Corso di laurea in Informatica AA 2006/07 Programmazione, M.A. Alberti

I Compitino - 10 novembre 2006

Cognome **Nome**

Matricola

---

1 (12 punti)

Data la classe **`Automobile`**:

```
public class Automobile {
 public static int autoCircolanti;
 private double serbatoio;
 private String colore;
 private String tipo;
 /* le costanti indicano il consumo in Km al litro di carburante
 private static final int ALTO=5;
 private static final int MEDIO=10;
 private static final int BASSO=15;

 public Automobile() {
  serbatoio = 0.0;
  colore = "";
  tipo = "";
  autoCircolanti++;
 }

 public Automobile(double dato) {
  serbatoio = dato;
  colore = "";
  tipo = "";
  autoCircolanti++;
 }

 public void setColore(String c) {
  colore = c;
 }

 public void setTipo( … t) {

  …

 }

 public String getColore() {

  …

 }
```

```
public      …       getTipo() {
 …


}

public void rifornire(double quanto) {
  this.serbatoio += quanto;
}

public double livello() {
  return serbatoio;
}

public double consumare(double quanto){
 return serbatoio -= quanto;
}

public double consumare(char t, int k){
 double quanto;

 switch ( … ) {

  case …

  case …

  case …

  default: …

 }/* fine switch

 return …


}/* fine consumare

public String toString() {
 return "Auto " + tipo + "; colore " +
         colore + "; serbatoio " +
         this.livello();
}
}/* fine classe Automobile
```

Le richieste 1, 10 e 15 sono relative al completamento o modifica del codice della classe **`Automobile`** le altre sono da intendersi come richiesta di codice di una diversa classe. Ad esempio come istruzioni del metodo **`main`** di una classe **`TestAutomobile`**.

1. Completare il codice dei metodi **`setTipo()`** **`getColore()`** e **`getTipo()`**
2. Istanziare un oggetto della classe **`Automobile`** di nome **`autoRossa`** disponendo un serbatoio di 60 litri:

3. Inizializzare il campo **`colore`** della variabile **`autoRossa`** al valore "**`rosso`**":

4. Inizializzare il campo **`tipo`** della variabile **`autoRossa`** al valore "**`Fiat Marea`**":

5. Istanziare un oggetto della classe **`Automobile`** di nome **`autoBlu`** con il costruttore di default:

6. Inizializzare il campo **`serbatoio`** della variabile **`autoBlu`** rifornendo 30 litri di carburante:

7. Inviare alla variabile **`autoRossa`** il messaggio **`consumare`** 40.5 litri di carburante:

8. Invocare il metodo opportuno per controllare il livello del carburante nel serbatoio di entrambe le variabili **`autoRossa`** e **`autoBlu`** e calcolare il valore di ritorno allo stato attuale, cioè dopo aver eseguito le istruzioni precedenti

9. Ricalcolare i livelli del carburante di entrambe le variabili **`autoRossa`** e **`autoBlu`** dopo aver eseguito:
```
autoBlu.rifornire(autoRossa.livello());
autoRossa.rifornire(autoBlu.livello());
```

10. Sovraccaricare il metodo **`consumare()`** della classe **`Automobile`** con un metodo che accetti in input 2 valori, di tipo char e di tipo int, che indicano rispettivamente il livello di consumo indicativo e il numero di Km percorsi. Il livello di consumo viene indicato convenzionalmente con il carattere `'a'` per indicare un consumo alto, `'m'` uno medio e `'b'` uno basso. Si realizzi il codice usando un'istruzione `switch` che seleziona in funzione del carattere di input l'istruzione da eseguire per ottenere il consumo effettivo in litri di carburante facendo uso delle costanti **`ALTO`**, **`MEDIO`** e **`BASSO`** definite nella classe. Il caso di default può essere definito come il caso di consumo medio. Si segua lo schema dato nel codice.

11. Scrivere le corrette dichiarazioni per l'apertura dei canali di input e output usando la libreria `prog.io`

12. Scrivere le istruzione che inviano un opportuno prompt e acquisiscono i valori del livello di consumo e i km percorsi, con cui inizializzare variabili **`catConsumo`** e **`km`** da dichiararsi in modo opportuno

13. Invocare il metodo **`consumare(catConsumo, km)`** appena definito, per entrambe le variabili **`autoRossa`** e **`autoBlu`** usando i valori appena raccolti per la variabile **`autoRossa`** e altri da richiedere per la variabile **`autoBlu`**

14. Calcolare ora il nuovo livello di carburante per le variabili **`autoRossa`** e **`autoBlu`** nell'ipotesi che l'utente alle istruzioni precedenti abbia digitato **`'a'`** e **`550`** per la variabile **`autoRossa`**, **`'b'`** e **`100`** per la variabile **`autoBlu`**

15. Sovraccaricare il costruttore della classe **`Automobile`** con un costruttore che riceva in input tre valori appropriati per inizializzare i tre campi **`serbatoio colore tipo`**

16. Istanziare quindi due nuove variabili: **autoMetallizzata** con i valori iniziali 43.7, `"Toyota Corolla", "grigio met"` e **miaAuto** con i valori `70.4, "Fiat Tipo", ""`

17. Interrogare la variabile **autoMetallizzata** per ottenerne il tipo

18. Scrivere lo stato attuale dopo aver eseguito (ricordarsi di completare l'ultima istruzione)

```
out.println(autoMetallizzata);
out.println(miaAuto);
out.println("numero macchine circolanti:          …                    )
```

---

2 (4 punti)

Scrivere un'istruzione condizionale che controlli che la richiesta **consumare(catConsumo, km)** inviata all'oggetto **autoRossa** possa essere soddisfatta e in caso contrario invii a stampa su video un segnale "macchina ferma: mancanza di carburante"

Riscrivere il metodo **livello()** utilizzando l'operatore condizionale ternario (**? : :**) e assegni il valore **0** al serbatoio se questo risulta essere negativo (perché ad esempio una precedente richiesta di consumo era troppo elevata per l'effettivo contenuto del serbatoio)

---

3 (.5 punto)

Dire qual è la caratteristica di Java che consente di definire due costruttori **Automobile**:
Dire come si distinguono i due costruttori:

---

4 (3 punti)

Assumendo la dichiarazione:

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

Indicare il range dei valori delle seguenti dichiarazioni:

```
rand.nextInt() % 10;
```

```
(int) (Math.random() * 5);
```

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

`[-1, 5]` usando l'oggetto `rand`

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

---

5 (1.5 punti)

Esprimere in linguaggio Java la seguente condizione, usando gli operatori di relazione e quelli logici:
il numero **n** deve essere **maggiore** di **5 ma non** di **10**

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

6 (2 punti)

Data la stringa:

```
String riga=new String("I cipressi che a Bolgheri alti e schietti");
```

calcolare l'output delle istruzioni:

```
riga.length();
riga.substring(4, 10).length();
riga.substring(4, 10).toUpperCase();
riga.replace('e', 'a').substring(4, 10);
String nuova = riga.substring(15, 27).replace('a', 'A');
System.out.println (nuova);

System.out.println (riga);
```

7 (1 punti)

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

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

x = numero.isDigit() && !finito
```

8 (2 punti)

Date le variabili: `int a = -2, b = 5;` eseguire i due blocchi di istruzioni separatamente:

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

E calcolare il valore di a, b:

`a:`	`a:`
`b:`	`b:`

9 (4 punti)

Data l'espressione booleana `numero > 5 || !finito` compilarne la tabella di verità.

`numero > 5`	`finito`	`!finito`	`numero > 5 || !finito`

Specificare una possibile coppia di valori delle variabili `numero` e `finito` per rendere falsa la condizione:

Se l'espressione fosse usata come condizione in un ciclo `while` (si entra e si resta nel ciclo se la condizione è vera) dire quante volte viene eseguito il blocco d'istruzioni:

```
int numero = 7;
boolean finito = false;
while (<condizione>) {
       <istruzioni>; --numero; finito = !finito}
```