Licence Info 3éme année - module Programmation Objet 2

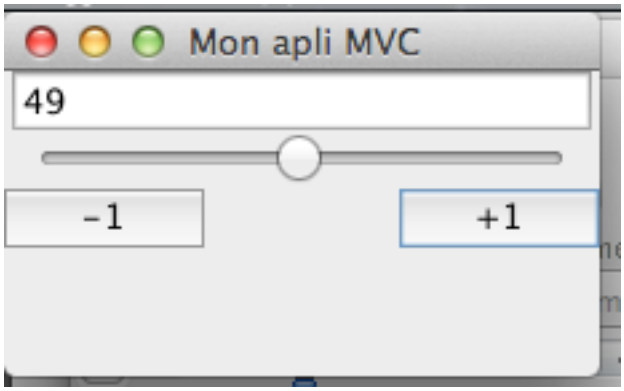
Examen – janvier 2015 – session 1 – durée : 2h

Documents autorisés : les polycopiés du cours **(pas de TD/TP)**

Partie A : Programmation graphique *(environ 10 points)*

L'objectif est de créer une interface permettant le contrôle d'une température en degrés Celsius.

L'interface se compose de deux vues et deux contrôleurs. Une vue est sous forme d'un texte, et l'autre vue est sous forme d'un Slider. Un contrôleur est sous forme d'un Slider, et l'autre est sous forme de deux boutons.



Concevoir une application avec MVC.

Question A.1 :
Faisant le lien avec cet exercise, expliquer le Modele, la Vue et le Contrôleur?

Question A.2 :

1, Nous vous donnons la classe de l'application où il y a la fonction main() à completer.

```
public class MonAppliMVC {
  public static void main(String[] args) {
    SwingUtilities.invokeLater(
      new Runnable() {
        public void run() {
            JFrame frame = new JFrame("Mon apli MVC");
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            JPanel panel = new JPanel();
            panel.setLayout(new GridLayout(5,1));
            // à compléter : créer le MVC et leur liens
            frame.pack();
           frame.setVisible(true);
        }
      });
  }
}
```

2, Pour Slider, nous donnons son écouteur à compléter comme le suivant

```
this.addChangeListener(new ChangeListener() {
      public void stateChanged(ChangeEvent e) {
       // à compléter
     }} );
```

et une méthode setValue(t) pour changer le marque du Slider.

3, Nous vous donnons encore la classe de contrôleur correspondant à deux boutons à completer:

```
public class ControlePlusMoins extends JPanel {
        private Modele modele;

        public ControlePlusMoins(Modele m) {
        super(new BorderLayout());
                modele=m;
        JButton PlusButton = new JButton(" +1 ");
        PlusButton.addActionListener(new ActionListener() {
        // à completer
        }} );

        JButton MoinsButton = new JButton(" -1 ");
        MoinsButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e){
        // à completer
        }} );

        this.add(PlusButton, BorderLayout.EAST);
        this.add(MoinsButton, BorderLayout.WEST);
        }
}
```

4, Coder les autres classes correspondantes au MVC.

Partie B : Généricité, Exception et Thread *(environ 10 points)*

Voici la classe Sac indispensable au jeu de Loto :
Un sac contient des jetons pour le tirage. On peut tirer aléatoirement des jetons du sac, voire en ajouter dans la limite d'une taille maximale.

```
import java.util.ArrayList;
import java.util.Collection;
import java.util.List;

public class Sac {
        private List liste = new ArrayList();
        private int tailleMax = 0;

        public Sac(Collection coll){
                liste = new ArrayList(coll);
                this.tailleMax = coll.size();
        }

        public int getTailleMax() {
                return tailleMax;
```

```
	}

	public int getQuantite() {
		return liste.size();
	}

	public void ajouter(Object jeton) {
		if ((jeton != null) && (liste.size() < tailleMax))
			liste.add(element);
	}

	public Object tirer() {
		if (liste.size() > 0)
			return liste.remove((int)(Math.random()*liste.size()));
		else
			return null;
	}
}
```

Voici un petit programme de Loto :
Les jetons sont içi des String numérotés de 1 à 6

```
import java.util.Arrays;
import java.util.Scanner;

public class Loto {
	public static void main(String args[]) {
		Sac sac = new Sac(Arrays.asList("1", "2", "3", "4", "5", "6"));
		String tirage;
		System.out.println("q(uitter), t(irer), a(jouter), r(este combien),
		   m(ax taille du sac)");
		Scanner sc = new Scanner(System.in);
		String rep;
		do {
			rep = sc.next().trim();
			if (rep.equals("t"))
				try {
					tirage = (String) sac.tirer();
					System.out.println("tirage  : "+ tirage);
				} catch (ClassCastException cce) {
						System.out.println("erreur de sorte de jeton");
		      }
			else if (rep.equals("r"))
				System.out.println("reste " + sac.getQuantite());
			else if (rep.equals("t"))
				System.out.println("reste " + sac.getTailleMax());
			else if (rep.equals("a")) {
				rep = sc.next().trim();
				sac.ajouter(rep);
				rep = "";
			}
		} while (! rep.equals("q")) ;
	}
}
```

Notre implémentation du sac est médiocre. Toutes les questions qui suivent peuvent être regroupées dans une seule et même classe.

Question A :
Améliorons-la pour que le sac ne contienne que des jetons du même type, donc qu'il n'y ait pas besoin de faire un "cast" comme dans notre loto.

Question B :
Les méthodes ajouter() et tirer() devraient lever des exception IllegalStateException quand le tirage se fait avec un sac vide ou l'ajout de jeton sur un sac plein.

Question C :
Pour parfaire en beauté cette nouvelle implémentation de Sac, rendons-la "clonable" (en surface seulement) et "sérialisable" simplement.

Que se passe t'il si plusieurs joueurs jouent au Loto ? Dans la vie courante, deux joueurs n'arrivent pas à tirer un jeton du sac en même temps.
Malheureusement, il n'en va pas de même pour notre implémentation informatique de la solution :
voici un test LotoAPlusieurs dans lequel trois joueurs tirent et remettent un jeton 100 fois.
Nous utilisons ici la première version de Sac.

```
import java.util.Arrays;

public class LotoAPlusieurs {
	public static void main(String args[]) {
		new LotoAPlusieurs();
	}
	public LotoAPlusieurs() {
		Sac sac = new Sac(Arrays.asList("1", "2", "3", "4", "5", "6"));
		Thread [] lesJoueurs = new Thread[3];
		lesJoueurs[0]  = new Thread(new Joueur("A", sac));
		lesJoueurs[1]  = new Thread(new Joueur("B", sac));
		lesJoueurs[2]  = new Thread(new Joueur("C", sac));
		for (int i = 0 ; i < 3 ; i++ )
			lesJoueurs[i].start();
	}
	public class Joueur implements Runnable {
		private String nom;
		private Sac sac;
		public Joueur(String nom, Sac sac) {
			this.nom = nom;
			this.sac = sac;
		}
		public void run() {
			for (int i = 0 ; i < 100 ; i++ )
				try {
					String tirage = (String) sac.tirer();
					System.out.println("le joueur "+ nom + " tire "+ tirage);
					sac.ajouter(tirage);
				} catch (ClassCastException cce) {
					System.out.println("erreur de sorte de jeton");
				}
		}
	}
}
```

Question D :
Améliorez la classe Sac pour qu'elle soit "thread-safe, donc de telle façon qu'un tirage ou un ajout se fasse en exclusion mutuelle.

Question E :
Améliorez aussi la classe Joueur de test LotoAPlusieurs pour que le tirage d'un jeton et sa remise dans le sac se fasse en exclusion mutuelle des autres joueurs : si le joueur A tire un jeton X, il faut qu'il puisse le remettre dans le sac avant qu'un autre joueur fasse un tirage.