Esercizio 1 (15 punti)

Si definisca una gerarchia di classi per rappresentare dei veicoli:

si definisca la classe Veicolo che contiene il campo marca (di classe String)

si definisca la classe Moto sottoclasse di Veicolo che contiene:

- il campo cilindrata (un intero),
- il metodo costruttore (che ha la cilindrata come parametro)
- un metodo che restituisce il bollo da pagare (cilindrata moltiplicata per 1.5).

si definisca la classe Auto sottoclasse di Veicolo che contiene:

- il campo potenza (un double),
- il metodo costruttore (che ha la potenza come parametro)
- un metodo che restituisce il bollo da pagare (la potenza moltiplicata per 2.5).

Si definisca una classe Garage che contiene come campo un array V in cui è possibile inserire sia Moto che Auto.
Si definisca inoltre un costruttore della classe Garage che ha due parametri interi m e n e che instanzia il vettore V con m+n elementi inserendo m Moto (di cilindrata casuale) ed n Auto (di potenza casuale)
Si definisca infine un metodo della classe Garage che restituisce la somma dei bolli da pagare per Moto ed Auto presenti nel vettore V.

Se necessario si modifichi la gerarchia di classi definita sopra.

Soluzione

```
import java.util.*;

abstract class Veicolo {

   String marca;

   public Veicolo(String marca)
   {
      this.marca=marca;
   }

   public abstract double getBollo();
}



class Moto extends Veicolo
{
   int cilindrata;
   private final  double aliquota = 1.5;

   public Moto(String marca, int cilindrata)
   {
      super(marca);
      this.cilindrata=cilindrata;
   }

   public double getBollo()
   {
      return cilindrata*aliquota;
   }
}
```

```
class Auto extends Veicolo
{
    double potenza;
    private final double ali=2.5;

    public Auto(String marca, int potenza)
    {
        super(marca);
        this.potenza=potenza;
    }

    public double getBollo()
    {
        return potenza*ali;

    }
}


public class Garage
{
   private Veicolo[] array;
   private final  int CIL_RANGE=1000;
   private final  int POT_RANGE=150;

   public Garage(final int n, final int m)
   {
       array=new Veicolo[n+m];
       for(int i=0;i<m;i++)
           array[i]=new Moto("Guzzi",((int)(CIL_RANGE*Math.random())));
        for(int i=m;i<n+m;i++)
          array[i]=new Auto("Alfa",((int)(POT_RANGE*Math.random())));
   }


   public double getSomma()
   {
       int somma=0;
       for(int i=0; i<array.length;i++)
           somma+=array[i].getBollo();
       return somma;
   }

}
```

Esercizio 2 (10 punti)

Si definisca una lista concatenata tramite puntatori a partire dalle classi:
Elemento, che contiene i campi: *valore* (di tipo intero) e *successivo* (di classe *Elemento*);
Lista, che contiene il campo *testa* (di classe *Elemento*), che rappresenta il riferimento al primo elemento della lista.
Nella classe *Lista* si definisca:

- il costruttore (che costruisce una lista vuota)
- il metodo *minimo*, che restituisce il valore minimo presente nella lista (o zero se la lista è vuota);
- il metodo *leggiFile* che, data come parametro una stringa che rappresenta il nome di un file di caratteri che per ogni riga contiene un valore intero, inserisce nella lista i valori numerici contenuti nel file (l'inserimento va fatto in testa).

Si realizzi infine una applicazione che:

- definisce un oggetto di classe lista
- usa il metodo leggiFile per inserire nella lista i valori contenuti nel file di testo "esame.txt"
- stampa a video il valore minimo contenuto nella lista.

Soluzione:

```
import java.io.*;
import java.util.*;

class Elemento
{
   int valore;
   Elemento next=null;
}

public class Lista
{
   private Elemento testa;


   public Lista()
   {
      testa=null;
   }

   public void leggiFile(String file)
   {
         BufferedReader fileIn= null;
         String num=null;
         int n=0;
      try {
         fileIn=new BufferedReader(new FileReader(file));
         while((num=fileIn.readLine())!= null) {
            n=(Integer.parseInt(num));
            Elemento nuovo=new Elemento();
            nuovo.valore=n;
            nuovo.next=testa;
            testa=nuovo;
          }
      }
      catch(IOException e) {
        System.out.println("ERRORE: " + e.getMessage());
      }
   }
```

```
 public int getMinimo()
   {
      Elemento temp;
      if(testa==null) return 0;
      else {
         int min=testa.valore;
         for(temp=testa;temp!=null;temp=temp.next) {
            if(min>temp.valore)
               min=temp.valore;
         }
         return min;
      }
   }


   public static void main(String arg[])
   {
     String file=null;
     try {
        System.out.println("inserisci il nome del file da aprire ...");
        BufferedReader reader=new BufferedReader(new InputStreamReader(System.in));
        file=reader.readLine();
        Lista lista=new Lista();
        lista.leggiFile(file);
        System.out.println("il valore minimo fra quelli inseriti e': " + lista.getMinimo());
     }
     catch(IOException e) {
        System.out.println("ERRORE: " + e.getMessage());
     }
   }
}
```

Esercizio 3 (5 punti)

Si definisca una applicazione grafica costituita da una finestra che contiene (in questo ordine):

- due campi di testo (finalizzati a contenere dei numeri)
- una etichetta
- un bottone

cliccando sul bottone deve comparire sull'etichetta il valore maggiore fra quelli presenti nei campi di testo.

Nel caso in cui si clicchi sul bottone quando uno dei campi di testo non contiene un numero, entrambi i campi di testo devono tornare vuoti; si gestisca questa situazione tramite una eccezione.

Soluzione

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class MaxFrame extends JFrame implements ActionListener {
   private JTextField testosopra;
   private JTextField testosotto;
   private JLabel label;
   private JButton button;

   public MaxFrame() {
      getContentPane().setLayout(new GridLayout(4,1));
      testosopra=new JTextField(10);
      testosotto=new JTextField(10);
      label=new JLabel("max= ...");
      button=new JButton("calcola max !");
      button.addActionListener(this);

      this.addWindowListener(new WindowAdapter(){
         public void windowClosing(WindowEvent ev) {System.exit(0);}
      });

      getContentPane().add(testosopra);      getContentPane().add(testosotto);
      getContentPane().add(button);          getContentPane().add(label);

      pack();
   }

   public void actionPerformed(ActionEvent evt) {
      try{
         if(Integer.parseInt(testosotto.getText())>Integer.parseInt(testosopra.getText()))
            label.setText(testosotto.getText());
         else
            label.setText(testosopra.getText());
      }
      catch(NumberFormatException e){
         testosotto.setText("");
         testosopra.setText("");
         label.setText("max= ...");
      }
   }

   public static void main(String arg[]) {
      new MaxFrame().setVisible(true);
   }
}
```