Les interfaces graphiques

Exercice I : Une interface graphique et ligne de commande pour les lampes

1. Implanter une interface graphique pour l'ensemble des classes de la lampe progressive.
2. Implanter une seconde interface, concurrente à la première qui fonctionne en ligne de commandes.

```
package fr.utt.sit.lo02.td.swing.lampes.core;

import java.util.Observable;

public class Lampe extends Observable {

    public final static int PUISSANCE_STANDARD = 100;
    public final static int NOMBRE_UTILISATION_MAXIMUM = 100;

    private int puissance;
    private boolean allumee;
    private int nombreUtilisations;

    public Lampe(int puissance) {
      this.puissance = puissance;
      this.allumee = false;
      this.nombreUtilisations = 0;
    }

    public int getPuissance() {
      return puissance;
    }

    public boolean isAllumee() {
      return allumee;
    }

    public void allumer() throws EtatInvalideException,
LampeGrilleeException{
      if (this.allumee == true) {
          throw new EtatInvalideException("La lampe est déjà allumée");
      } else {
          if (nombreUtilisations < Lampe.NOMBRE_UTILISATION_MAXIMUM) {
          this.allumee = true;
          this.nombreUtilisations++;
          this.setChanged();
          this.notifyObservers();
          } else {
            throw new LampeGrilleeException();
          }
      }
    }

    public void eteindre() throws EtatInvalideException {
      if (this.allumee == false) {
          throw new EtatInvalideException("La lampe est déjà éteinte");
      } else {
          this.allumee = false;
          this.setChanged();
          this.notifyObservers();
      }
    }
```

```
    public String toString() {
      StringBuffer sb = new StringBuffer();
      sb.append("Lampe (");
      sb.append("Puissance : " + puissance);
      sb.append("\t");
      sb.append("Allumée : " + allumee);
      sb.append(")\n");
      return sb.toString();
    }
}
```

```
package fr.utt.sit.lo02.td.swing.lampes.core;

import java.util.ArrayList;

public class Commutateur {

    private ArrayList<Lampe> lampes;
    private int etat;
    private int nombreEtats;

    public Commutateur (int nombreLampes) {

      etat = 0;
      nombreEtats = nombreLampes * 2;
      lampes = new ArrayList<Lampe>(nombreLampes);

      Lampe lampeFaible = new Lampe (Lampe.PUISSANCE_STANDARD / 2);
      lampes.add(0, lampeFaible);

      for (int i = 1; i < nombreLampes; i++) {
          Lampe lampeStandard = new Lampe (Lampe.PUISSANCE_STANDARD);
          lampes.add(i, lampeStandard);
      }
    }

    public int getNombreLampes() {
      return this.lampes.size();
    }

    public ArrayList<Lampe> getLampes() {
      return lampes;
    }

    public void commuter() throws CommutationImpossibleException {
      etat = (etat + 1) % nombreEtats;

      int puissanceRequise = etat * Lampe.PUISSANCE_STANDARD / 2;

      try {
          // Allumage/eteignage de la lampe faible
          Lampe lampeFaible = lampes.get(0);

          if ((puissanceRequise % Lampe.PUISSANCE_STANDARD) != 0) {
            lampeFaible.allumer();
          } else {
            lampeFaible.eteindre();
          }

          // Allumage/Eteignage des autres lampes
          for (int i = 1; i < lampes.size(); i++) {
            Lampe lampe = lampes.get(i);
```

```
            if (2 * i > etat) {
                if (lampe.isAllumee()) {
                  lampe.eteindre();
                }
            } else {
                if (lampe.isAllumee() == false) {
                  lampe.allumer();
                }

            }
          }

      } catch (EtatInvalideException e) {
          //          System.out.println("Attention : " + e.getMessage());
      } catch (LampeGrilleeException e) {
          CommutationImpossibleException cie = new
CommutationImpossibleException();
          cie.initCause(e);
          throw cie;
      }
    }

    public String toString() {
      StringBuffer sb = new StringBuffer();
      sb.append("Commutateur (");
      sb.append("Etat : " + etat);
      sb.append(")\n");
      for (int i = 0; i < lampes.size(); i++) {
          sb.append(lampes.get(i));
      }
      return sb.toString();
    }
}
```

```
package fr.utt.sit.lo02.td.swing.lampes.core;

public class Interrupteur {
    private Commutateur commutateur;

    public Interrupteur (int nombreLampes) {
      commutateur = new Commutateur(nombreLampes);
    }

    public void appuyer() {
      try {
          commutateur.commuter();
      } catch (Exception e) {
          e.printStackTrace();
      }
    }

    public Commutateur getCommutateur () {
      return this.commutateur;
    }

    public String toString() {
      StringBuffer sb = new StringBuffer();
      sb.append("Interrupteur\n");
      sb.append(commutateur);
      return sb.toString();
    }
```

```
    public static void main (String[] args) {

      Interrupteur inter = new Interrupteur(5);

      for (int i = 0; i < 20; i++) {
          inter.appuyer();
          System.out.println(inter);
      }
    }
}
```

```
package fr.utt.sit.lo02.td.swing.lampes.gui;

import java.awt.Color;
import java.util.Observable;
import java.util.Observer;

import javax.swing.JButton;

import fr.utt.sit.lo02.td.swing.lampes.core.Lampe;

public class VueLampe extends JButton implements Observer {

    private static final long serialVersionUID = 4455268480233165375L;

    private Lampe lampe;

    public VueLampe (Lampe l) {
      super("Eteinte");
      this.lampe = l;
      l.addObserver(this);
      super.setForeground(this.computeColor());
    }

    @Override
    public void update(Observable arg0, Object arg1) {
      super.setForeground(this.computeColor());
      super.setText(this.computeTexte());
      super.repaint();
    }

    private String computeTexte() {
      if (lampe.isAllumee()) {
          return new String("Allumée");
      } else {
          return new String("Eteinte");
      }
    }

    private Color computeColor () {
      Color c;
      if (lampe.isAllumee()) {
          if (lampe.getPuissance() == Lampe.PUISSANCE_STANDARD) {
            c = Color.WHITE;
          } else {
            c = Color.GRAY;
          }
      } else {
          c = Color.BLACK;
      }
      return c;
    }
```

```
}
```

```
package fr.utt.sit.lo02.td.swing.lampes.gui;

import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;

import javax.swing.*;

import fr.utt.sit.lo02.td.swing.lampes.cli.VueTexte;
import fr.utt.sit.lo02.td.swing.lampes.core.Interrupteur;
import fr.utt.sit.lo02.td.swing.lampes.core.Lampe;

public class VueLampes extends JFrame {

    private static final long serialVersionUID = 337134961473143560 1L;

    public static int NOMBRE_DE_LAMPES = 4;

    private JButton boutonInterrupteur;
    private ArrayList<JButton> boutonLampes;

    private Interrupteur interrupteur;

    public VueLampes (Interrupteur inter) {

      this.interrupteur = inter;

      ArrayList<Lampe> lampes = interrupteur.getCommutateur().getLampes();
      boutonLampes = new ArrayList<JButton>();

      JPanel panelLampes = new JPanel();
      panelLampes.setLayout(new GridLayout(1, lampes.size()));

      for (int i = 0; i < lampes.size(); i++) {
          JButton boutonLampe = new VueLampe(lampes.get(i));
          boutonLampes.add(i, boutonLampe);
          panelLampes.add(boutonLampe);
      }

      boutonInterrupteur = new JButton("Interrupteur");

      boutonInterrupteur.addActionListener(new ActionListener() {
          public void actionPerformed(ActionEvent e) {
            interrupteur.appuyer();
          }
      });

      Container c = this.getContentPane();
      c.add(boutonInterrupteur, BorderLayout.SOUTH);
      c.add(panelLampes, BorderLayout.NORTH);
    }

    public static void main(String[] args) {

      Interrupteur inter = new Interrupteur(VueLampes.NOMBRE_DE_LAMPES);
      VueLampes vl = new VueLampes(inter);
      VueLampes vl2 = new VueLampes(inter);

      VueTexte vt = new VueTexte(inter);
      vt.demarrer();
```

```
      vl2.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      vl2.pack();
      vl2.setVisible(true);

      vl.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      vl.pack();
      vl.setVisible(true);
    }
}
```

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.Observable;
import java.util.Observer;

import fr.utt.sit.lo02.td.swing.lampes.core.Interrupteur;
import fr.utt.sit.lo02.td.swing.lampes.core.Lampe;

public class VueTexte implements Observer, Runnable {

    public static String QUITTER = "Quit";
    public static String COMMUTER = "C";
    public static String PROMPT = ">";

    private Interrupteur interrupteur;
    private boolean active;

    public VueTexte(Interrupteur inter) {
      this.interrupteur = inter;

      ArrayList<Lampe> lampes = interrupteur.getCommutateur().getLampes();

      for (int i = 0; i < lampes.size(); i++) {
          lampes.get(i).addObserver(this);
      }

      this.active = false;
    }

    public void demarrer() {
      this.active = true;
      Thread t = new Thread(this);
      t.start();
    }

    public void arreter() {
      this.active = false;
     }

    public void run() {

      String saisie = null;

      System.out.println("Taper " + VueTexte.COMMUTER + " pour commuter ; "
+ VueTexte.QUITTER + " pour quitter.");

      do {
          saisie = this.lireChaine();
```

```
            if (saisie != null) {
              if (saisie.equals(VueTexte.COMMUTER) == true) {
                  interrupteur.appuyer();
              } else if (saisie.equals(VueTexte.QUITTER) == true) {
                  this.arreter();
              } else {
                  System.out.println("Commande non reconnue...");
              }
            }
      } while (this.active == true);
      System.exit(0);
    }

    private String lireChaine() {
      BufferedReader br = new BufferedReader (new
InputStreamReader(System.in));
      String resultat = null;
      try {
          System.out.print(VueTexte.PROMPT);
          resultat = br.readLine();
      } catch (IOException e) {
          System.err.println(e.getMessage());
      }
      return resultat;
    }

    @Override
    public void update(Observable arg0, Object arg1) {
      if (arg0 instanceof Lampe) {
          Lampe lampe = (Lampe) arg0;
          System.out.print("Lampe " + lampe.getPuissance() + " est ");
          if (lampe.isAllumee()) {
            System.out.println("allumée.");
          } else {
            System.out.println("éteinte.");
          }

      } else {
          System.err.println("Objet observable non reconnu !");
      }
    }
}
```

```
package fr.utt.sit.lo02.td.swing.lampes.core;

public class CommutationImpossibleException extends Exception {

    private static final long serialVersionUID = 1L;

}
```

```
package fr.utt.sit.lo02.td.swing.lampes.core;

public class EtatInvalideException extends Exception {

    private static final long serialVersionUID = 1L;

    public EtatInvalideException (String message) {
      super(message);
    }
}
```

```java
package fr.utt.sit.lo02.td.swing.lampes.core;

public class LampeGrilleeException extends Exception {

    private static final long serialVersionUID = 1L;

}
```