TP 5 : Serialisation

On veut maintenant que l'état de notre chronomètre puisse être sauvegardé quand on le ferme et restitué quand on le lance.

On va donc définir une nouvelle classe "EtatChronometre" qui contiendra les informations relatives à l'état du chronomètre lorsqu'on termine l'application à savoir :

- Les valeurs de la mesures actuellement affichée (minutes et secondes ou secondes et 1/100)
- La valeur de la mesure intermédiaire actuellement affichée
- La vitesse de chronométrage actuellement choisie (à la seconde ou au 1/100 de seconde)

Cette classe sera sérialisée dans un fichier lorsque l'on quittera l'application. Quand on lancera l'application,si ce fichier existe, le chronomètre sera remis dans son état antérieur.

1°) Ecrire la classe "EtatChronometre" qui doit être sérialisable, doit avoir un constructeur permettant d'initialiser toutes ses propriétés et doit avoir des méthodes get et set pour chacune de ses propriétés.

2°) Modifier l'action associée à la fermeture de la fenêtre pour qu'avant de terminer le programme elle crée un objet de classe "EtatChronometre" représentant l'état actuel du chronomètre et lesérialise dans un fichier de nom "chronometre.ser".

3°) Modifier le constructeur de l'interface du chronomètre de façon à ce que :

- S'il existe un fichier "chronometre.ser" l'état initial du chronomètre soit celui décrit dans ce fichier ;
- Sinon l'état initial du chronomètre sera celui par défaut c'est à dire comme c'était jusqu'à présent.

ATTENTION : n'oubliez pas que l'action de l'écouteur d'événements associé à un widget est déclenchée aussi bien par une action de l'utilisateur que par une action du programme. Ainsi si le programme modifie le choix dans la liste déroulante après qu'un 'écouteur d'événements ait été associé à cette liste son action sera exécutée. Ce qui veut dire ici que les étiquettes seront modifiées pour correspondre aux unités de chronométrage choisi (ce qui est bien) mais que les valeurs de chronométrage seront remises à zéro (ce qui est moins bien puisqu'on veut retrouver celles qui ont été sauvegardées).

Vérifier que la sauvegarde/restitution fonctionne correctement et que l'on retrouve bien l'état complet du chronomètre.

SOLUTION

```java
package tp5;

import java.io.Serializable;

public class EtatChronometre implements Serializable {

	private static final long serialVersionUID = 1L;
	private String valeur1, valeur2, valeurInt;
	private boolean minSec;

	public EtatChronometre(String v1, String v2, String vi, boolean mod) {
		valeur1  = v1;
		valeur2  = v2;
		valeurInt  = vi;
		minSec = mod;
	}

	public void setValeur1(String v1) { valeur1  = v1; }
	public String getValeur1() { return valeur1; }

	public void setValeur2(String v2) { valeur2  = v2; }
	public String getValeur2() { return valeur2; }

	public void setValeurInt(String vi) { valeurInt  = vi; }
	public String getValeurInt() { return valeurInt; }

	public void setminSec(boolean mod) { minSec=mod; }
	public boolean getMinSec() { return minSec; }

}
```

```java
package tp5;

import javax.swing.JFrame;
import java.awt.BorderLayout;
import javax.swing.JPanel;
import java.awt.FlowLayout;
import javax.swing.JTextField;
import javax.swing.JLabel;
import java.awt.GridLayout;
import javax.swing.JButton;
import javax.swing.JComboBox;
import java.awt.Font;
import java.awt.Color;
import javax.swing.SwingConstants;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
```

```
import java.io.OptionalDataException;

public class Chronometre extends JFrame {
   private JTextField minutes;
   private JTextField secondes;
   private JTextField tempsIntermediaire;
   private JButton start, stop, capture, reset;
   private JComboBox<String> mode;
   private JLabel unite1, unite2;
   private boolean minsec;
   private Moteur moteur;

   public Chronometre() {
      super("Chronomètre");
      addWindowListener(new FermetureFenetre());
      minsec=true;
      getContentPane().setLayout(new BorderLayout(0, 0));

      JPanel panel = new JPanel();
      getContentPane().add(panel, BorderLayout.CENTER);
      FlowLayout fl_panel = new FlowLayout(FlowLayout.CENTER, 10, 15);
      panel.setLayout(fl_panel);

      minutes = new JTextField();
      minutes.setToolTipText("Minutes ou secondes");
      minutes.setHorizontalAlignment(SwingConstants.CENTER);
      minutes.setText("00");
      minutes.setForeground(Color.RED);
      minutes.setBackground(Color.WHITE);
      minutes.setFont(new Font("Tahoma", Font.PLAIN, 36));
      minutes.setEditable(false);
      panel.add(minutes);
      minutes.setColumns(2);

      unite1 = new JLabel("Min");
      panel.add(unite1);

      secondes = new JTextField();
      secondes.setToolTipText("Secondes ou 1/100 de seconde");
      secondes.setText("00");
      secondes.setHorizontalAlignment(SwingConstants.CENTER);
      secondes.setForeground(Color.RED);
      secondes.setBackground(Color.WHITE);
      secondes.setEditable(false);
      secondes.setFont(new Font("Tahoma", Font.PLAIN, 36));
      panel.add(secondes);
      secondes.setColumns(2);

      unite2 = new JLabel("Sec  ");
      panel.add(unite2);

      JPanel panel_1 = new JPanel();
      getContentPane().add(panel_1, BorderLayout.EAST);
      panel_1.setLayout(new GridLayout(3, 1, 1, 4));

      start = new JButton("Start");
      start.addActionListener(new ActionStart());
      start.setToolTipText("Lancer le chronom\u00E8tre");
      panel_1.add(start);
```

```
stop = new JButton("Stop");
stop.setEnabled(false);
stop.addActionListener(new ActionStop());
stop.setToolTipText("Arr\u00EAter le chronom\u00E8tre");
panel_1.add(stop);

reset = new JButton("Reset");
reset.addActionListener(new ActionReset());
reset.setToolTipText("Remise \u00E0 z\u00E9ro du chronom\u00E8tre");
panel_1.add(reset);

JPanel panel_2 = new JPanel();
getContentPane().add(panel_2, BorderLayout.SOUTH);
FlowLayout fl_panel_2 = new FlowLayout(FlowLayout.CENTER, 10, 5);
panel_2.setLayout(fl_panel_2);

tempsIntermediaire = new JTextField();
tempsIntermediaire.setToolTipText("Temps interm\u00E9diaire");
tempsIntermediaire.setText("00:00");
tempsIntermediaire.setHorizontalAlignment(SwingConstants.CENTER);
tempsIntermediaire.setBackground(Color.WHITE);
tempsIntermediaire.setEditable(false);
tempsIntermediaire.setFont(new Font("Tahoma", Font.PLAIN, 20));
panel_2.add(tempsIntermediaire);
tempsIntermediaire.setColumns(5);

capture = new JButton("Capture");
capture.addActionListener(new ActionCapture());
capture.setToolTipText("Capture du temps interm\u00E9diaire");
panel_2.add(capture);

mode = new JComboBox<String>();
mode.addActionListener(new ActionChoix());
mode.setToolTipText("Choix du mode (min:sec ou sec:1/100)");
mode.setFont(new Font("Tahoma", Font.ITALIC, 13));
mode.addItem("Min:Sec");
mode.addItem("Sec:1/100");
panel_2.add(mode);

EtatChronometre sauve = null;
boolean lu=false;
File lit=new File("chronometre.ser"); // fichier de sauvegarde
ObjectInputStream entree=null;
try {
   entree= new ObjectInputStream(new FileInputStream(lit));
   sauve=(EtatChronometre) entree.readObject();
   lu=true;
   entree.close();
   }
catch (FileNotFoundException fnf) {
   System.out.println("Fichier de sauvegarde introuvable");
   }
catch (ClassNotFoundException cnf){
   System.out.println("Lecture des objets impossible : erreur de classe");
   }
catch (OptionalDataException ode){
   System.out.println("Lecture des objets impossible : erreur de données");
   }
```

```
   catch (IOException io1) {
      tem.out.println("Création/lecture ou fermeture de flux impossible");
      }
   if (lu) {
      if (sauve.getMinSec()) mode.setSelectedIndex(0);
      else mode.setSelectedIndex(1);
      minutes.setText(sauve.getValeur1());
      secondes.setText(sauve.getValeur2());
      tempsIntermediaire.setText(sauve.getValeurInt());
   }

   pack();
   setVisible(true);
}

private class FermetureFenetre extends WindowAdapter {
   @Override
   public void windowClosing(WindowEvent e) {
      EtatChronometre sauve = new EtatChronometre(minutes.getText(),
                  secondes.getText(), tempsIntermediaire.getText(), minsec);
      File ecrit=new File("chronometre.ser"); // fichier de sauvegarde
      try { ecrit.createNewFile(); }
      catch (IOException io1) {
         System.out.println("création impossible du fichier d'enregistrement");
         }
      FileOutputStream flot=null;
      try { flot=new FileOutputStream(ecrit); }
      catch (FileNotFoundException fnf) {
         System.out.println("Fichier d'enregistrement introuvable");
         }
      ObjectOutputStream sortie=null;
      try { sortie=new ObjectOutputStream(flot); }
      catch (IOException io2) {
         System.out.println("Création du flot objet impossible");
         }
      try {
         sortie.writeObject(sauve);
         }
      catch (IOException io3) {
         System.out.println("Ecriture des objets impossible");
         }
      try {sortie.flush(); sortie.close(); }
      catch (IOException io4) {
         System.out.println("Fermeture du flot objet impossible");
         }

      System.exit(0); // arrêt du programme

   }
}

private class ActionStart implements ActionListener {
   public void actionPerformed(ActionEvent e) {
      start.setEnabled(false);
      mode.setEnabled(false);
      stop.setEnabled(true);
      moteur=new Moteur(minutes, secondes, !minsec);
      moteur.start();
   }
```

```
   }

   private class ActionStop implements ActionListener {
      public void actionPerformed(ActionEvent e) {
         start.setEnabled(true);
         mode.setEnabled(true);
         stop.setEnabled(false);
         moteur.arreter();
      }
   }

   private class ActionReset implements ActionListener {
      public void actionPerformed(ActionEvent e) {
         minutes.setText("00");
         secondes.setText("00");
         tempsIntermediaire.setText("00:00");
      }
   }

   private class ActionCapture implements ActionListener {
      public void actionPerformed(ActionEvent e) {
         String v1,v2;
         v1 = minutes.getText();
         v2 = secondes.getText();
         tempsIntermediaire.setText(v1+":"+v2);
      }
   }

   private class ActionChoix implements ActionListener  {
      public void actionPerformed(ActionEvent e) {
         switch (mode.getSelectedIndex()) {
         case 0: unite1.setText("Min");
            unite2.setText("Sec");
            minsec=true;
            break;
         case 1: unite1.setText("Sec");
         unite2.setText("1/100");
         minsec=false;
         break;
         }
         minutes.setText("00");
         secondes.setText("00");
      }
   }

   /**
    * @param args
    */
   public static void main(String[] args) {
      // TODO Auto-generated method stub
      new Chronometre();
   }

}
```