Corso di laurea in Informatica AA 2005/06 Programmazione, M.A. Alberti

I Compitino - 9 novembre 2005

Cognome **Nome**

Matricola

---

1 (8 punti)

Data la classe **Vendemmia**:

```
public class Vendemmia {
   public int carico;
   public float mosto;
   public int pigiatura;
   private static int caricoTotale= 0;
   private static float mostoTotale;
   private static int numeroPigiature= 0;

/* Costruisce un oggetto per la vendemmia
di un agricoltore */
public Vendemmia() {
   carico = 0;
   pigiatura = 0;
   mosto = 0;
}

/* Costruisce un oggetto per la vendemmia
di un agricoltore con carico iniziale*/
public Vendemmia(int caricoUva) {
   carico = caricoUva;
   caricoTotale += caricoUva;
   pigiatura = 0;
   mosto = 0;
}

/* metodo per aggiungere un carico di uva
da pigiare */
public int carica(int caricoUva) {
   carico = caricoUva;
   caricoTotale += carico;
   return carico;
}
```

```
/* metodo per effettuare la
pigiatura.
Si tiene conto della perdita del 10%
in raspi e il parametro rappresenta
il rendimento dell'uva raccolta */
public void pigia(int rendimento) {
   pigiatura = (int) (carico * 0.9);
   mosto= pigiatura * rendimento/100;
   mostoTotale += mosto;
   numeropigiature++;
}

/* metodo per produrre una
descrizione opportuna della vendemmia
di ciascun agricoltore */
public String toString() {
   return
      "vendemmia: " + carico +
      ", mosto: " + mosto;
}

/* metodo per leggere la quantità
totale di uva vinificata */
public static float getMosto() {
   return mostoTotale;
}

/* metodo statico per leggere il
numero di pigiature fatte*/
public static int getPigiature(){
   return numeroPigiature;
}

/* metodo per leggere la raccolta
totale d'uva */
public static int getCarico() {
   return caricoTotale;
}
```

Istanziare un oggetto della classe **Vendemmia** di nome **miaVendemmia** per predisporre la raccolta futura:

```
Vendemmia miaVendemmia = new Vendemmia();
```

Istanziare un oggetto **tuaVendemmia** con un carico di uva di 200 kg:

```
Vendemmia tuaVendemmia = new Vendemmia(200);
```

Raccolti 100 kg di uva caricateli per la pigiatura della mia vendemmia, invocando il metodo:

```
miaVendemmia.carica(100);
```

Procedere con la pigiatura della mia vendemmia, tenendo conto che il rendimento della mia uva e' dell'70%:

```
miaVendemmia.pigia(70);
```

Ora e' il turno della pigiatura della tua vendemmia che rende l'80%:

```
tuaVendemmia.pigia(80);
```

In un altro campo raccolgo altri 50 kg di uva da vinificare che rende il 90%. Quindi invoco i metodi:

```
miaVendemmia.carica(50);

miaVendemmia.pigia(90);
```

Stampo il rapporto della mia e della tua vendemmia ora, ottenendo:

```
System.out.println(miaVendemmia);      vendemmia: 50, mosto: 40.0

System.out.println(tuaVendemmia);      vendemmia: 200, mosto: 144.0
```

Invocare i metodi seguenti e calcolare il valore che riportano:

```
Vendemmia.getPigiature();              3

Vendemmia.getMosto();                  247.0

Vendemmia.getCarico();                 350.0
```

---

2 (3 punti)

Dato l'oggetto

```
Vendemmia vendemmia2005 = new Vendemmia(300);
```

Dire se sono corrette le istruzioni:

```
int r = vendemmia2005.carica(100);        SI    NO

vendemmia2005.pigia(80);                  SI    NO
```

Valutate le seguenti espressioni:

```
r                                         100

vendemmia2005.carico;                     100

Vendemmia.getMosto();                     72 (319 se pensate che le istruzioni siano
```

sequenziali rispetto a quelle dell'esercizio precedente)

---

3 (1 punto)

Dire qual è la caratteristica di Java che consente di definire due costruttori **Vendemmia**:

overloading o sovraccaricamento

Dire come si distinguono i due costruttori:

dal numero, tipo e ordine dei parametri

---

4 (3 punti)

Assumendo la dichiarazione:

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

Indicare il range dei valori delle seguenti dichiarazioni:

```
rand.nextInt() % 9;                    [-8, 8]

(int) (Math.random() * 4);             [0, 3]
```

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

[-2, 4] usando l'oggetto `rand` `Math.abs(rand.nextInt()% 7) - 2`

[4, 12] usando il metodo `random()` della classe `Math`: `(int) (Math.random()*9) + 4;`

5 (3 punti)

Esprimere in linguaggio Java la seguente condizione, usando gli operatori di relazione e quelli logici:

il numero `n` deve essere maggiore di `10` ma non di `30`

```
n>10 && n<=30
```

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

```
n<=10 || n>30
```

---

6 (2 punti)

Data la stringa:

```
String riga = new String("alta, solenne, vestita di nero");
```

calcolare l'output delle istruzioni:

```
riga.length();      30
riga.substring(4, 9).length();        5
riga.substring(3, 10).toUpperCase(); A, SOLE
riga.replace('e', 'a').substring(3, 14); a, solanna,
String nuova = riga.substring(8, 13).replace('n', 'm'); lemme
System.out.println (nuova); lemme
System.out.println (riga);  alta, solenne, vestita di nero
```

---

7 (2 punti)

Assumendo le dichiarazioni:

```
int x;
String parola = new String("Ei fu");
```

Indicare l'ordine di valutazione degli operatori nelle seguenti espressioni, mettendo il numero corrispondente sotto al simbolo dell'operatore (considerate anche l'operatore `dot`).

```
x = 2 * parola.length() + 4
  5   3       1      2  4
x = parola.length() + 2 * 3
  5       1      2  4  3
x = 2 * parola.toUpperCase().length()
  6   5       1           2 3       4
```

---

8 (3 punti)

Date le variabili:

```
int a = 1, b = 5;
```

dopo aver eseguito separatamente i due blocchi di istruzioni:

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

calcolare il valore di :

a: 6	a: 10
b: 15	b: 10

9 (5 punti)
Data l'espressione booleana:

```
numero < 10 || !finito
```

compilarne la tabella di verità.

numero < 10	finito	!finito	numero < 10 || !finito
F	F	V	V
F	V	F	F
V	F	V	V
V	V	F	V

Specificare una possibile coppia di valori delle variabili **numero** e **finito** per rendere falsa la condizione:
ad esempio **numero** = 12 e **finito** = **true**

Se la condizione fosse usata in un ciclo (si entra e si resta nel ciclo se la condizione è vera):

```
Finché
        <condizione>
Fai
        <istruzioni; numero++; finito = !finito>
```

nell'ipotesi che, prima d'iniziare il ciclo, `numero` sia `8` e `finito` sia `false`, dire quante volte viene eseguito il blocco d'istruzioni :

```
<istruzioni; numero++; finito = !finito>
```

Il ciclo viene eseguito 3 volte con le seguenti tre coppie di valori per le variabili **numero** e **finito**

```
8 false
9 true
10 false
```