



## Classe complète

Voici la classe complète :

```
public class TristateCheckBox extends JCheckBox {
    private static final long serialVersionUID = 1L;

    /** Enumeration des états possible pour notre case à cocher. */
    public static enum StateTristate {
        /** Aucune selection. */
        NOT_SELECTED(false, false, false),
        /** Selection total. */
        SELECTED(false, false, true),
        /** Selection partielle. */
        DONT_CARE(true, true, true);

        /**
         * Constructeur privée de l'énumération
         * @param armed Valeur de la propriété armed
         * @param pressed Valeur de la propriété pressed
         * @param selected Valeur de la propriété selected
         */
        private StateTristate(boolean armed, boolean pressed, boolean selected) {
            this.armed = armed;
            this.pressed = pressed;
            this.selected = selected;
        }

        /** Valeur que prendra la propriété armed du ButtonModel lorsque ce status sera
        sélectionné. */
        private boolean armed;
        /** Valeur que prendra la propriété pressed du ButtonModel lorsque ce status
        sera sélectionné. */
        private boolean pressed;
        /** Valeur que prendra la propriété selected du ButtonModel lorsque ce status
        sera sélectionné. */
        private boolean selected;

        /** @return the armed */
        public boolean isArmed() { return armed; }
        /** @return the pressed */
        public boolean isPressed() { return pressed; }
    }
}
```



```
/** @return the selected */
public boolean isSelected() { return selected; }

/**
 * Definition du niveau suivant par rapport au statut actuel.
 * @return La valeur suivante du statut actuel.
 */
public StateTristate nextState() {
    if(this==NOT_SELECTED) {
        return DONT_CARE;
    } else if(this == SELECTED) {
        return NOT_SELECTED;
    } else {
        return SELECTED;
    }
}
};

/** Modèle de bouton de notre case à cocher à trois niveaux. */
private final TristateDecorator model;

/**
 * Constructeur de notre case à cocher.
 * @param text Texte initial
 * @param icon Icon optionnelle
 * @param initial Etat initial de la case à cocher
 */
public TristateCheckBox(String text, Icon icon, StateTristate initial) {
    super(text, icon);
    // Ajout d'un écouteur lorsque la souris est actionnée
    super.addMouseListener(new MouseAdapter() {
        @Override
        public void mousePressed(MouseEvent e) {
            grabFocus();
            model.nextState();
        }
    });
    // Enregistrement du nouveau modèle
    model = new TristateDecorator(getModel());
    setModel(model);
    setState(initial);
}
/**
 * Constructeur de notre case à cocher.
 * @param text Texte initial
 * @param initial Etat initial de la case à cocher
 */
```



```
*/
public TristateCheckBox(String text, StateTristate initial) {
    this(text, null, initial);
}
/**
 * Constructeur de notre case à cocher.
 * @param text Texte initial
 */
public TristateCheckBox(String text) {
    this(text, StateTristate.DONT_CARE);
}
/**
 * Constructeur de notre case à cocher.
 */
public TristateCheckBox() {
    this(null);
}

/**
 * Désactivation des événements de souris sur le composant,
 * dans le but que swing ne puisse plus interagir avec le composant.
 */
@Override
public void addMouseListener(MouseListener l) { }

/**
 * Enregistre le nouvel état de la case à cocher.
 * @param state nouvel état du composant.
 */
public void setState(StateTristate state) {
    if(state != null)
        model.setState(state);
}
/**
 * Donne l'état courant de la case à cocher.
 * @return L'état de la case à cocher.
 */
public StateTristate getState() {
    return model.getState();
}

/**
 * Surcharge de la méthode {@link JCheckBox#setSelected(boolean)} afin de
 * redéfinir les états.
 * @param selected Indique si la case est cochée.
```



```
*/
@Override
public void setSelected(boolean selected) {
    if (selected) {
        setState(StateTristate.SELECTED);
    } else {
        setState(StateTristate.NOT_SELECTED);
    }
}

/**
 * Modèle spécifique pour notre case à cocher.
 */
private class TristateDecorator implements ButtonModel {
    /** Modèle par défaut. */
    private final ButtonModel other;
    /** Etat de notre case à cocher */
    private StateTristate state;

    /**
     * Constructeur du modèle.
     * @param other Le modèle par défaut.
     */
    private TristateDecorator(ButtonModel other) {
        this.other = other;
    }

    /**
     * Enregistre le nouvel état de la case à cocher.
     * @param state nouvel état du composant.
     */
    private void setState(StateTristate state) {
        this.state = state;
        other.setArmed(state.isArmed());
        setPressed(state.isPressed());
        setSelected(state.isSelected());
    }

    /**
     * Donne l'état courant de la case à cocher.
     * @return L'état de la case à cocher.
     */
    private StateTristate getState() {
        return state;
    }
}

/**
```



```
* Passe l'état de la case à cocher dans l'état suivant de l'état courant.
*/
private void nextState() {
    setState(getState().nextState());
}

armed. */
/** Surcharge pour ne plus prendre en compte le changement de la propriété
public void setArmed(boolean b) {}

enabled. */
/** Surcharge pour ne plus prendre en compte le changement de la propriété
public void setEnabled(boolean b) {
    setFocusable(b);
    other.setEnabled(b);
}

// Toutes les méthodes sont simplement déléguées au modèle par défaut,
// Mise à part armed, selected et pressed qui sont données par l'état courant.
public boolean isArmed() { return state.isArmed(); }
public boolean isSelected() { return state.isSelected(); }
public boolean isEnabled() { return other.isEnabled(); }
public boolean isPressed() { return state.isPressed(); }
public boolean isRollover() { return other.isRollover(); }
public void setSelected(boolean b) { other.setSelected(b); }
public void setPressed(boolean b) { other.setPressed(b); }
public void setRollover(boolean b) { other.setRollover(b); }
public void setMnemonic(int key) { other.setMnemonic(key); }
public int getMnemonic() { return other.getMnemonic(); }
public void setActionCommand(String s) { other.setActionCommand(s); }
public String getActionCommand() { return other.getActionCommand(); }
public void setGroup(ButtonGroup group) { other.setGroup(group); }
public void addActionListener(ActionListener l) { other.addActionListener(l); }
public void removeActionListener(ActionListener l) {
other.removeActionListener(l); }
public void addItemListener(ItemListener l) { other.addItemListener(l); }
public void removeItemListener(ItemListener l) {
other.removeItemListener(l); }
public void addChangeListener(ChangeListener l) {
other.addChangeListener(l); }
public void removeChangeListener(ChangeListener l) {
other.removeChangeListener(l); }
public Object[] getSelectedObjects() { return other.getSelectedObjects(); }
}

/**
* Méthode de test de notre composant.
```



## Hamel Ultra Développement

```
* Affichage d'une case à cocher à trois niveaux  
* et d'une case à cocher normale par LookAndFeel disponibles.  
* @param args Aucun arguments utilisés  
* @throws Exception  
*/
```

```
public static void main(String args[]) throws Exception {  
    SwingUtilities.invokeLater(new Runnable() {  
        @Override  
        public void run() {  
            JFrame frame = new JFrame("Test de TristateCheckBox");  
            LookAndFeelInfo[] lookAndFeelInfos =  
            UIManager.getInstalledLookAndFeels();  
            frame.getContentPane().setLayout(new GridLayout(0, 1, 5, 5));  
            try {  
                for(LookAndFeelInfo lookAndFeelInfo : lookAndFeelInfos)  
{  
                    lookAndFeelInfo.getName();  
  
                    UIManager.setLookAndFeel(lookAndFeelInfo.getClassName());  
                    JPanel panel = new JPanel(new  
                    GridLayout(2,1));  
  
                    panel.setBorder(BorderFactory.createTitledBorder(lookAndFeelInfo.getName()));  
                    panel.add(new TristateCheckBox("La case à  
                    cocher à trois niveaux", StateTristate.DONT_CARE));  
                    panel.add(new JCheckBox("La case à cocher  
                    normale", true));  
  
                    frame.getContentPane().add(panel);  
                }  
            } catch (Exception e) {  
                e.printStackTrace();  
            }  
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
            frame.pack();  
            frame.setLocationRelativeTo(null);  
            frame.setVisible(true);  
        }  
    });  
}
```