Corso di laurea in Comunicazione digitale Fondamenti di architettura e programmazione

AA 2008/09 Prof. M.A. Alberti

II Compitino - 17 dicembre 2008

Cognome **Nome**

Matricola

---

1 (12 punti)

Data la classe **`Veicoli`**:

```
public class Veicoli {
     private int ruote;
     private String classe;  //esempio: carro, auto, triciclo
     private boolean inMovimento;
     public boolean avvia() {
          if (!inMovimento) inMovimento = true;
          return inMovimento;
     }
     public String toString() {
          return "veicolo "+classe+" con "+ruote+" ruote. "+
          ((inMovimento)?"In moto. ":"");
     }
```

- Definire il costruttore della classe **`Veicoli`** con due parametri, di tipo **`int`** e **`String`**, per inizializzare i campi **`ruote`** e **`classe`**

```
Veicoli(int r, String c) {
   ruote = r;
   classe = c;
}
```

- Definire il metodo **`public boolean arresta()`**

```
public boolean arresta(){
        if (inMovimento) inMovimento = false;
        return inMovimento;
}
```

- Cosa dovremmo modificare se volessimo essere in grado di registrare il numero di oggetti istanziati della classe? **Aggiungere un campo statico e un eventuale metodo statico, utile soprattutto se si dichiara il campo con visibilita' `private`:**

```
private static int count;
public static int getCount() {
   return count;
}
```

e aggiungere l'istruzione **`count++`** nel codice del costruttore;

```
Veicoli(int r, String c) {
   ruote = r;
   classe = c;
   count++;
```

```
}
```

- Definire una classe **Macchina**, estensione della precedente, i cui oggetti modellano le comuni autovetture. Ogni oggetto della classe fornisce i seguenti dati: **acceso**, che indica se il motore e' avviato e **marcia**, che indica la marcia attuale della vettura in movimento (la marcia **0** indica la vettura ferma e **-1** la vettura in retromarcia).

```
class Macchina extends Veicoli {
   private boolean acceso;
   private int marcia;
   …
  }
```

- Definire un costruttore della classe **Macchina** senza parametri (per default il valore del campo **classe** e' "auto")

```
Macchina() {
   super(4, "auto");
}
```

- Sovrascrivere il metodo **avvia()** che deve accendere il motore cambiando il valore del campo **acceso** e mettere in movimento l'autovettura, senza riscrivere inutilmente codice.

```
public boolean avvia() {
   super.avvia();
   marcia = 1;
   return acceso = true;
}
```

- Definire il metodo **accelera()** che incrementa la marcia corrente, se la vettura e' in moto e la riporta all'ambiente chiamante

```
int accelera() {
   if (acceso) marcia++;
   return marcia;
}
```

- Sovrascrivere il metodo **toString()** in modo da ottenere il seguente effetto di stampa (si faccia uso dell'operatore condizionale per valutare il campo <acceso>:

```
Veicolo auto con 4 ruote. <In moto.|> Motore <acceso>. Marcia <marcia>
public String toString() {
   return super.toString() +
   "Motore "+ ((acceso)?"acceso. ":"spento. ") +
   "Marcia " + marcia;
}
```

---

2 (4 punti)

È corretta la seguente dichiarazione? **SI**

```
Veicoli mezzoTrasporto = new Macchina();
```

Quale metodo **avvia()** viene eseguito nell'istruzione seguente?

```
mezzoTrasporto.avvia();
```

quello della classe `Macchina`
Posso invocare il metodo **`accelera()`** per la variabile **`mezzoTrasporto`**? **Si, opportunamente invocato:**

```
((Macchina)mezzoTrasporto).accelera()
```

---

3 (4 punti)
Supponendo di aver dichiarato una variabile **`garage`**

```
Veicoli[] garage = new Veicoli[3];
```

Valutare il risultato dell'esecuzione delle istruzioni:

```
for (int i=0; i < garage.length; i++)
     out.println(garage[i]);
```

```
null
null
null
```

```
for (int i=0; i < garage.length; i++)
    out.println(garage[i].avvia())
```

```
Exception in thread "main": java.lang.NullPointerException
```

Inizializzate ora la variabile **`garage`** con veicoli diversi (carri, tricicli, bici) e macchine a vostra scelta

```
garage[0] = new Veicoli(3, "triciclo");
garage[1] = new Veicoli(2, "bici");
garage[2] = new Macchina();
```

Valutate ora il risultato dell'esecuzione delle istruzioni

```
for (int i=0; i < garage.length; i++)
     out.println(garage[i]);
```

```
Veicolo triciclo con 3 ruote.
Veicolo bici con 2 ruote.
Veicolo auto con 4 ruote. Motore spento. Marcia 0
```

```
for (int i=0; i < garage.length; i++)
    out.println(garage[i].avvia())
```

```
true
true
true
```

4 (5 punti)

Considerate le seguenti classi:

```
public class C {
   static int h = 3;
   private int x = 6;
   public int i = 7;

   public void inc() {
      x++; i++; h++;
   }

   public int sum() {
      return x + h + i;
   }
}
```

```
public class B extends C {
   public int i = 4, k = 5;

   public int sum() {
      int j = super.sum();
      j = j + i + k;
      return j;
   }
```

```
public class A {
   public static void main (String[] a) {
      C w  = new C();
      out.println(w.sum());
      w = new B();
      out.println(w.sum());
   }
}
```

Indicare l'output prodotto dal metodo **main** della classe **A**:

16
25

Indicare se le seguenti affermazioni sono vere o false:

	Vero	Falso
1. **C** ha un metodo statico		F
2. **A** è superclasse di **C** e **B**		F
3. ogni istanza di **C** ha un campo statico		F
4. **B** è superclasse di **C**		F
5. **C** ha un costruttore	V	
6. **C** è una superclasse di **B**	V	
7. **C** ha un campo statico	V	
8. nel codice di **A** si può utilizzare il campo **k** di **B**	V	
9. nel codice di **B** si può utilizzare il campo **x** di **C**		F
10. nella classe **B** la variabile **i** oscura quella di **C**	V	

---

5 (5 punti)

Scrivere un ciclo di acquisizione di input per stringhe di caratteri (nomi ad esempio), che dovranno popolare un array. Il ciclo termina quando si digita "**return**> al prompt. Il prompt sia "**nome?** ".

```
String[] archivio = new String[MAX];
String nome = in.readLine("nome? ");
int indice = 0;
while(!nome.equals("")&& indice < MAX){
     archivio[indice++] = nome;
     nome = in.readLine("nome? ");
```

```
}
```

Nel caso si volesse usare il tipo **StringBuffer** come tipo base dell'array l'unica differenza sta nell'istruzione d'assegnamento delle singole posizioni dell'array, oltre che ovviamente nella dichiarazione dell'array:

```
StringBuffer[] archivio = new StringBuffer[MAX];
String nome = in.readLine("nome? ");
int indice = 0;
while(!nome.equals("") && indice < MAX){
     archivio[indice++] = new StringBuffer(nome);
     nome = in.readLine("nome? ");
}
```