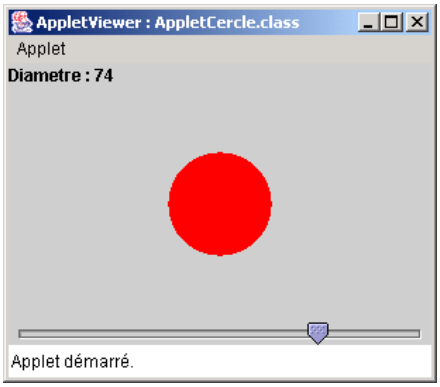
TP JAVA n°9: Les applets en Java

Exercice 1 : Dessin paramétré dans une applet

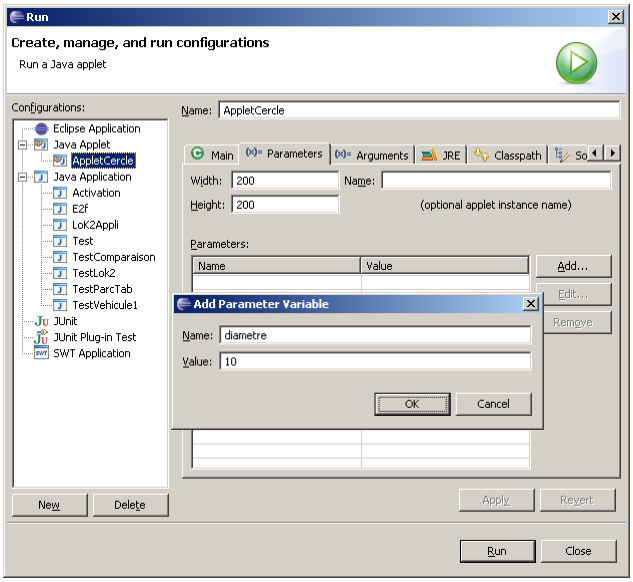
A partir d'Eclipse, la création d'une applet se fait exactement comme n'importe quel projet Java.
Créer un nouveau projet `TestApplet` dans votre workspace.

1. Implémenter l' **`AppletCercle`** vue en TD…

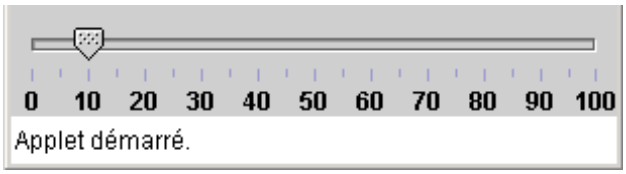


↳ Pour tester, sélectionnez : `Run` → `Run as` → `Java Applet`
et l'applet s'exécute dans l'AppletViewer.
Que vaut le diamètre du cercle à l'exécution et pourquoi ?

↳ Pour passer des paramètres à l'AppletViewer, il faut modifier la configuration d'exécution du projet.
Pour exécuter l'application, au lieu de `Run` → `Run as`, choisir : **`Run`** → **`Run`** …
Ceci ouvre une boîte de dialogue.
En cliquant sur Add vous pouvez choisir le nom du paramètre et sa valeur.
Une fois, la valeur souhaitée entrée, cliquez sur **`Run`**



2. Nous souhaitons maintenant que le slider ait l'apparence suivante.



Pour vous aider, allez consulter la javadoc de la classe `JSlider` sur le site d'Oracle et cliquez sur le lien : **`How to Use Sliders`**

javax.swing
Class JSlider

java.lang.Object
  └ java.awt.Component
    └ java.awt.Container
      └ javax.swing.JComponent
        └ **javax.swing.JSlider**

All Implemented Interfaces:
ImageObserver, MenuContainer, Serializable, Accessible, SwingConstants

---

```
public class JSlider
extends JComponent
implements SwingConstants, Accessible
```

A component that lets the user graphically select a value by sliding a knob within a bounded interval.

The slider can show both major tick marks, and minor tick marks between the major ones. The number of values between the tick marks is controlled with `setMajorTickSpacing` and `setMinorTickSpacing`. Painting of tick marks is controlled by `setPaintTicks`.

Sliders can also print text labels at regular intervals (or at arbitrary locations) along the slider track. Painting of labels is controlled by `setLabelTable` and `setPaintLabels`.

For further information and examples see How to Use Sliders, a section in *The Java Tutorial*.

Warning: Swing is not thread safe. For more information see Swing's Threading Policy

… Profitez du temps qui reste pour travailler sur le projet `cabinetMedical` …

Correction TP JAVA n°9: Les applets en Java

Exercice 1 : Dessin paramétré dans une applet

En vous aidant de la documentation, vous pouvez changer l'apparence du `JSlider` …

```
import java.util.*; // pour hashtable
……
// réglage des dimensions
   diametreSlider = new JSlider (JSlider.HORIZONTAL,0,100,diametre);
   // Représentation Echelle graduée
   diametreSlider.setMajorTickSpacing(10);
   diametreSlider.setMinorTickSpacing(5);
   diametreSlider.setPaintTicks(true);

   Hashtable MesValeurs = new Hashtable();
   MesValeurs = diametreSlider.createStandardLabels(10);
   diametreSlider.setLabelTable(MesValeurs);
   diametreSlider.setPaintLabels(true);
   ////////////////////////////////////////
   add(diametreSlider,BorderLayout.SOUTH);
……
```

↳ JSlider : glissière

Le **`Jslider`** est créé en indiquant une orientation : HORIZONTAL ou VERTICAL. On peut également préciser les valeurs minimum et maximum ainsi qu'une valeur intiale :

```
final JSlider slider = new JSlider (JSlider.HORIZONTAL,0,50,0);
```

Un JSlider comporte des marques de repères, petites lignes tracées à certaines valeurs le long de la zone de défilement. Les repères *majeurs* sont légèrement plus épais que les *mineurs*. Pour tracer des repères, il suffit d'indiquer les intervalles des repères majeurs, puis de les peindre.

```
slider.setMajorTickSpacing(5);
slider.setMinorTickSpacing(1);
slider.setPaintTicks(true);
```

Un JSlider permet aussi de libeller les repères par des chaînes de texte utilisant la méthode **`setLabelTable()`**.
`Le JSLider` déclenche des **`ChangeEvent`** lorsque la valeur change.

1. 3 fichiers pour implémenter cette applet :
   - Fichier : PanelPrincipal.java : dont on rédéfinit la méthode `paintComponent`.
   - Fichier : AppletCercle.java :
   - Fichier de lancement de l'applet: TestAppletCercle.html

Fichier : PanelPrincipal.java :

```
import javax.swing.*;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;

import java.awt.*; //pour les couleurs
import java.util.Hashtable;
public class PanelPrincipal extends JPanel
{
int diametre;
JLabel infoDiametre;
JSlider diametreSlider;
   // Constructeur permettant de donner une valeur au diametre
   // et d'organiser l'IHM
   public PanelPrincipal (int diametre)
   {
   // valeur du diametre
   this.diametre=diametre;

   // JLabel
   setLayout(new BorderLayout());
   infoDiametre = new JLabel("Diametre : " + diametre );
   add(infoDiametre,BorderLayout.NORTH);


   // réglage des dimensions
   diametreSlider = new JSlider (JSlider.HORIZONTAL,0,100,diametre);
   // Représentation Echelle graduée
   diametreSlider.setMajorTickSpacing(10);
   diametreSlider.setMinorTickSpacing(5);
   diametreSlider.setPaintTicks(true);

   Hashtable MesValeurs = new Hashtable();
   MesValeurs = diametreSlider.createStandardLabels(10);
   diametreSlider.setLabelTable(MesValeurs);
   diametreSlider.setPaintLabels(true);
   ////////////////////////////////////////
   add(diametreSlider,BorderLayout.SOUTH);

   // ecouteur d'événement de diamtreSlider
   diametreSlider.addChangeListener(new ChangeListener()
      {
      public void stateChanged(ChangeEvent e)
         {
         setDiametre(diametreSlider.getValue());
         infoDiametre.setText("Diametre : " + getDiametre() ); // diametre
est private !
         repaint();
         }
      });


   }// fin Constructeur

   //Getteur
   public int getDiametre ()
```

```
    { return diametre; }

    //Setteur
    public void setDiametre (int nouvDiametre)
    { this.diametre= nouvDiametre; }

    // Dessin d'un cercle ROUGE plein au milieu du panel
    protected void paintComponent(Graphics g)
    {
    super.paintComponent(g);// Ne pas l'oublier !!!
    g.setColor(Color.RED);
    int xCercleTopLeft = (this.getWidth()/2) - (diametre/2);
// x_coin_haut_gauche - rayon
    int yCercleTopLeft  = (this.getHeight()/2) - (diametre/2);
// y_coin_haut_gauche - rayon

    g.fillOval(xCercleTopLeft ,yCercleTopLeft,diametre,diametre);
    }
}// fin classe PanelPrincipal
```

↳ Fichier : `AppletCercle.java` :

```
import javax.swing.*;

public class AppletCercle extends JApplet
  {
   PanelPrincipal monPanel;

  public void init()
     {
     // Récupération des paramètres
     // Avant car diametre sera passé en paramètre au panel Principal
     // Attention !!! getParameter renvoie un String !!!
     int diametre; // déclarer ici pour pouvoir utiliser dans le catch !!!
     try
         {
        String chDiametre = getParameter("diametre");
        diametre = Integer.parseInt(chDiametre);
        }
   catch (NumberFormatException e) {diametre=50;}
// Exception levée si parametre diametre n'existe pas dans fichier HTML
   // ou si valeur de diametre non compatible avec un format d'entier

     // Instanciation du panel principal
     monPanel = new PanelPrincipal(diametre);
     getContentPane().add(monPanel);
     }

 }
```

↳ Fichier de lancement de l'applet: `TestAppletCercle.html` :

```
<HTML>
<HEAD>
<TITLE> Dessin paramétré dans une applet </TITLE>
</HEAD>
<BODY>
<APPLET
   code  = "AppletCercle.class"
   width = "300"
   height   = "200"
   >
<PARAM NAME="diametre" VALUE="23">
</APPLET>
</BODY>
</HTML>
```