) di prodotti a €2.0 al chilo

```
frutta.acquista(30, 2.0)
```

4. Indicare l'istruzione con cui s'intende che la bancarella `frutta` vende 15 (chili) di prodotti a €2.0 al chilo

```
frutta.vende(15, 2.0)
```

5. Indicare l'istruzione con cui s'intende che la bancarella `verdura` vende 60 (chili) di prodotti a €1.0 al chilo

```
verdura.vende(60, 1.0)
```

6. Indicare l'istruzione con cui s'intende che la bancarella `verdura` vende 50 (chili) di prodotti a €1.0 al chilo

```
verdura.vende(50, 1.0)
```

7. Calcolare il ricavo di questa seconda vendita:

```
4.0
```

8. Dire se è lecito accedere al campo cassa per leggere il contenuto mediante l'espressione `frutta.cassa:` **NO** Perché: **il campo `cassa` è dichiarato `private` e quindi è inaccessibile al di fuori della classe `Banco`. Per leggerne il valore occorre definire nella classe stessa un metodo, dichiarato `public`, che vi acceda e ne riporti il valore.**

9. Scrivere l'output della istruzione: `frutta.toString()`

```
banco frutta in cassa 76.0, in magazzino 25
```

10. Scrivere l'output della istruzione: `verdura.toString()`

```
banco verdura in cassa 116.0, in magazzino 0
```

11. Valutare il valore del campo `bancarelle:` **2**

---

2 (2 punti)

Scrivere un metodo `getCassa()` per la classe `Banco`, che riporti il valore dell campo `cassa`, indicandone chiaramente l'intestazione completa e il corpo del metodo.

```
public double getCassa() {
    return cassa;
}
```

---

3 (.5 punto)

Dire qual'è la caratteristica di Java che consente di definire due costruttori diversi ad esempio per la classe `Banco`: **overloading** o **sovraccaricamento**

Dire come si distinguono i due costruttori: **dal numero, tipo e ordine dei parametri**.

---

4 (.5 punto)

Dire qual'è la caratteristica di Java che consente di specializzare un metodo, come il metodo `toString()`, per una data classe: **overridding** o **sovrascrittura**

---

5 (3 punti)

Assumendo la dichiarazione:

```
Random rand = new Random();
```

Indicare il range dei valori delle seguenti dichiarazioni:

```
rand.nextInt() % 10;          [-9, 9]
(int) (Math.random() * 5);    [0, 4]
```

Inoltre scrivere un'istruzione per produrre valori pseudo-casuali nell'intervallo:

`[-1, 5]` usando l'oggetto `rand` **`rand.nextInt() % 4 + 2`**

o **`Math.abs(rand.nextInt() % 7) – 1`**

o **`rand.nextInt(7) - 1`**

`[6, 12]` usando il metodo `random()` della classe `Math`

`(int) Math.random() * 7 + 6`

---

6 (2 punti)

Esprimere in linguaggio Java la seguente condizione, usando gli operatori di relazione e quelli logici:
il numero **`n`** deve essere **maggiore** di **`3`** **ma non** di **`8`**

`n > 3 && !(n > 8)` o anche **`3 < n && n <= 8`**

Esprimere in linguaggio Java la negazione della condizione precedente senza introdurre l'operatore di negazione (applicate la legge di De Morgan).

`!(3 < n && n <= 8)` equivale a **`n <= 3 || n > 8`**

---

7 (3 punti)

Date le stringhe:

```
String riga=new String("Sempre caro mi fu quest'ermo colle");
String nuova;
```

calcolare il valore delle espressioni:

```
riga.length()                          34
riga.substring(7, 18).length()         11
riga.substring(7, 18).toUpperCase()    CARO MI FU
nuova = riga.substring(0, 7) + riga.substring(7).replace('m', 't')
                                       Sempre caro ti fu quest'erto colle
nuova.substring(7, 18).replace('c', 'C')
                                          Caro ti fu
Riga                                   Sempre caro mi fu quest'ermo colle
```

---

8 (2 punti)

Indicare l'ordine di valutazione degli operatori nelle seguenti espressioni che assumiamo corrette, scrivendo sotto al simbolo dell'operatore (considerate anche l'operatore `dot`) il numero corrispondente all'ordine

```
x = a = b-- * ++a;
  5   4   1 3  2

x = parola.endsWith("ino") && !(parola.length() == 3)
  8       2         1       7 6       4       3  5
```

---

9 (4 punti)

Date le variabili: `int a = -3, b = 4;` eseguire i due blocchi di istruzioni separatamente:

```
a  = a + b;
b += a + --b;
```

```
a  = b-- * ++a;
b += a - b--;
```

E calcolare il valore di a, b:

```
a: 1
b: 8
```

```
a: -8
b: -8
```

---

10 (3 punti)

Data l'espressione booleana `totale < MAX && !finito` compilarne la tabella di verità.

totale < MAX	finito	!finito	totale < MAX && !finito
F	**F**	**V**	**F**
F	**V**	**F**	**F**
V	**F**	**V**	**V**
V	**V**	**F**	**F**

Sapendo che `MAX=10` specificare una possibile coppia di valori delle variabili `totale` e `finito` per rendere vera la condizione:

totale = 5 e **finto = false**