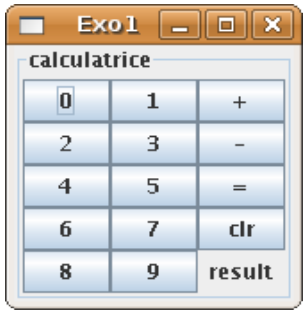
TD et TP du Module POA licence Informatique 3

Le code à compléter se trouve sur le site web : http://www.u-picardie.fr/ferment/poa

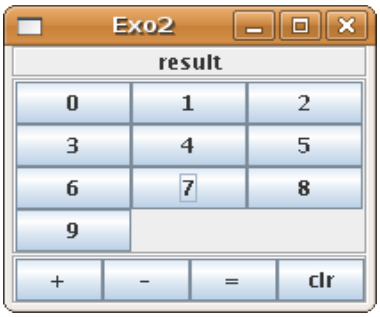
Exercice 1 : TD



Écrire la classe correspondante :
avec 10 boutons correspondant aux chiffres,
4 boutons pour + - = et clr
un label pour afficher le résultat.

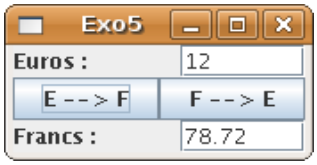
Écrire uniquement la création et le placement des composants.

Exercice 2 : TD et TP



Idem avec 3 zones :
une zone supérieure pour le résultat,
une au milieu pour le clavier numérique
une inférieure pour le clavier des opérations.
Pour le TP : Exo2_a_completer.java

Exercice 3 : TD et TP



Écrire une application graphique (classe Calculatrice1)
qui convertit des euros en francs et inversement.
Compléter le code Calculatrice1_a_completer.java

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

public class Calculatrice1 extends JFrame {
  JTextField Euro;
  JTextField Franc;
  public Calculatrice1() {
    super("Calculatrice1");
    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 panelBoutons = new JPanel(new BorderLayout());
    panelBoutons.add(boutonEuro2Franc, BorderLayout.WEST);
    panelBoutons.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(panelBoutons, 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 Calculatrice1();
        }
    });
  }
}
```

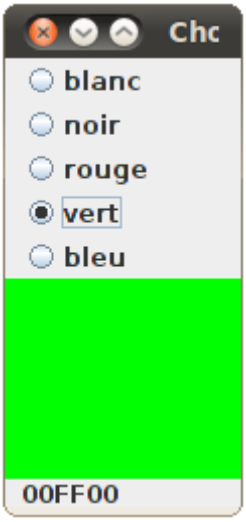
Comment prendre en compte la validation de la saisie par la touche « entrée » ?
Ecrire Calculatrice3.java.

Exercice 4 : TD

Écrire une classe ChoixCouleur1 permettant de choisir une couleur parmi 5 à l'aide de boutons radios
et de l'afficher dans une zone rectangulaire (un JPanel).

Utilisez le shéma de programmation suivant :

```
composant.addActionListener(new ActionListener() {
   public void actionPerformed(ActionEvent ae) {
     instructions de l'action à effectuer
   }
});
```



Comment faire pour ajouter un label qui afficherais, en plus, la valeur hexadécimale de la couleur ? Comme ci-contre.
Dans cette classe ChoixCouleur4 il devient difficile de gérer les interactions entre contrôleur et afficheurs …
Ci-dessous le code convertissant un entier en Sting de 2 chiffres héxa :

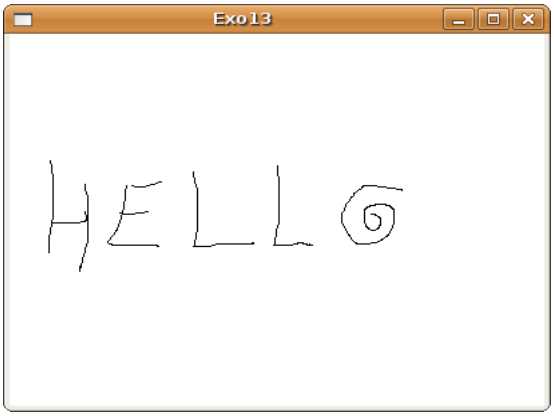
```
  private String toHex(int val) {
    String result = Integer.toHexString(val).toUpperCase();
    if (result.length() == 1)
      return "0"+result ;
    else
      return result;
  }
```

Exercice 5 : TP
Enrichir la classe de l'exercice 2 par une liste de choix des couleurs pour obtenir la classe ChoixCouleur2.

Exercice 6 : TD

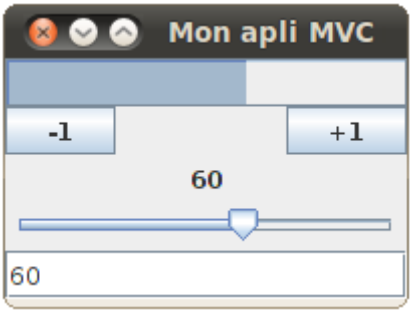
Reprendre la classe Calculatrice1 en réalisant une classe interne non anonyme écouteur des événements provenant des boutons.
Utilisez l'ActionCommand pour distinguer quel bouton a été cliqué.
La nouvelle classe s'appellera Calculatrice2.

Exercice 7 : TP



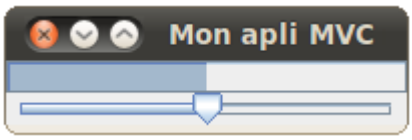
Réalisez une interface graphique pour dessiner à la souris .
Dans cette 1ère version, le masquage de la fenêtre provoquera la perte du dessin.
Compléter la classe : Dessiner1_a_completer.java

Exercice 8 : TD



Réalisez un composant 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.

Pour ce faire , vous disposez déjà d'une classe MonAppliMVC1 qui utilise le contrôle d'une Tirette et l'affichage d'un curseur. Voir ci-dessous :



Votre classe finale se nommera : MonAppliMVC2. Elle ajoutera :

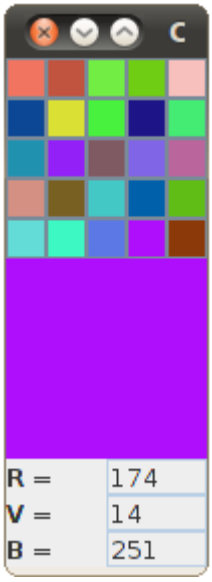
- un contrôle par boutons +1 et -1
- un affichage de la valeur du niveau (grâce à un Jlabel),
- un contrôle et affichage par une zone de saisie

```
import java.util.*;
import javax.swing.*;
import java.awt.*;
public class MonAppliMVC1 {
  public static void main(String[] args) {
    SwingUtilities.invokeLater(
      new Runnable() {
        public void run() {
          Niveau modele = new Niveau(0,50,100);
          Tirette tirette = new Tirette(modele);
          Curseur curseur = new Curseur(0,50,100);
          modele.addObserver(curseur);
          JFrame frame = new JFrame("Mon apli MVC numero 1");
          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
          JPanel panel = new JPanel();
          panel.setLayout(new GridLayout(2,1));
          panel.add(curseur);
          panel.add(tirette);
          frame.getContentPane().add(panel);
          frame.pack();
          frame.setVisible(true);
        }
      });
  }
}
------------------------
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 9 : TP

A partir de ChoixCouleur3_a_completer.java, faite fonctionner l'application graphique 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 :
les instructions relatives au pattern Observer-Observable ont été gommées.
Vous pouvez exécutez la classe : ChoixCouleur3.class … à récupérer dans le ChoixCouleur3.zip



Exercice 10 : TP

Améliorez Dessiner1 en gérant, sans perte, le masquage de la fenêtre.
Complétez la classe Dessiner2_a_completer.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 dessin en cours et contient un JPanel (opaque) des autres traits déjà dessinés.
Vous pouvez exécutez la classe : Dessiner3.class … à récupérer dans le Dessiner3.zip