TP 6 : Client/Serveur

On veut maintenant que notre chronomètre puisse être commandé à distance.
Par son application l'utilisateur distant peut lancer et arrêter le chronomètre mais il ne voit pas le temps mesuré.
Il peut capturer une valeur intermédiaire qu'il voit s'afficher sur son interface.
Il ne peut pas changer la vitesse de chronométrage ni remettre à zéro le chronomètre.

Son interface est la donc suivante :

ATTENTION : Dans Eclipse vous devrez créer 2 projets, l'un contenant le serveur construit à partir du chronomètre des TPs précédents modifié, l'autre contenant le client.

1°) Le serveur : le chronomètre doit se comporter comme un serveur qui reçoit les commandes de l'application cliente distante. Pour cela il doit, au démarrage, lancer un thread qui ouvre une socket de service sur le numéro de port 4001 et attend de recevoir des commandes. Si la commande reçue correspond au lancement ou à l'arrêt du chronomètre il lance ou arrête le chronomètre exactement comme c'était fait par les boutons "Start" et "Stop" de l'interface du chronomètre. Si la commande reçue correspond à une capture de valeur intermédiaire, il fait cette capture exactement comme c'était fait par le bouton "Capture" de l'interface du chronomètre et en envoie la valeur au client qui l'affichera.

Modifier le chronomètre de façon à ce qu'il remplisse ces fonctionnalités de serveur. Ne pas oublier que le thread assurant le rôle de serveur est lancé dès le démarrage du chronomètre et fonctionne tout le temps puisqu'il peut recevoir des commandes à tout moment. En revanche il doit être arrêté quand l'application du chronomètre l'est.

Vous êtes libre de choisir sous quelle forme la commande est envoyée (une chaîne de caractères, un entier ou un objet de votre création).

2°) Le client : écrire l'application du client qui propose l'interface ci dessus et associe aux 3 boutons les actions appropriées. Lors de l'action sur l'un des boutons, le client devra établir une connexion au serveur et lui envoyer une commande. Si cette commande est une capture de valeur intermédiaire il attendra la réponse du serveur pour afficher la valeur capturée. La connexion est fermée à chaque fin d'action.

Remarque : pour tester plus facilement votre programme vous pouvez faire fonctionner le client et le serveur sur la même machine que vous désignerez par le nom "localhost". Bien entendu si tout fonctionne correctement ça fonctionnera tout aussi bien si le client est réellement distant en mettant le nom ou l'adresse IP de la machine qui accueille le serveur à la place de "localhost".

SOLUTION

1°)

```
package tp6serveur;

import javax.swing.JFrame;
...
import java.io.OptionalDataException;

public class Chronometre extends JFrame {
   private JTextField minutes;
   ...
   private Service serv;

   public Chronometre() {
      super("Chronomètre");
      ...
      pack();
      setVisible(true);
      serv = new Service(this);
      serv.start();
   }

   protected boolean isStartEnabled() { return start.isEnabled(); }

   protected boolean isStopEnabled() { return stop.isEnabled(); }

   protected String getCapture() { return tempsIntermediaire.getText(); }

   protected void actionStart() {
      start.setEnabled(false);
      mode.setEnabled(false);
      stop.setEnabled(true);
      moteur=new Moteur(minutes, secondes, !minsec);
      moteur.start();
   }

   protected void actionStop() {
      start.setEnabled(true);
      mode.setEnabled(true);
      stop.setEnabled(false);
      moteur.arreter();
   }

   protected void actionCapture() {
      String v1,v2;
      v1 = minutes.getText();
      v2 = secondes.getText();
      tempsIntermediaire.setText(v1+":"+v2);
   }

   private class FermetureFenetre extends WindowAdapter {
      @Override
      public void windowClosing(WindowEvent e) {
         serv.arreter();
         ...
         System.exit(0); // arrêt du programme
```

```
      }
   }

   private class ActionStart implements ActionListener {
      public void actionPerformed(ActionEvent e) {
         actionStart();
      }
   }

   private class ActionStop implements ActionListener {
      public void actionPerformed(ActionEvent e) {
         actionStop();
      }
   }

   private class ActionReset implements ActionListener {
      public void actionPerformed(ActionEvent e) {
         ...
      }
   }

   private class ActionCapture implements ActionListener {
      public void actionPerformed(ActionEvent e) {
         actionCapture();
      }
   }

   private class ActionChoix implements ActionListener  {
      public void actionPerformed(ActionEvent e) {
         ...
      }
   }

   public static void main(String[] args) {
      new Chronometre();
   }
}
```

```
package tp6serveur;

import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.net.SocketException;

public class Service extends Thread {
   private Chronometre pere;
   private boolean enMarche;
   private ServerSocket service;
   private final int START=1, STOP=2, CAPTURE=3;

   public Service(Chronometre p) {
      pere = p;
      enMarche = true;
   }

   public void run() {
```

```
      try {
         service = new ServerSocket(4001);
         while(enMarche) {
            Socket lienClient = service.accept();
            ObjectInputStream lire = new
ObjectInputStream(lienClient.getInputStream());
            try {
               Integer demande = (Integer)lire.readObject();
               switch (demande.intValue()) {
               case START:
                  if (pere.isStartEnabled()) {
                      pere.actionStart();
                  }
                  break;
               case STOP :
                  if (pere.isStopEnabled()) {
                      pere.actionStop();
                  }
                  break;
               case CAPTURE :
                  pere.actionCapture();
                  ObjectOutputStream ecrire = new
ObjectOutputStream(lienClient.getOutputStream());
                  ecrire.writeObject(pere.getCapture());
                  ecrire.flush();
                  ecrire.close();
                  break;
               default :
                  System.out.println("Requete incorrecte");
               }
               lire.close();
               lienClient.close();
            }
            catch (ClassNotFoundException e) {
               System.out.println("Requete de type incorrect");
            }
         }
         service.close();
      }
      catch (SocketException se) {
      }
      catch (IOException e) {
         System.out.println("Impossible d assurer le service");
         e.printStackTrace();
      }
   }

   public void arreter() {
      enMarche=false;
      try {
         service.close();
      }
      catch (IOException e) {
         System.out.println("Impossible de fermer le service");
      }
   }

}
```

2°)

```
package tp6client;

import javax.swing.JFrame;
import java.awt.GridLayout;
import javax.swing.JButton;
import javax.swing.JTextField;
import java.awt.Font;
import javax.swing.SwingConstants;
import java.awt.Color;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.Socket;
import java.net.UnknownHostException;

public class CommandeDistante extends JFrame {

   private JTextField tempsIntermediare;
   private JButton start, stop, capture;
   private final int START=1, STOP=2, CAPTURE=3;

   public CommandeDistante() {
      super ("Commande Distante");
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      getContentPane().setLayout(new GridLayout(0, 2, 5, 5));

      start = new JButton("Start");
      start.addActionListener(new ActionStart());
      getContentPane().add(start);

      stop = new JButton("Stop");
      stop.addActionListener(new ActionStop());
      getContentPane().add(stop);

      tempsIntermediare = new JTextField();
      tempsIntermediare.setBackground(Color.WHITE);
      tempsIntermediare.setHorizontalAlignment(SwingConstants.CENTER);
      tempsIntermediare.setEditable(false);
      tempsIntermediare.setText("00:00");
      tempsIntermediare.setFont(new Font("Tahoma", Font.PLAIN, 22));
      getContentPane().add(tempsIntermediare);
      tempsIntermediare.setColumns(5);

      capture = new JButton("Capture");
      capture.addActionListener(new ActionCapture());
      getContentPane().add(capture);

      pack();
      setVisible(true);
   }

   private class ActionStart implements ActionListener {
      public void actionPerformed(ActionEvent e) {
         try {
            Socket lien = new Socket("localhost",4001);
```

```
            ObjectOutputStream ecrire = new
ObjectOutputStream(lien.getOutputStream());
            ecrire.writeObject(new Integer(START));
            ecrire.flush();
            ecrire.close();
            lien.close();
         }
         catch (UnknownHostException e1) {    e1.printStackTrace(); }
         catch (IOException e2) { e2.printStackTrace(); }
      }
   }

   private class ActionStop implements ActionListener {
      public void actionPerformed(ActionEvent e) {
         try {
            Socket lien = new Socket("localhost",4001);
            ObjectOutputStream ecrire = new
ObjectOutputStream(lien.getOutputStream());
            ecrire.writeObject(new Integer(STOP));
            ecrire.flush();
            ecrire.close();
            lien.close();
         }
         catch (UnknownHostException e1) {    }
         catch (IOException e1) { }
      }
   }

   private class ActionCapture implements ActionListener {
      public void actionPerformed(ActionEvent e) {
         try {
            Socket lien = new Socket("localhost",4001);
            ObjectOutputStream ecrire = new
ObjectOutputStream(lien.getOutputStream());
            ecrire.writeObject(new Integer(CAPTURE));
            ecrire.flush();
            ObjectInputStream lire = new ObjectInputStream(lien.getInputStream());
            String rep = (String) lire.readObject();
            tempsIntermediare.setText(rep);
            lire.close();
            ecrire.close();
            lien.close();
         }
         catch (UnknownHostException e1) {    }
         catch (IOException e1) { }
         catch (ClassNotFoundException e2) { }
      }
   }

   public static void main(String[] args) {
      new CommandeDistante();
   }

}
```