Corso di laurea in Comunicazione digitale — Programmazione I

AA 2009/10 — turno II, Prof. M.A. Alberti

Prova scritta V appello – 23 settembre 2010

Cognome **Nome**

Matricola

1

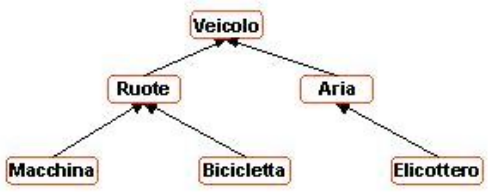
La classe `ParcoVeicoli` modella il parco mezzi per il trasporto di merci e persone di una azienda, rappresentato mediante un array di oggetti di classe `Veicolo`.
Il metodo `main` della classe nel caso particolare di una specifica azienda potrebbe essere:

```
public class ParcoVeicoli {
   Veicolo[] parco = new Veicolo[MAX];
   parco[0]= new Macchina(4, "benzina", 1600);
   parco[1]= new Macchina(3, "elettrica", 0);
   parco[2]= new Bicicletta(2, "a pedale", 12);
   parco[3]= new Bicicletta(2, "a pedale", 4);
   parco[4]= new Elicottero("elica");
   System.out.println("parco veicoli di "+Veicolo.numeroVeicoli()
                                    +" veicoli per il trasporto.");
   for (int i = 0; i < Veicolo.numeroVeicoli(); i++)
      System.out.println((i+1)+ ". " + veicoli[i]);
}
```

La cui esecuzione produce il seguente output:

```
parco veicoli di 5 veicoli per il trasporto.
1. merci e persone su strada: 4 ruote, benzina, cilindrata 1600
2. merci e persone su strada: 3 ruote, elettrica, cilindrata 0
3. persone su strada: 2 ruote, a pedale, cambio 12 marce
4. persone su strada: 2 ruote, a pedale, cambio 4 marce
5. diporto aereo: elica
```

Per capire come viene prodotto l'output precedente occorre conoscere la gerarchia delle classi coinvolte:



e tenere presente la definizione della classe `Veicolo`:

```
abstract class Veicolo {
   private String uso;
   private static int numero;

  Veicolo(String u) {
     uso = u;
     numero++;
```

```
}

abstract String classe();
public String toString() {
   return this.classe() + uso + ": " ;
}
public static int numeroVeicoli() {
   return numero;
}
```

a. La classe `Veicolo` di che tipo è? **classe astratta**

b. E' corretta l'istruzione `veicoli[n] = new Veicolo("privato");` ? SI **NO**

c. Si possono istanziare oggetti di classe `Veicolo` ? SI **NO**

d. La classe `Ruote` è così definita (completare ove presenti …) :

```
abstract class Ruote extends Veicolo {
   private int numeroRuote;
   private String tecnica;
   Ruote(int r, String t)   {
        super(" su strada");
        numeroRuote = r;
        tecnica = t;
   }
   public String toString() {
        return super.toString()+numeroRuote+" ruote, "+tecnica;
   }
}
```

è possibile istanziare oggetti di classe `Ruote` ? SI **NO**

e. La classe `Macchina` estende la classe `Ruote`. Definirne il costruttore, riempiendo gli spazi lasciati vuoti:

```
class Macchina extends Ruote  {
   private int cilindrata;
   Macchina(int r, String t, int cil) {
        super (r, t);
        cilindrata = cil;
   }
   public String classe() {
        return "merci e persone";
   }
   public String toString() {
        return super.toString()+", cilindrata "+cilindrata;
   }
}
```

f. La classe `Bicicletta`, che pure estende la classe `Ruote`, è definita dal codice seguente di cui si chiede di riempire gli spazi lasciati vuoti:

```
class Bicicletta extends Ruote {
```

```
    int marce;
    Bicicletta(int r, String t, int m) {
        super(r, t);
        marce = m;
    }
    public String classe() {
    //implementa il metodo astratto con il tipo di trasporto, vedi output
        return "persone";
    }
    public String toString() {
    //completare studiando l'output
        return super.toString() + ", cambio " + marce + " marce";
    }
}
```

g. Considerate la classe astratta Aria:

```
abstract class Aria extends Veicolo {
    private String tecnica;
    Aria(String t) {
        super( " aereo");
        tecnica = t;
    }
    public String toString() {
        return super.toString() + tecnica;
    }
}
```

Perché è definita come classe astratta? **perché non implementa il metodo astratto classe**

h. Implementare il metodo `classe()` per la classe `Elicottero`, analizzando l'output e i metodi `toString()` delle classi in gerarchia.

```
public String classe() {
    return "diporto";
}
```

2

Data la funzione ricorsiva seguente:

```
public static int f(int m, int n) {
   if (m < 5)
      return n;
   else if (m > n)
          return 1 + f(m-1, n+1);
       else
           return 1 - f(m-2, n+2);
   }
```

calcolate il valore riportato dalle funzioni e indicate le successive chiamate ricorsive nei diversi casi:

```
f(7, 1) = 7
   f(6, 2)
      f(5, 3)
```

```
        f(4,4)

f(6, 2) = 6
   f(5, 3)
      f(4,4)

f(7, 7) = 11
   f(5, 9)
      f(4, 11)
```

3

Scrivere l'espressione booleana che controlla che la variabile `numero` dichiarata di tipo `int` appartenga all'intervallo `[-5, 5)` - estremo inferiore incluso e superiore escluso - oppure sia uguale a `99`.

```
(num >= -5 && num < 5) || num == 99
```

5

Definire una classe `Tabelline` che inizializza una struttura dati a matrice di dimensione `10 x 10` con le tabelline dei primi 10 numeri da 1 a 9 (cioè righe e colonne con lo stesso indice contengono la tabellina del numero rappresentato dall'indice). Ovvero inizializzi la matrice nel seguente modo:

```
1       2       3       4       5       6       7       8       9
2       4       6       8      10      12      14      16      18
3       6       9      12      15      18      21      24      27
4       8      12      16      20      24      28      32      36
5      10      15      20      25      30      35      40      45
6      12      18      24      30      36      42      48      54
7      14      21      28      35      42      49      56      63
8      16      24      32      40      48      56      64      72
9      18      27      36      45      54      63      72      81
```

```
public class Tabelline {
     int[][] dati;
     Tabelline(){
          int[][] matrice = new int[9][];
          for (int i = 0; i < matrice.length; i++) {
               matrice[i] = new int[9];
               for (int j = 0; j < matrice[i].length; j++)
                    matrice[i][j] = (i+1) * (j+1);
          }
          dati = matrice;
     }

     public static void main(String[] args) {
          Tabelline t = new Tabelline();
          for (int i = 0; i < t.dati.length; i++) {
               for (int j = 0; j < t.dati[i].length; j++) {
                    System.out.print(t.dati[i][j] + "\t");
               }
               System.out.println();
          }
     }
```

```
}
```