```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 *
 * @author toshiba
 */

    import java.awt.*;
import java.awt.event.*;

public class RoundButton extends Component{

    ActionListener actLis ;
    String titre;
    boolean enfonce = false ;
    Color filling,border;

    public RoundButton (){
    this("");
    enableEvents(AWTEvent.MOUSE_EVENT_MASK);
    }


    public RoundButton(String s){
    this.titre=s;
    enableEvents(AWTEvent.MOUSE_EVENT_MASK);
    }

public RoundButton(Color border,Color filling){
    this.border = border;
    this.filling = filling;
    this.titre="";
    enableEvents(AWTEvent.MOUSE_EVENT_MASK);
    }

    public RoundButton(String s,Color border,Color filling){
    this.border = border;
    this.filling = filling;
    this.titre=s;
    enableEvents(AWTEvent.MOUSE_EVENT_MASK);
    }


    public String getLabel(){
    return titre;
    }

    public void setLabel (String s){
    this.titre=s;
    invalidate();
    repaint();
    }

    public void paint(Graphics g){

    int n = Math.min(getSize().width-1, getSize().height-1);

    // on dessine le cercle
    if(enfonce){
            g.setColor(getColorFilling());
            g.fillOval(0,0,n,n);

    }

    g.setColor(getColorBorder());
    g.drawOval(0,0,n,n);
    g.drawLine(0,0,55,55);

    // puis le label
    Font f = getFont();
    if(f!=null){
        FontMetrics fm = getFontMetrics(getFont());
        g.setColor(getForeground());
        g.drawString(titre, n/2-fm.stringWidth(titre)/2,n/2+fm.getMaxDescent());
    }
    }

public void setColor(Color b,Color f){
border = b;
filling = f;
}

public Color getColorBorder(){
return border;
}

public Color getColorFilling(){
return filling;
}

    public Dimension getPreferredSize(){
    Font f = getFont();
    if (f!= null){
        FontMetrics fm = getFontMetrics(getFont());
        int max=Math.max(fm.stringWidth(titre)+40,fm.getHeight()+40);
        return new Dimension(max,max);
    }
    else return new Dimension (10,10);
    }

    public Dimension detMinimumSize(){
    return new Dimension(10,10);
    }


    public void addActionListener(ActionListener lis){
    // gestion 'manuelle' de l'ajout et suppression d'actionListener sur notre bouton
    // voir doc de AWTEventMulticaster
    actLis = AWTEventMulticaster.add(actLis,lis);
    enableEvents(AWTEvent.MOUSE_EVENT_MASK);
    }
    public void removeActionListener(ActionListener lis){
    actLis=AWTEventMulticaster.remove(actLis,lis);
    }

    public boolean contains(int x, int y){
    int mx = getSize().width/2;
    int my = getSize().height/2;
    return  (((mx-x)*(mx-x)+(my-y)*(my-y))<= mx*mx);
    }

    public void processMouseEvent(MouseEvent e){
    Graphics g;
    switch(e.getID()){
    case MouseEvent.MOUSE_PRESSED :
        enfonce = true;
        repaint();
        break;

    case MouseEvent.MOUSE_RELEASED:
        if(actLis != null)
        actLis.actionPerformed(new ActionEvent(this,ActionEvent.ACTION_PERFORMED,titre));
        if(enfonce){
        enfonce=false;
        repaint();
        }
        break;

    case MouseEvent.MOUSE_ENTERED:
        break;

    case MouseEvent.MOUSE_EXITED:
        if(enfonce){
        enfonce=false;
        repaint();
        }
        break;
    }

    super.processMouseEvent(e);
    }
}
```