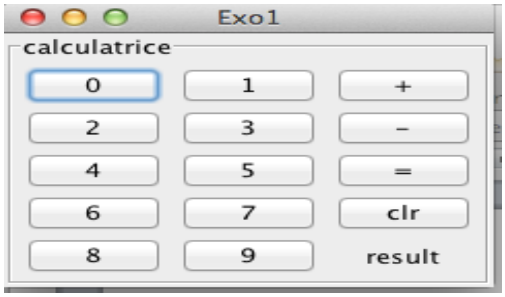
TD/TP – Partie 1
PO2 Licence Informatique 3
(2014 – 2015)

Exercice 1

Ecrire une interface graphique qui contient :

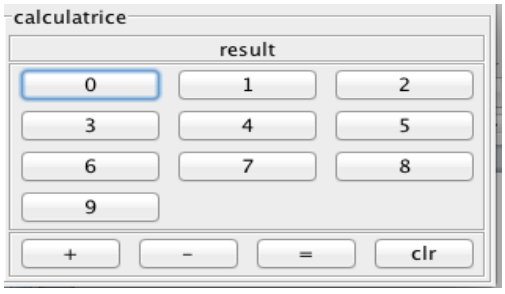
- 10 boutons correspondant aux chiffres : 0, 1, …,9
- 4 boutons correspondant aux opérations : + - = clr
- 1 label pour afficher le résultat



Exercice 2

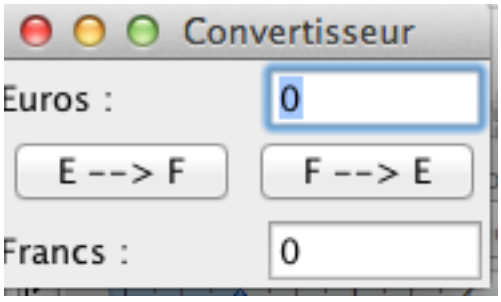
Idem avec 3 zones :

- une zone supérieure pour le résultat
- une zone au milieu pour des chiffres
- une zone inférieur pour des opérations



Exercice 3

Ecrire une interface graphique qui convertit des euros en francs et inversement.



Compléter le code suivant :

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

public class Convertisseur extends JFrame {
  JTextField Euro;
  JTextField Franc;

  public Convertisseur() {
    super("Convertisseur");
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    Euro = new JTextField("0", 6);
    Franc = new JTextField("0", 6);

    JButton boutonEuro2Franc = new JButton("E --> F");
    JButton boutonFranc2Euro = new JButton("F --> E");

    JPanel panelHaut = new JPanel(new BorderLayout());
    panelHaut.add(new JLabel("Euros :  "), BorderLayout.WEST);
    panelHaut.add(Euro, BorderLayout.EAST);

    JPanel panelMilieu = new JPanel(new BorderLayout());
    panelMilieu.add(boutonEuro2Franc, BorderLayout.WEST);
    panelMilieu.add(boutonFranc2Euro, BorderLayout.EAST);

    JPanel panelBas = new JPanel(new BorderLayout());
    panelBas.add(new JLabel("Francs :  "), BorderLayout.WEST);
    panelBas.add(Franc, BorderLayout.EAST);

    JPanel panelTout = new JPanel(new BorderLayout());
    panelTout.add(panelHaut, BorderLayout.NORTH);
    panelTout.add(panelMilieu, BorderLayout.CENTER);
    panelTout.add(panelBas, BorderLayout.SOUTH);

    this.getContentPane().add(panelTout);
    this.pack();
    this.setVisible(true);
  }

  public static void main(String[] args) {
    javax.swing.SwingUtilities.invokeLater(new Runnable() {
        public void run() {
           new Convertisseur();
        }
    });
  }
}
```

1. Réaliser deux écouteurs avec classe interne anonyme pour traiter des événements provenant des boutons.
2. Réaliser un écouteur avec classe interne non anonyme pour traiter des événements provenant des boutons. Utiliser la méthode « getActionCommand » de « ActionEvent » pour distinguer quel bouton a été cliqué.
3. Comment prendre en compte la validation de la saisie par la touche « entrée » ?

Exercice 4

1. Ecrire une interface graphique permettant de choisir une couleur parmi 5 à l'aide de bouton radios et de l'afficher dans une zone rectangulaire. (Fig.1)
2. Enrichir l'interface par une liste de choix des couleurs. (Fig. 2)
3. Enrichir l'interface par un label qui affiche, en plus, la valeur hexadécimale de la couleur. (Fig. 3)
   Ci-dessous le code convertissant un entier en String de 2 chiffres hexa :

```
private String toHex(int val) {
        String result = Integer.toHexString(val).toUpperCase();
        if (result.length() == 1)
                return "0"+result ;
        else
                return result;
}
```

   Dans cet exercice, il devient difficile de gérer les interactions entre contrôleur et afficheurs, …

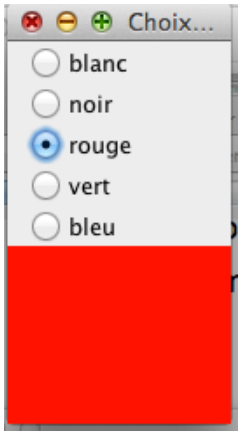


Fig. 1

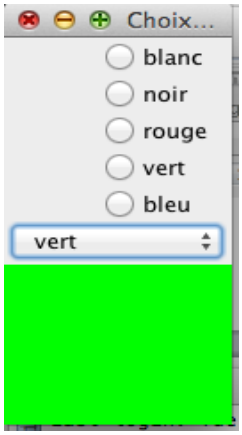


Fig. 2

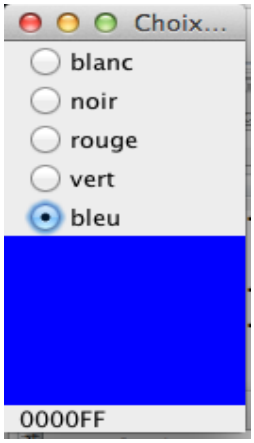
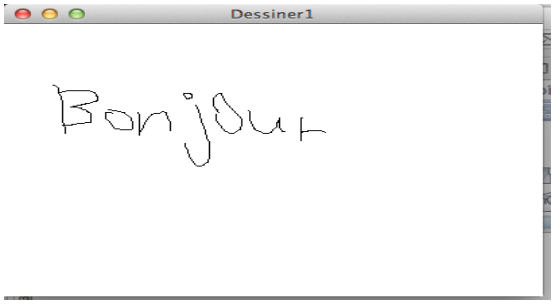


Fig. 3

Exercice 5

Réaliser une interface graphique pour dessiner à la souris :



Dans cette 1ère version, le masquage de la fenêtre provoquera la perte du dessin. Compléter la classe Dessiner1.java :

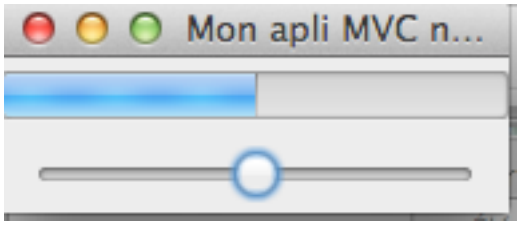
```
import javax.swing.*;
import java.awt.*;
```

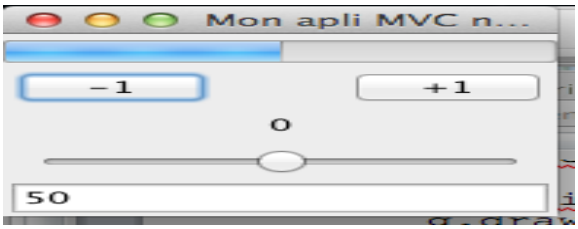
```
import java.awt.event.*;

public class Dessiner1 extends JFrame {
  private int lastX, lastY;
  private JPanel panelDessin;
  public Dessiner1() {
    super("Dessiner1");
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    panelDessin = new JPanel();
    panelDessin.setPreferredSize(new Dimension(400,300));
    panelDessin.setBackground(Color.WHITE);
....
    this.getContentPane().add(panelDessin);
    this.pack();
    this.setVisible(true);
  }
  private void dessiner(int x0, int y0, int x1, int y1) {
    if ((x0 != -1) && (y0 != -1)) {
      Graphics g = panelDessin.getGraphics();
      g.drawLine(x0, y0, x1, y1);
    }
  }
  public static void main(String[] args) {
    javax.swing.SwingUtilities.invokeLater(new Runnable() {
        public void run() {
           new Dessiner1();
        }
    });
  }
}
```

Exercice 6

On vous fournit une interface graphique qui permet de contrôler et visualiser un niveau entre 2 bornes limites : ici, un entier compris entre 0 et 100.





Le modèle est défini dans la classe Niveau, le contrôle dans la classe Tirette, et la vue dans la classe Curseur.

Vous disposez aussi d'une classe MonAppliMVC1 qui utilise le contrôle d'une tirette et l'affichage d'un curseur.

Maintenant, vous enrichissez cette interface par l'ajoute de :

- Un contrôle par boutons +1 et -1
- Un affichage de la valeur du niveau (grâce à un JLabel)
- Un contrôle et affichage par une zone de saisie

```
import java.util.*;
import javax.swing.*;
```

```
import java.awt.*;
public class MonAppliMVC1 extends JFrame {
  public MonAppliMVC1() {
    super("Mon apli MVC numero 1");
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    Niveau modele = new Niveau(0,50,100);
    Tirette tirette = new Tirette(modele);
    Curseur curseur = new Curseur(0,50,100);
    modele.addObserver(curseur);

    JPanel panel = new JPanel();
    panel.setLayout(new GridLayout(2,1));
    panel.add(curseur);
    panel.add(tirette);
    this.getContentPane().add(panel);
    this.pack();
    this.setVisible(true);
  }

  public static void main(String[] args) {
    SwingUtilities.invokeLater( new Runnable() {
        public void run() {
          new MonAppliMVC1();
        }
      });
  }
}

------------------------
import java.util.Observable;
public class Niveau extends Observable {
  private int niveau, min, max;
  public Niveau(int mi, int n, int ma) {
    niveau=n; min=mi; max=ma;
  }
  public int getNiveau() { return niveau; }
  public int getMin() { return min; }
  public int getMax() { return max; }
  public void setNiveau(int nouveau) {
    if ((nouveau != niveau)  && (nouveau>=min) && (nouveau<=max))
    { niveau=nouveau;
      setChanged();
      notifyObservers(new Integer(niveau));
    }
  }
  public String toString() { return niveau+" in ["+min+","+max+"]"; }
}

---------------------------

import javax.swing.*;
import javax.swing.event .*;
public class Tirette extends JSlider {
  private Niveau modele;
  public Tirette(Niveau model) {
    super(model.getMin(),model.getMax(),model.getNiveau());
         this.modele=model;
    this.addChangeListener(new ChangeListener() {
        public void stateChanged(ChangeEvent e) {
          modele.setNiveau(Tirette.this.getValue() );
```

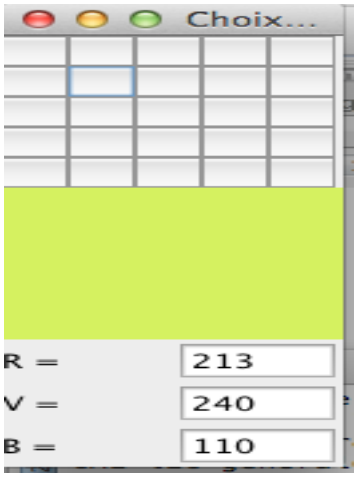
```
    }} );
  }
}

----------------------------

import java.util.*;
import javax.swing.*;
public class Curseur extends JProgressBar  implements Observer {
      public Curseur(int min, int niv, int max) {
    super(min, max);
    setValue(niv);
  }
  public void update(Observable source, Object donnees) {
    if ((source instanceof Niveau) && (donnees != null)
        && (donnees instanceof Integer))
    { int niv= (Integer)donnees;
      this.setValue(niv);
    }
  }
}
```

Exercice 7

A partir de l'interface de l'exercice 4, réaliser une nouvelle interface avec MVC qui permet à l'utilisateur de choisir des couleurs et de voir la couleur obtenue ainsi que ses intensités de rouge, vert et bleu.



Exercice 8

Améliorer l'interface de l'exercice 5, sans perte, le masquage de la fenêtre.

Compléter la classe Dessine2.java qui applique le MVC :

- Un modèle « traits déjà dessinés » à partir d'une classe « trait » de l'ensemble des points d'une ligne brisée
- Un modèle « trait en cours »

Un JPanel transparent (non opaque) affiche le dessine en cours et contient un JPanel (opaque) dans autres traits déjà dessinés.

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

public class Dessiner2 extends JFrame {
  private ModeleTraits modeleTraits;
  public Dessiner2() {
    super("Dessiner2");
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    modeleTraits = new ModeleTraits();
    PanelDessinEnCours dessinEnCours = new PanelDessinEnCours(modeleTraits);
    PanelDejaDessine dejaDessine = new PanelDejaDessine();
    dessinEnCours.add(dejaDessine);
    modeleTraits.addObserver(dejaDessine);
    this.getContentPane().add(dessinEnCours);
    this.pack();
    this.setVisible(true);
    dejaDessine.repaint();
  }
  public static void main(String[] args) {
    javax.swing.SwingUtilities.invokeLater(new Runnable() {
        public void run() {
           new Dessiner2();
        }
    });
  }
}

class PanelDessinEnCours extends JPanel  {
  private ModeleTraits modeleTraits;
  private TraitEnCours traitEnCours;
  public PanelDessinEnCours(ModeleTraits mt) {
    ....
}
class PanelDejaDessine extends JPanel implements Observer {
  private Trait[] tabTraits;
  .....
}

class Trait {
  private int[] coordX, coordY;
  public Trait(int[] tabX, int[] tabY ) {
    coordX = tabX; coordY = tabY;
  }
  public int[] getCoordX() { return coordX; }
  public int[] getCoordY() { return coordY; }
}
class TraitEnCours {
  private java.util.List<Integer> coordXY;
  public TraitEnCours() {
    coordXY = new ArrayList<Integer>();
  }
  public int getLastX() {
    int n = coordXY.size();
    if (n >= 2)
      return coordXY.get(n-2);
    else
      return -1;
  }
  public int getLastLastX() {
```

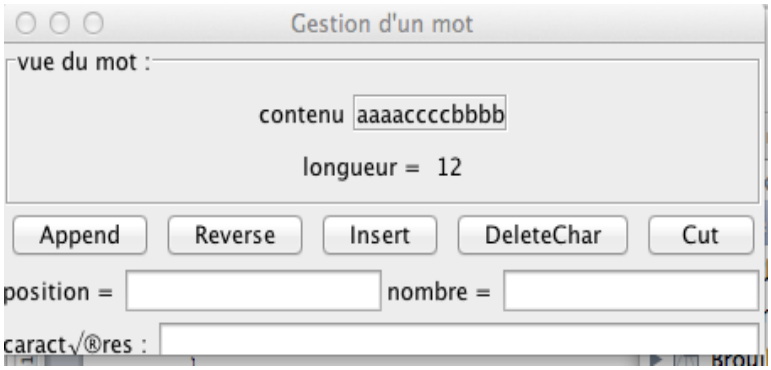
```
    int n = coordXY.size();
    if (n >= 4)
      return coordXY.get(n-4);
    else
      return -1;
  }
  public int getLastY() {
    int n = coordXY.size();
    if (n >= 2)
      return coordXY.get(n-1);
    else
      return -1;
  }
  public int getLastLastY() {
    int n = coordXY.size();
    if (n >= 4)
      return coordXY.get(n-1);
    else
      return -1;
  }
  public void add(int x, int y) {
    coordXY.add(x);
    coordXY.add(y);
  }
  public Trait getTrait() {
    int n = coordXY.size()/2;
    int[] tabX = new int[n];
    int[] tabY = new int[n];
    for (int i=0; i<n; i++) {
      tabX[i] = coordXY.get(2*i);
      tabY[i] = coordXY.get(2*i+1);
    }
    return new Trait(tabX, tabY);
  }
}

class ModeleTraits extends Observable {
  private java.util.List<Trait> listTraits;
  public ModeleTraits() {
    listTraits = new ArrayList<Trait>();
  }
  public void add(TraitEnCours traitEnCoursFini) {
    listTraits.add(traitEnCoursFini.getTrait());
    this.setChanged();
    this.notifyObservers(listTraits.toArray(new Trait[1]));
  }
}
```

Exercice 9

Ecrire une interface graphique avec MVC qui permet de gérer un mot à la console : ajout à la fin, insertion, suppression d'un caractère ou d'un morceau, renverser, … Le modèle (Mot.java) est fourni. Commencer par un contrôleur et une vue en console, puis ajouter une vue graphique, puis un contrôleur graphique.

Ci-dessous le fichier Mot.java, une capture d'écran de l'interface graphique et de la console :




```
----------

----------
app aaaa
----------
aaaa
----------
app bbbb
----------
aaaabbbb
----------
ins 4 cccc
----------
aaaaccccbbbb
----------
```

```
import java.util.*;
/** gestion d'un mot
 */
public class Mot extends Observable
{
  /** le mot*/
  private StringBuffer mot;
  /** constructeur sans parametre */
  public Mot() {
    mot = new StringBuffer("");
  }
  /** ajoute un mot à la fin */
  public void append(String ajout) {
    if (ajout != null) {
      mot.append(ajout);
      this.setChanged();
      this.notifyObservers(mot.toString());
    }
  }
  /** insert un mot  */
  public void insert(int pos, String ajout) {
    if ((ajout != null)&&(pos >= 0)&&(pos < mot.length())) {
      mot.insert(pos, ajout);
      this.setChanged();
      this.notifyObservers(mot.toString());
    }
  }
  /** supprime une partie du mot*/
  public void cut(int pos, int nombre) {
    if ((pos >= 0)&&((pos + nombre -1)  < mot.length())) {
        mot.delete(pos, pos + nombre -1);
        this.setChanged();
        this.notifyObservers(mot.toString());
      }
  }
  /** supprime une lettre du mot*/
  public void deleteChar(int pos) {
    if ((pos >= 0)&&(pos < mot.length())) {
        mot.deleteCharAt(pos);
        this.setChanged();
        this.notifyObservers(mot.toString());
      }
  }
  /** inverse les lettres du mot*/
  public void reverse() {
      mot.reverse();
      this.setChanged();
      this.notifyObservers(mot.toString());
```

```
  }
  /** retourne le contenu du mot*/
  public String toString() {
    return mot.toString();
  }
}
```