TRAVAUX PRATIQUES

WindowBuilder (ECLIPSE) et le modèle MVC

Lien pour le téléchagement d'Eclipse
https://eclipse.org/downloads/

Liens WindowBuilder
https://eclipse.org/windowbuilder/
http://www.eclipse.org/windowbuilder/download.php

Tutoriel simple pour l'installation de WindowBuilder à partir d'Eclipse
http://vanhouteghem-jonathan.fr/2015/09/eclipse-installation-de-windowbuilder/

Documentation sur les bibliothèques Java
http://docs.oracle.com/javase/6/docs/api/

Forum d'aide pour Eclipse
http://eclipse.developpez.com/

TP WindowBuilder

L'objectif de ce TP est double : le premier consiste à prendre en main WindowBuilder (un plug'in d'Eclipse) pour construire rapidement des interfaces graphiques ; le second est de concevoir et développer une application en respectant une modélisation MVC (Modèle – Vue – Contrôleur).

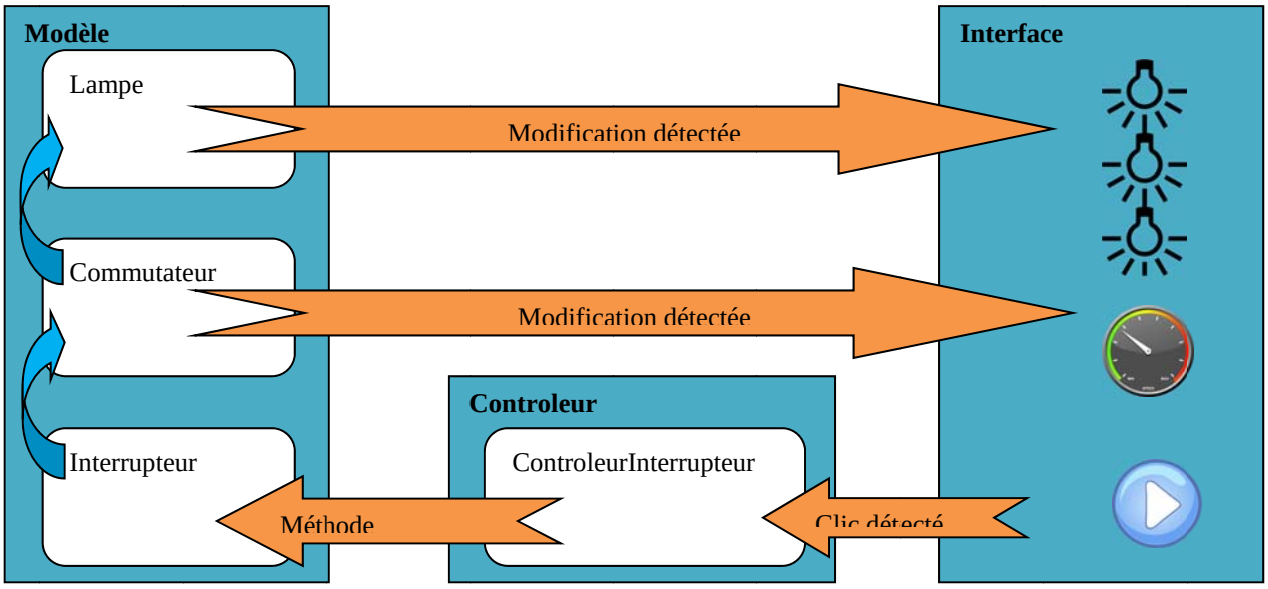
1- Principes et fonctionnement général

Pour mener à bien ce TP nous reprendrons nos 3 classes Java *Lampe*, *Commutateur* et *Interrupteur* déjà développées. Ces trois classes forment le ***Modèle***.

Nous allons y associer une interface graphique pour visualiser les trois lampes, le commutateur et l'interrupteur. Cette interface graphique représente la ***Vue***.

Et enfin, nous construirons une classe *ControleurInterrupteur*. Cette classe représentera le ***Contrôleur***. Elle fera le lien entre la ***Vue*** et le ***Modèle***. Une action de l'utilisateur sur la ***Vue*** est perçue par le ***Contrôleur*** (un clic de bouton par exemple), vérifié puis transmis au ***Modèle***. Lorsqu'un objet du modèle est modifié (une lampe s'allume par exemple) la ***Vue*** devra percevoir ce changement et le répercuter graphiquement.

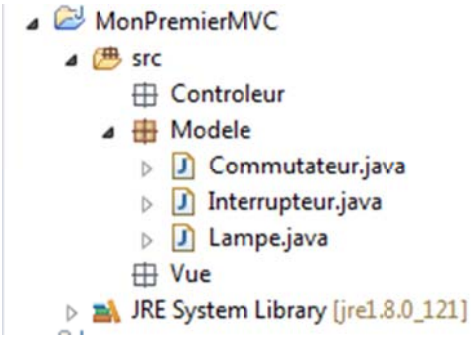
Ainsi, nous pouvons représenter la structure que nous voulons réaliser par le schéma ci-dessous.



2- Premier prototype

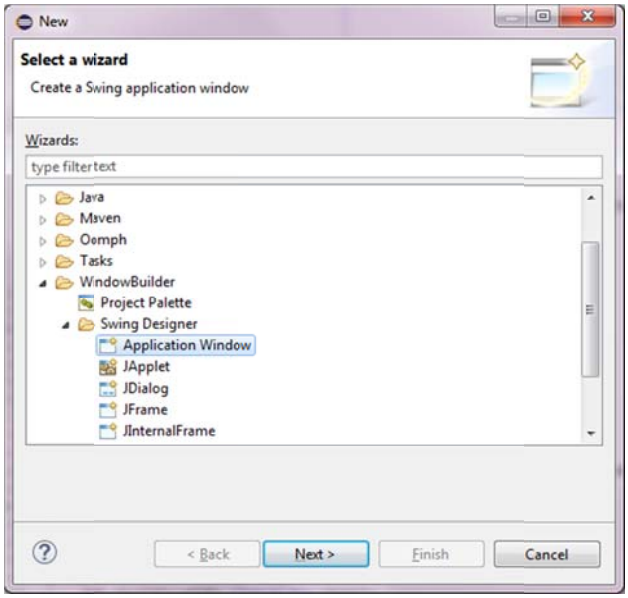
Créer un projet Java nommé *MonPremierMVC* par exemple.
Créer 3 packages : *Modele*, *Vue* et *Controleur*
Dans le package *Modele*, recopier les 3 classes *Lampe*, *Commutateur* et *Interrupteur* déjà écrites lors des séances de TD précédentes.

Vous devez obtenir la structure suivante :

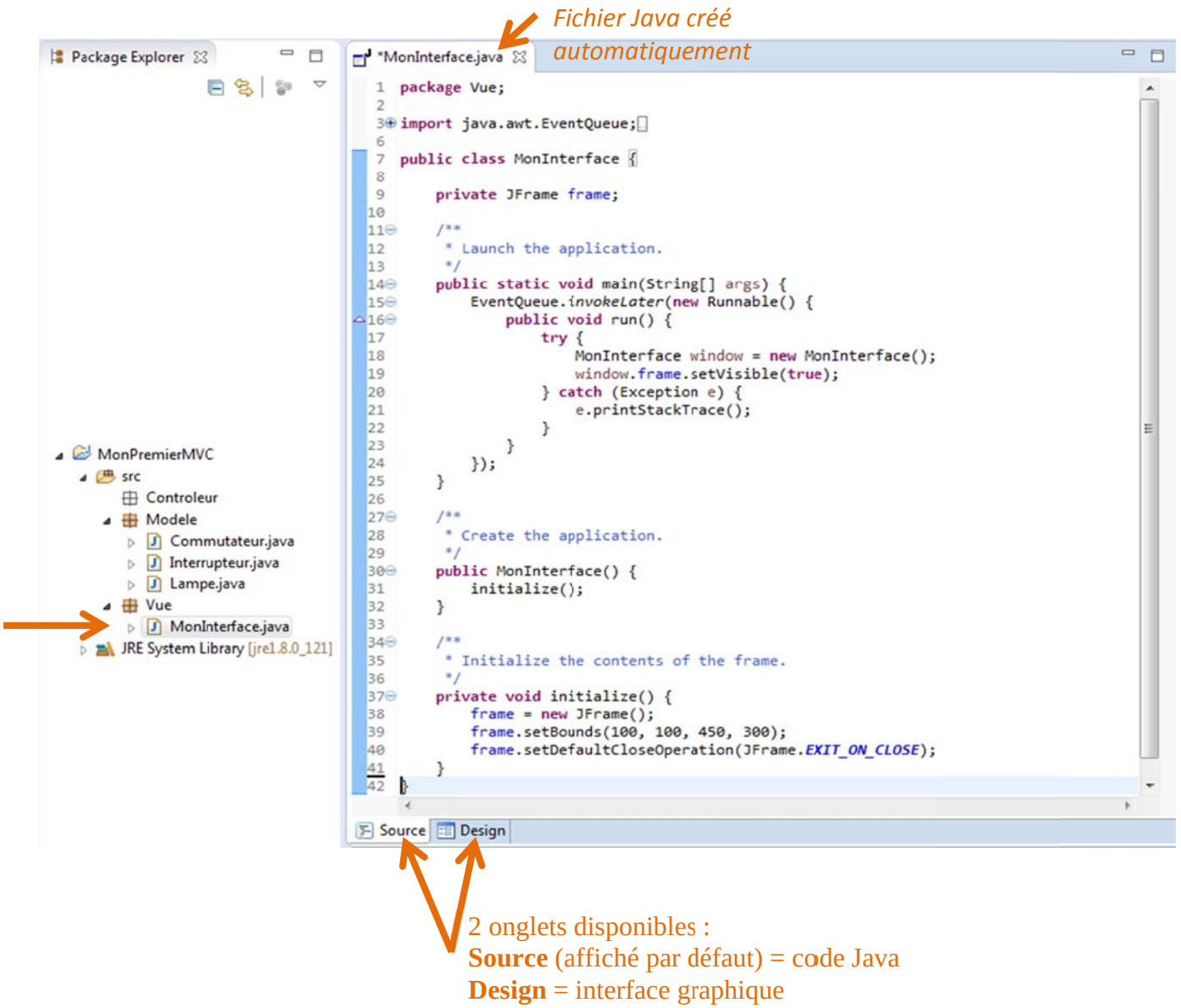


Modifier la classe Lampe en ajoutant une nouvelle propriété *numero* qui contiendra la position de la lampe dans le tableau des lampes du commutateur. La première lampe aura le numéro 0, la seconde 1, etc.
Vous développerez également un getter qui renvoie ce numéro.

Nous allons maintenant créer l'interface dans le package *Vue*.
Faire un clic droit sur le package *Vue*, puis dans le menu surgissant, faites *New – Other…*
Dans la fenêtre qui va s'ouvrir, dans le répertoire *WindowBuilder* choisir *Application Window*

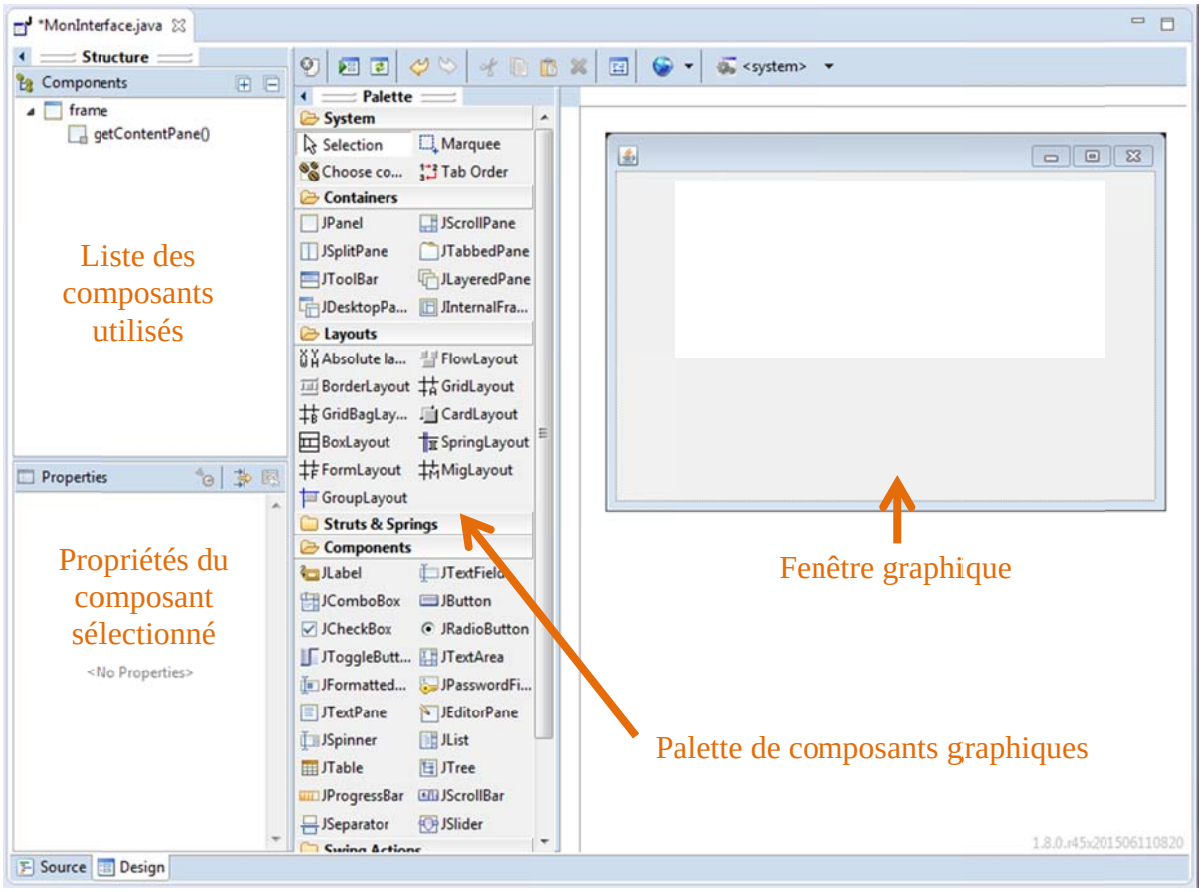


Donnez ensuite un nom à votre fenêtre Window : par exemple *MonInterface*
Après quelques secondes *Eclipse* crée un fichier *MonInterface.java* contenant le code de l'affichage d'une fenêtre vide.




```
1  package Vue;
2
3  import java.awt.EventQueue;
6
7  public class MonInterface {
8
9      private JFrame frame;
10
11     /**
12      * Launch the application.
13      */
14     public static void main(String[] args) {
15         EventQueue.invokeLater(new Runnable() {
16             public void run() {
17                 try {
18                     MonInterface window = new MonInterface();
19                     window.frame.setVisible(true);
20                 } catch (Exception e) {
21                     e.printStackTrace();
22                 }
23             }
24         });
25     }
26
27     /**
28      * Create the application.
29      */
30     public MonInterface() {
31         initialize();
32     }
33
34     /**
35      * Initialize the contents of the frame.
36      */
37     private void initialize() {
38         frame = new JFrame();
39         frame.setBounds(100, 100, 450, 300);
40         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
41     }
42 }
```

Sélectionnez l'onglet *Design* pour voir le dessin de la fenêtre



Fenêtre vide

Exécutez le programme et vérifiez que l'application avec une interface « vide » fonctionne bien.

3- Personnalisation de la première fenêtre

Pour commencer, nous allons placer un *Layout* sur notre fenêtre. Un *layout* sert à gérer la position des éléments sur la fenêtre. Comme vous pouvez le voir, il en existe plusieurs. Nous allons utiliser le *Absolute layout* qui permet de placer (et déplacer) un objet n'importe où sur la fenêtre sans contrainte particulière. Vous pouvez avoir un aperçu des différents *layouts* à l'adresse suivante https://openclassrooms.com/courses/apprenez-a-programmer-en-java/positionner-des-boutons

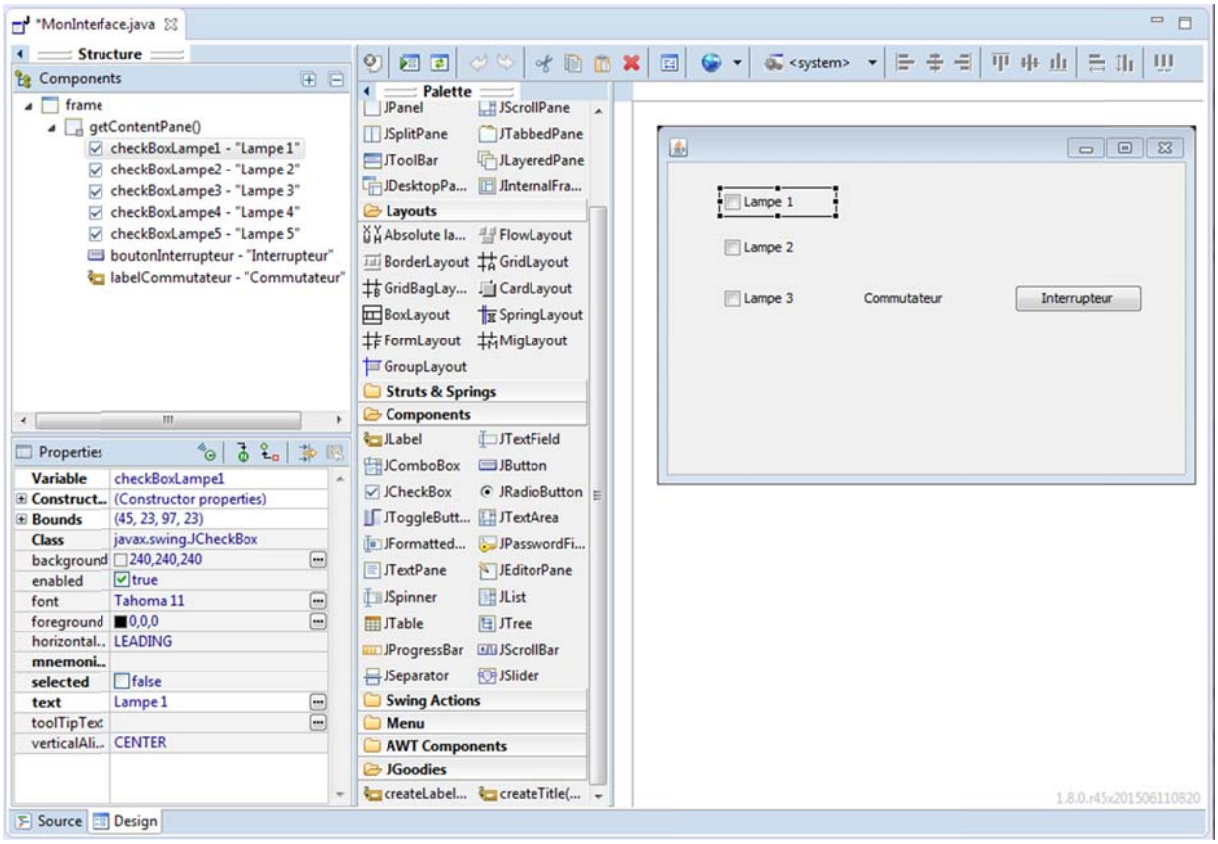
Cliquez sur le composant *Absolute layout* de la palette des composants puis cliquer dans la fenêtre graphique.

Vous pouvez maintenant placer les objets graphiques sur la fenêtre. Vous pouvez également modifier les propriétés d'un objet graphique en le sélectionnant avec la souris et en agissant dans la zone *Properties*.

Nous allons réaliser une interface simple :

- Placer 3 cases à cocher pour les 3 lampes

- Placer 1 label pour voir l'état du commutateur
- Placer 1 bouton pour l'interrupteur
- Donner à ces objets graphiques des noms parlants, par exemple *checkboxLampe1*, *checkboxLampe2*, etc.



Relancer l'application pour vérifier que vos objets graphiques s'affichent bien à l'exécution.

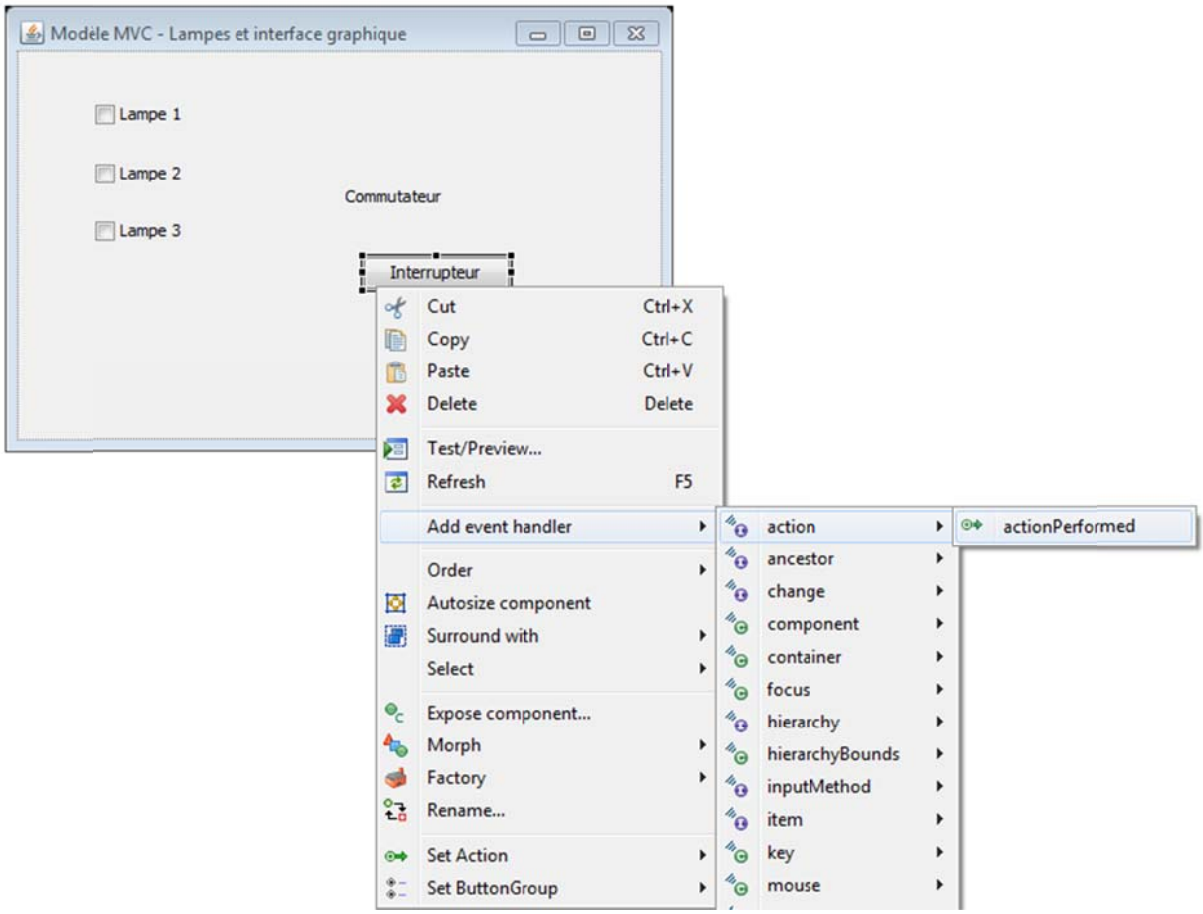
4- Mise en place du Contrôleur

Nous allons maintenant mettre en place le *Controleur* qui gère la communication entre l'objet graphique *boutonInterrupteur* (de la *Vue*) et l'objet *interrupteur* du *Modèle*. Dans le package *Controleur*, créez la classe *ControleurInterrupteur* pour gérer le clic du bouton et actionner l'interrupteur

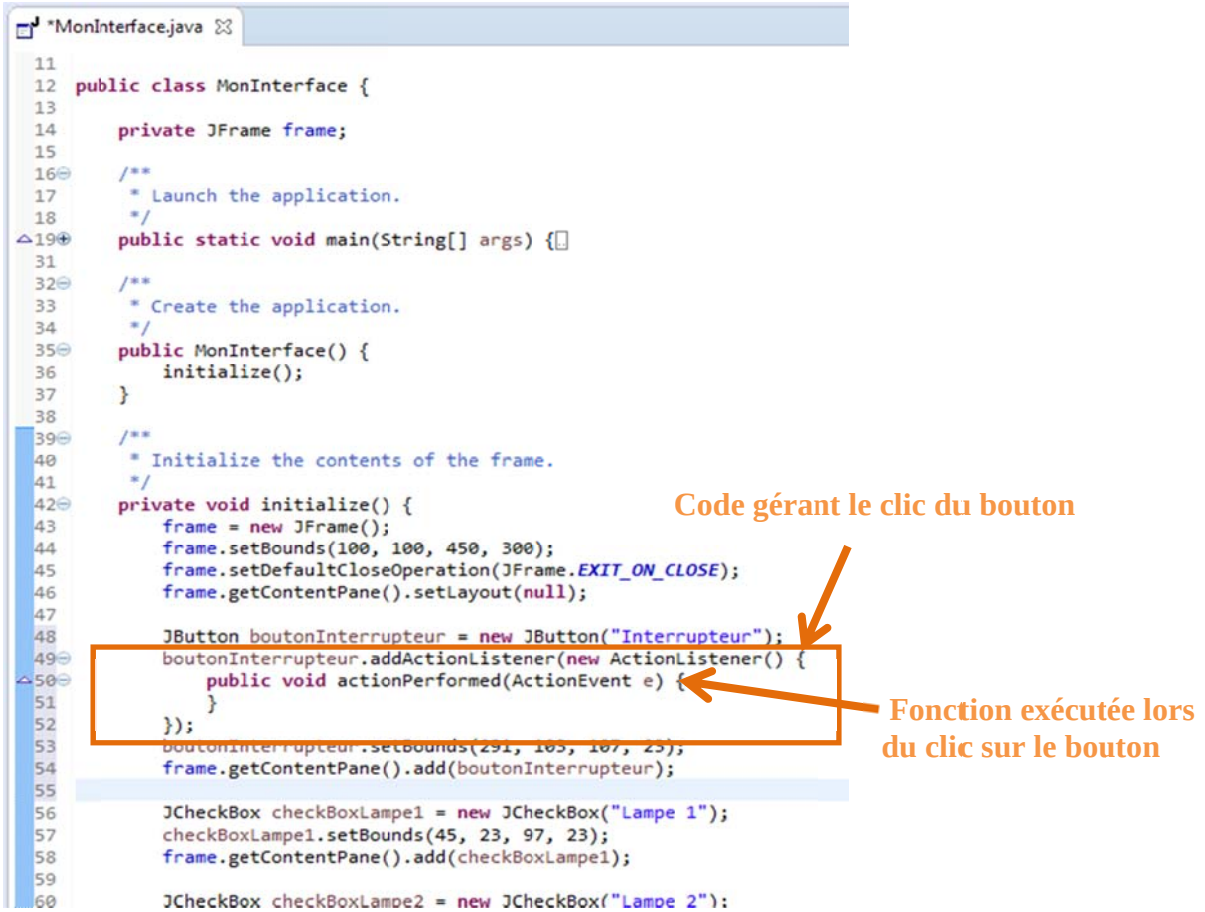
Le constructeur de la classe *ControleurInterrupteur* fait le lien entre un bouton graphique (de la *Vue*) et d'un objet Interrupteur (du *Modèle*).

Le contrôleur de l'interrupteur doit intercepter l'évènement « clic sur le bouton graphique » (de la *Vue*) pour ensuite traiter l'information et exécuter une méthode associée à un objet de la classe Interrupteur (du *Modèle*).
WindowBuilder génère automatiquement ce code. Il suffit ensuite de le recopier dans la classe *ControleurInterrupteur* au bon endroit.

Sur la fenêtre graphique (onglet *Design*), sélectionner le bouton graphique puis faites un clic-droit pour faire apparaître un menu surgissant. Choisissez alors l'option *Add event handler – action – actionPerformed.*



WindowBuilder crée alors le code ci-dessous dans le constructeur de la fenêtre graphique




```
11
12  public class MonInterface {
13
14      private JFrame frame;
15
16      /**
17       * Launch the application.
18       */
19      public static void main(String[] args) {
31
32      /**
33       * Create the application.
34       */
35      public MonInterface() {
36          initialize();
37      }
38
39      /**
40       * Initialize the contents of the frame.
41       */
42      private void initialize() {
43          frame = new JFrame();
44          frame.setBounds(100, 100, 450, 300);
45          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
46          frame.getContentPane().setLayout(null);
47
48          JButton boutonInterrupteur = new JButton("Interrupteur");
49          boutonInterrupteur.addActionListener(new ActionListener() {
50              public void actionPerformed(ActionEvent e) {
51              }
52          });
53          boutonInterrupteur.setBounds(291, 163, 107, 23);
54          frame.getContentPane().add(boutonInterrupteur);
55
56          JCheckBox checkBoxLampe1 = new JCheckBox("Lampe 1");
57          checkBoxLampe1.setBounds(45, 23, 97, 23);
58          frame.getContentPane().add(checkBoxLampe1);
59
60          JCheckBox checkBoxLampe2 = new JCheckBox("Lampe 2");
```

Le code gérant le clic du bouton doit alors être déplacé **dans le constructeur** de la classe *controleurInterrupteur* et complété avec l'action à faire lors de l'appuie sur le bouton.

Le contrôleur gère donc maintenant l'interaction de la *Vue* vers le *Modèle*.

6- Finalisation de l'interface

6.1- Ajustement de la structure de la classe MonInterface

La classe de l'interface graphique doit mettre à disposition de plusieurs méthodes (*update* par exemple) les objets graphiques qu'elle manipule. Vous devez donc y définir des attributs correspondant aux objets graphiques manipulées.
Vous créerez également un attribut permettant à la classe d'atteindre les objets du *Modèle* à partir d'un interrupteur.

```
// Les propriétés de la classe
private JButton boutonInterrupteur;
private JLabel labelCommutateur;
private JCheckBox checkBoxLampe1;
private JCheckBox checkBoxLampe2;
// etc…

private Interrupteur unInterrupteur;
```

Du coup, il faut modifier la méthode *initialize*() où les objets graphiques ont été définis automatiquement par WindowBuilder. Par exemple,

```
Jcheckbox checkBoxLampe1 = new JCheckBox("Lampe 1");
checkBoxLampe1.setBounds(45, 23, 97, 23);
frame.getContentPane().add(checkBoxLampe1);
```

deviendra

```
checkBoxLampe1 = new JCheckBox("Lampe 1");
checkBoxLampe1.setBounds(45, 23, 97, 23);
frame.getContentPane().add(checkBoxLampe1);
```

6.2- Les instances des classes

Vous devez maintenant coder les instructions créant les objets des classes du *Modèle* au début du programme principale *main ()*.

```
// Construction des objets du Modèle
// Création de l'interrupteur qui crée le commutateur qui crée 3 lampes
Interrupteur unInterrupteur = new Interrupteur();
```

Dans le constructeur de la classe *MonInterface*, après l'appel de la méthode *initialize()* qui créer les objets graphiques, *v*ous coderez les instructions créant l'objet de la classe *ControleurInterrupteur*.

```
// * Création du Controleur : lien entre le Modéle et la Vue
new ControleurInterrupteur (unInterrupteur, boutonInterrupteur);
```

6.3- Définir la classe de la Vue en tant qu'Observer

Il faut maintenant gérer le retour d'information du *Modele* vers la *Vue*. Par exemple si une lampe (du *Modèle*) passe dans l'état allumé, il faut que la checkbox (de la *Vue*) qui lui est associée soit cochée.

Modifiez l'entête de la classe *MonInterface* pour la déclarer en tant qu'observateur. Ecrivez également la méthode update que l'on laissera vide pour l'instant.

```
public class MonInterface implements Observer {
```

Il faut maintenant spécifier les objets qui sont observés par l'interface graphique : le commutateur et les lampes. Vous pouvez faire ce lien à la fin du constructeur.

```
public MonInterface(Interrupteur inter) {

        initialize();

        // L'interface graphique Observe les lampes et le commutateur
        this.unInterrupteur = inter;
        Lampe[] lampes = unInterrupteur.getCommutateur().getLampes();
        for (int i = 0; i < lampes.length; i++) {
            lampes[i].addObserver(this);
        }
        unInterrupteur.getCommutateur().addObserver(this);

        // * Création du Controleur : lien entre le Modéle et la Vue
        new ControleurInterrupteur(unInterrupteur, boutonInterrupteur);
}
```

6.4- La conception du update

Il faut maintenant construire le *update* de la classe *MonInterface* qui va mettre à jour la *Vue* lors d'un changement sur une lampe ou sur le commutateur.

```
public void update(Observable instanceObservable, Object arg1)
```

Cette méthode sera déclenchée après une modification d'une lampe ou du commutateur. Il faut donc prévoir de distinguer ces deux cas :

```
if (instanceObservable instanceof Lampe){
        switch (((Lampe)instanceObservable).getNumero()) {
        case 0 : // à remplir
        case 1: //…
                //… à remplir
        }
}

if (instanceObservable instanceof Commutateur){
// à remplir
}
```

7- Une seconde interface concurrente.

Implanter maintenant une seconde interface, concurrente à la première, qui fonctionne en ligne de commandes.

Pour vous aider voici un squelette possible pour cette interface

```
public class VueTexte implements Observer, Runnable {

    public static String QUITTER = "Quit";
    public static String COMMUTER = "C";
    public static String PROMPT = ">";

    private Interrupteur interrupteur;

    public VueTexte(Interrupteur inter)
      // A compléter

      Thread t = new Thread(this);
      t.start();
    }

    public void run() {

      String saisie = null;
      boolean quitter = false;

      System.out.println("Taper " + VueTexte.COMMUTER + " pour commuter ; " +
VueTexte.QUITTER + " pour quitter.");

      do {
          saisie = this.lireChaine();

          if (saisie != null) {
            // A compléter
          }
      } while (quitter == false);
      System.exit(0);
    }

    private String lireChaine() {
      BufferedReader br = new BufferedReader (new InputStreamReader(System.in));
      String resultat = null;
      try {
          System.out.print(VueTexte.PROMPT);
          resultat = br.readLine();
      } catch (IOException e) {
          System.err.println(e.getMessage());
      }
      return resultat;
    }

    @Override
    public void update(Observable arg0, Object arg1) {
      // A compléter
    }
```