AP3 – Programmation en Java
TD 04

Mickael Montassier et Arnaud Pêcher

6 septembre 2011

Résumé

Présentation de AWT et SWING.

L'ensemble des fichiers de cette séance est disponible dans :
`/net/Bibliotheque/AP3/TD04/code/`

1 Cours

Une **interface graphique** est constituée de **composants** graphiques (boutons, menus, fenêtres,...) **disposés** dans des **conteneurs** (cadres, panneaux, ...) capables de réagir à des **évènements** (souris, clavier).

En Java, les interfaces graphiques sont obtenues par assemblage de composants – **Component** –. Les composants sont regroupés dans des conteneurs – **Container** –. La disposition des composants est gérée par des **LayoutManager**. Enfin, la réaction aux évènements se fait via des **EventListener**.

Sur le site de **SUN**, parcourez les pages **Component, Container, LayoutManager, EventListener**.

1.1 AWT & SWING

```
import java.awt.*;
import javax.swing.*;
```

AWT : Abstract Window Toolkit
- Abstract : Unix, DOS, Window, MacOSX
- Window : gestion des fenêtres graphiques
- Toolkit : boite à outils

SWING : extension de AWT (**J**Classe)

1.2 Composant/Component

Les composants sont les briques élémentaires d'une interface graphique. Quelques exemples :

- **JLabel** : texte non modifiable
- **JTextField** : ligne de texte éditable
- **JButton** : bouton
- **JMenu** : menu
- ...

1.3 Component vs. Container

Un **Container** est un **Component** qui possède la particularité de pouvoir contenir d'autres **Component**. Par exemple, les **JComponent** sont des **Container**. Il est possible d'imbriquer des conteneurs et de construire une hiérarchie d'objets graphiques.

Pour ajouter un composant x dans un conteneur y, il suffit d'utiliser la méthode `add` :

```
y.add(x);
```

Quelques conteneurs :

- **JPanel** : très simple, convient pour recevoir des boutons,...
- **JFrame** : fenêtre top-level avec titre et bordures
- **JDialog** : permet d'obtenir une entrée de l'utilisateur
- ...

1.4 Gestionnaire d'affichage/LayoutManager

L'interface **LayoutManager** est implémentée par un certain nombre de classes de AWT. Un **LayoutManager** gère la disposition des composants dans un conteneur.

Les principaux **LayoutManager** sont :

- **FlowLayout** : arrange les composants de gauche à droite
- **BorderLayout** : dispose les composants dans les régions - `north`, `south`, `east`, `west`, `center`
- **GridLayout** : arrange les éléments sur une grille.

Un conteneur possède un attribut **LayoutManager** affecté par défaut : FlowLayout pour JPanel, BorderLayout pour JPanel, ... Il est possible de changer le LayoutManager d'un conteneur : par exemple,

```
p.setLayout(new BorderLayout());
```

ou de faire l'agencement à la main :

```
p.setLayout(null);
x.setLocation(a,b);     ou   x.setBounds(a,b,c,d);
x.setSize(c,d)
```

1.5 Exemples

Dans cet exemple, nous allons créer une fenêtre principale de deux façons différentes : par délégation et par héritage.

1.5.1 Création d'une fenêtre par délégation

```
// Fenetre.java
import java.awt.*;
import javax.swing.*;

public class Fenetre {
    private JFrame frm;
    private JButton b1, b2;
    private JLabel lbl;

    public Fenetre(String titre){
        frm=new JFrame(titre);
        frm.setBounds(0,0,400,200);
        b1=new JButton("B1");
        b2=new JButton("B2");
        lbl=new JLabel("texte quelconque");
        frm.add(b1,"North");
        frm.add(lbl,"Center");
        frm.add(b2,"South");
        frm.setVisible(true);
        //frm.pack();
    }
}

// FenetreApp.java
public class FenetreApp{
    public static void main(String[] args){
        Fenetre f=new Fenetre("Exemple 1");
    }
}
```

Pour toute méthode, reportez-vous à la doc SUN.

1.5.2 Création d'une fenêtre par héritage

```
import java.awt.*;
import javax.swing.*;

public class Fenetre extends JFrame {
    private JButton b1, b2;
    private JLabel lbl;

    public Fenetre(String titre){
        super(titre);
        setBounds(0,0,400,200);
        b1=new JButton("B1");
        b2=new JButton("B2");
        lbl=new JLabel("texte quelconque");
        add(b1,"North");
        add(lbl,"Center");
        add(b2,"South");
        setVisible(true);
        // pack();
    }
}
```

1.6 Graphics

La classe **java.awt.graphics** est une classe abstraite héritant directement de **Object**. Elle représente le contexte graphique permettant de dessiner sur un composant.

Il est possible d'obtenir le contexte graphique d'un composant x par :

```
x.getGraphics();
```

Il est possible de dessiner dans le contexte graphique :

- `drawOval(int x, int y, int w, int h)`
- `drawLine(int x1, int y1, int x2, int y2)`
- `drawRect(int x, int y, int w, int h)`
- `setColor(Color c)`
- ...

1.7 Exemple

Dans une fenêtre, ajouter un **Canvas** dans lequel sera dessinée une ellipse.

```
// Fenetre.java
import java.awt.*;
import javax.swing.*;

public class Fenetre extends JFrame {
    private Canvas can;

    public Fenetre(String titre){
        super(titre);
        setBounds(0,0,400,200);
        can=new Canvas();
        add(can,"Center");
        setVisible(true);
        Graphics g = can.getGraphics();
        g.setColor(Color.black);
        g.drawOval(160,10,65,100);
    }
}
```

Bizarre... l'ellipse apparaît une fraction de seconde et disparait.

En fait quand un conteneur s'affiche ou se réaffiche, il appelle sa méthode **paint(Graphics g)** qui appelle la méthode **paint(Graphics g)** de chacun des composants présents dans le conteneur. Or la méthode **paint** de Canvas remplit le dessin de blanc... Que faire ?

Une réponse possible : Il faut alors créer une classe **Dessin** dérivée de **Canvas** et redéfinir la méthode `paint` de **Canvas** dans cette classe.

```
// Dessin.java
import java.awt.*;

public class Dessin extends Canvas{
    public void paint (Graphics g) {
        super.paint(g);
        g.setColor(Color.black);
        g.drawOval(10,10,150,100);
    }
}
```

```
// Fenetre.java
import java.awt.*;
import javax.swing.*;

public class Fenetre extends JFrame {
    private Dessin dessin;

    public Fenetre(String titre){
        super(titre);
        setBounds(0,0,400,200);
        dessin=new Dessin();
        add(dessin,"Center");
        setVisible(true);
    }
}
```

1.8 Réponse aux évènements

Principe :

- Dans certaines circontances (clic souris, touche pressée, etc), un composant crée un objet **évènement**.
- Un **contrôleur** (gestionnaire d'évènement) est un objet capable de traiter un évènement.
- Il suffit d'**associer** un **contrôleur** à un **composant** pour gérer les évènements.

Un gestionnaire d'évènements (ou contrôleur) est une instance d'une classe qui implémente une interface dérivée de **EventListener**.

Voir : **ActionListener, MouseListener, MouseMotionListener,...**

Par exemple :

```
class BoutonClic implements ActionListener {
      public void actionPerformed(ActionEvent e){
          System.out.println("clic");
      }
}
```

Il faut ensuite instancier un gestionnaire d'évènements approprié et l'associer au composant concerné :

```
b.addActionListener(new BoutonClic())
```

Grâce à cette association, lorsque le composant b émettra un **ActionEvent**, cet évènement sera traité par le **BoutonClic** qui exécutera sa méthode `actionPerformed`.

1.9 Exemple

Dans une fenêtre, ajoutons deux boutons et une ellipse. En réponse au clic sur le premier bouton, l'ellipse change de couleur et devient rouge. En réponse au clic sur le second bouton, l'ellipse retrouve sa couleur initiale. Dans un premier temps essayez de le faire seul.

Un peu d'aide ? :

```
// ChangeCouleur.java
import java.awt.event.*;
import java.awt.*;

public class ChangeCouleur implements ActionListener{
    private Color couleur;
    private Dessin dessin;

    public ChangeCouleur(Color couleur,Dessin dessin){
        this.couleur=couleur;
        this.dessin=dessin;
    }

    public void actionPerformed(ActionEvent e){
        dessin.setColor(couleur);
        dessin.repaint();
    }
}

// Dessin.java
import java.awt.*;

public class Dessin extends Canvas{
    private Color couleur;

    public void setColor(Color couleur){
        this.couleur=couleur;
    }

    public void paint (Graphics g){
        super.paint(g);
        g.setColor(couleur);
        g.drawOval(10,10,150,100);
    }
}


// Fenetre.java
import java.awt.*;
import javax.swing.*;

public class Fenetre extends JFrame {
    private JButton b1, b2;
    private Dessin dessin;

    public Fenetre(String titre){
        super(titre);
        setBounds(0,0,400,200);
        b1=new JButton("B1");
        b2=new JButton("B2");
        dessin=new Dessin();
        b1.addActionListener(new ChangeCouleur(Color.red,dessin));
        b2.addActionListener(new ChangeCouleur(dessin.getForeground(),dessin));
        add(b1,"North");
        add(dessin,"Center");
        add(b2,"South");
        setVisible(true);
    }
}
```

1.10 Exercice : le jeu de Taquin

Vous allez réaliser le jeu en 4 étapes. Pour chacune de ces étapes, il est conseillé de lancer l'éxécutable disponible dans :
`/net/Bibliotheque/AP3/TD04/code/02-taquin/Etape-0?/`

Étape 1. Dessiner une grille vide. Une grille est un **Canvas** qui possède les attributs suivant : `nbLignes`, `nbColonnes`, `abs`, `ord`.

Étape 2. Ajouter une classe **Jeton** : un Jeton possède une position dans la grille, un numéro, une couleur. Ajouter dans Grille une structure de données permettant de stocker l'ensemble des jetons.

Étape 3. Ajouter une classe **Click** étendant **MouseInputAdapter** (implémentation par défaut de MouseInputListener) qui permet de déplacer les jetons.

Étape 4. Ajouter les boutons de contrôle - ajout/suppresion d'une ligne, ajout/suppression d'une colonne, réinitialisation - ainsi que la gestion des évènements associés à chaque clic.

Un peu d'aide ? :

Correction Étape 1.

```
// Grille.java
import java.awt.*;
import javax.swing.*;

public class Grille extends Canvas {
    private int nbLignes;
    private int nbColonnes;
    private int abs;
    private int ord;
    final static int largeurColonne = 40;
    final static int hauteurLigne   = 40;

    public Grille(int nbLignes, int nbColonnes,
                  int abs, int ord){
        this.nbLignes = nbLignes;
        this.nbColonnes = nbColonnes;
        this.abs = abs;
        this.ord = ord;
    }
    // Override paint
    public void paint(Graphics g){
        super.paint(g);
        g.setColor(Color.black);

        // dessin de la grille
        for(int i=0;i<nbLignes;i++){
                 for(int j=0;j<nbColonnes;j++){
                          g.drawRect(abs+j*largeurColonne,ord+i*hauteurLigne,
                                     largeurColonne,hauteurLigne);
                 }
        }
    }
}

// Taquin.java
import java.awt.*;
import javax.swing.*;

public class Taquin extends JFrame {
    private Grille grille;
    private int nbLignesTaquin;
```

```
    private int nbColonnesTaquin;
    final static int largeurFenetre = 400;
    final static int hauteurFenetre = 400;

    public Taquin(String titre){
        super(titre);
        nbLignesTaquin = 3;
        nbColonnesTaquin = 3;
        setBounds(0,0,largeurFenetre,hauteurFenetre);
        int x_0 = (largeurFenetre - Grille.getLargeurColonne()*nbColonnesTaquin)/2;
        int y_0 = (hauteurFenetre - Grille.getHauteurLigne()*nbLignesTaquin)/2;
        grille=new Grille(nbLignesTaquin,nbColonnesTaquin,x_0,y_0);
        add(grille,"Center");
        setVisible(true);
    }
    public static void main (String args []){
        new Taquin("Jeu de Taquin");
    }
}
```

Correction Étape 2.

```
// Jeton.java

public class Jeton {
    private int abs;
    private int ord;
    private int numero;
    private Color couleur;

    public Jeton(int abs,int ord,int numero,Color couleur) {
        this.abs = abs;
        this.ord = ord;
        this.numero = numero;
        this.couleur = couleur;
    }
    public int getAbs(){
        return abs;
    }
    public int getOrd(){
        return ord;
    }
    public int getNumero(){
        return numero;
    }
    public Color getCouleur(){
        return couleur;
    }
}

// Grille.java
import java.awt.*;
import javax.swing.*;

public class Grille extends Canvas{
    private int nbLignes;
    private int nbColonnes;
    private int abs;
    private int ord;
    final static int largeurColonne = 40;
    final static int hauteurLigne   = 40;
    private Jeton jetons [][];
    private int nbJetons;

    public Grille(int nbLignes, int nbColonnes,
                  int abs, int ord)
    {
```

```
        this.nbLignes = nbLignes;
        this.nbColonnes = nbColonnes;
        this.abs = abs;
        this.ord = ord;
        nbJetons = nbLignes * nbColonnes - 1;
        jetons = new Jeton[nbLignes][nbColonnes];
        int pas = 256/(nbJetons+2); // pas top...
        int num=0;
        for(int i=0;i<nbLignes;i++)
            for(int j=0;j<nbColonnes;j++){
                jetons[i][j]=new Jeton(abs+j*largeurColonne,
                                       ord+i*hauteurLigne,
                                       num++,
                                       new Color(100,num*pas,num*pas));
            }
        jetons[nbLignes-1][nbColonnes-1]=null;
    }
    // Override paint
    public void paint(Graphics g){
        super.paint(g);
        // dessin des jetons
        for(int i=0;i<nbLignes;i++)
            for(int j=0;j<nbColonnes;j++){
                     if(jetons[i][j]!=null){
                            g.setColor(jetons[i][j].getCouleur());
                            g.fillRect(jetons[i][j].getAbs(),jetons[i][j].getOrd(),
                                       largeurColonne,hauteurLigne);
                            g.setColor(Color.BLACK);
                            g.drawString(""+jetons[i][j].getNumero(),
                                         (int)(jetons[i][j].getAbs()+largeurColonne/2),
                                         (int)(jetons[i][j].getOrd()+hauteurLigne/2));
                     }
            }
        // dessin de la grille
        g.setColor(Color.black);
        for(int i=0;i<nbLignes;i++)
            for(int j=0;j<nbColonnes;j++)
               g.drawRect(abs+j*largeurColonne,ord+i*hauteurLigne,
                          largeurColonne,hauteurLigne);


    }
}

// Taquin.java
import java.awt.*;
import javax.swing.*;

public class Taquin extends JFrame {
    private Grille grille;
    private int nbLignesTaquin;
    private int nbColonnesTaquin;
    final static int largeurFenetre = 400;
    final static int hauteurFenetre = 400;

    public Taquin(String titre){
        super(titre);
        nbLignesTaquin = 4;
        nbColonnesTaquin = 4;
        setBounds(0,0,largeurFenetre,hauteurFenetre);
        int x_0 = (largeurFenetre - Grille.getLargeurColonne()*nbColonnesTaquin)/2;
        int y_0 = (hauteurFenetre - Grille.getHauteurLigne()*nbLignesTaquin)/2;
        grille=new Grille(nbLignesTaquin,nbColonnesTaquin,x_0,y_0);
        add(grille,"Center");
        setVisible(true);
    }
    public static void main (String args []){
        new Taquin("Jeu de Taquin");
```

```
    }
}
```

Correction Étape 3.

```
// Click.java
import javax.swing.event.*;
import java.awt.event.*;
import javax.swing.*;

public class Click extends MouseInputAdapter {
    private Grille grille;

    public Click(Grille g){
        grille=g;
    }
    public void mouseClicked(MouseEvent e){
        int absClick = e.getX();
        int ordClick = e.getY();
        grille.deplacerJeton(absClick,ordClick);
        grille.validate();
        grille.repaint();
    }
}


// Jeton.java
import java.awt.*;
public class Jeton{
    private int abs;
    private int ord;
    private int numero;
    private Color couleur;

    public Jeton(int abs,int ord,int numero,Color couleur){
        this.abs = abs;
        this.ord = ord;
        this.numero = numero;
        this.couleur = couleur;
    }
    public Jeton(Jeton j){
        abs=j.abs;
        ord=j.ord;
        numero=j.numero;
        couleur=j.couleur;
    }
    // Ajouter mutateurs si besoin
}

// Grille.java
import java.awt.*;
import javax.swing.*;

public class Grille extends Canvas{
    private int nbLignes;
    private int nbColonnes;
    private int abs;
    private int ord;
    final static int largeurColonne = 40;
    final static int hauteurLigne   = 40;
    private Jeton jetons [][];
    private int nbJetons;
    private int i_null;
    private int j_null;

    public Grille(int nbLignes, int nbColonnes,
                  int abs, int ord){
```

```
    this.nbLignes = nbLignes;
    this.nbColonnes = nbColonnes;
    this.abs = abs;
    this.ord = ord;
    nbJetons = nbLignes * nbColonnes - 1;
    jetons = new Jeton[nbLignes][nbColonnes];
    int pas = 256/(nbJetons+2); // pas top...
    int num=0;
    for(int i=0;i<nbLignes;i++)
        for(int j=0;j<nbColonnes;j++){
                jetons[i][j]=new Jeton(abs+j*largeurColonne,
                                       ord+i*hauteurLigne,
                                       num++,
                                       new Color(100,num*pas,num*pas));
           }
    jetons[nbLignes-1][nbColonnes-1]=null;
    j_null = nbLignes-1;
    i_null = nbColonnes-1;
}
// Ajouter mutateurs si besoin
// Override paint
public void paint(Graphics g){
    // super.paint(g);
    // dessin des jetons
    for(int i=0;i<nbLignes;i++)
        for(int j=0;j<nbColonnes;j++){
                if(jetons[i][j]!=null){
                        g.setColor(jetons[i][j].getCouleur());
                        g.fillRect(jetons[i][j].getAbs(),jetons[i][j].getOrd(),
                                   largeurColonne,hauteurLigne);
                        g.setColor(Color.BLACK);
                        g.drawString(""+jetons[i][j].getNumero(),
                                     (int)(jetons[i][j].getAbs()+largeurColonne/2),
                                     (int)(jetons[i][j].getOrd()+hauteurLigne/2));
                }
        }
    // dessin de la grille
    g.setColor(Color.black);
    for(int i=0;i<nbLignes;i++)
            for(int j=0;j<nbColonnes;j++)
                    g.drawRect(abs+j*largeurColonne,ord+i*hauteurLigne,
                               largeurColonne,hauteurLigne);
}
public void deplacerJeton(int x,int y){
    // trouver le jeton
    int i_jeton=x-abs;
    int j_jeton=y-ord;
    if( i_jeton >= 0  && i_jeton < nbColonnes*largeurColonne &&
        j_jeton >= 0  && j_jeton < nbLignes*hauteurLigne){
            i_jeton/=largeurColonne;
            j_jeton/=hauteurLigne;
    }
    else
        return;
    //verifier l'adjacence et si ok permutter
    if((i_null == i_jeton && j_null == j_jeton+1) ||
       (i_null == i_jeton && j_null == j_jeton-1) ||
       (i_null == i_jeton+1 && j_null == j_jeton) ||
       (i_null == i_jeton-1 && j_null == j_jeton)){
            Jeton tmp = jetons[j_jeton][i_jeton];
            jetons[j_jeton][i_jeton]=null;
            jetons[j_null][i_null]=tmp;
            jetons[j_null][i_null].setAbs(abs+i_null*largeurColonne);
            jetons[j_null][i_null].setOrd(ord+j_null*hauteurLigne);
            i_null=i_jeton;
            j_null=j_jeton;
    }
}
```

```
}

// Taquin.java
import java.awt.*;
import javax.swing.*;

public class Taquin extends JFrame {
    private Grille grille;
    private int nbLignesTaquin;
    private int nbColonnesTaquin;
    final static int largeurFenetre = 400;
    final static int hauteurFenetre = 400;
    private Click click;

    public Taquin(String titre){
        super(titre);
        nbLignesTaquin = 5;
        nbColonnesTaquin = 5;
        setBounds(0,0,largeurFenetre,hauteurFenetre);
        int x_0 = (largeurFenetre - Grille.getLargeurColonne()*nbColonnesTaquin)/2;
        int y_0 = (hauteurFenetre - Grille.getHauteurLigne()*nbLignesTaquin)/2;
        grille=new Grille(nbLignesTaquin,nbColonnesTaquin,x_0,y_0);
        click = new Click(grille);
        grille.addMouseListener(click);
        add(grille,"Center");
        setVisible(true);
    }
    public static void main (String args []){
        new Taquin("Jeu de Taquin");
    }
}
```

Correction Étape 4.

```
// Taquin.java
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

public class Taquin extends JFrame {
    private Grille grille;
    private int nbLignesTaquin;
    private int nbColonnesTaquin;
    final static int largeurFenetre = 600;
    final static int hauteurFenetre = 400;
    private Click click;
    private JPanel controle;
    private JButton bLPlus, bLMoins, bCPlus, bCMoins, bReInit;

    class bRClick implements ActionListener{
        public void actionPerformed(ActionEvent e){
            reInit();
        }
    }
    class bLClick implements ActionListener{
        private int valIncr;
        public bLClick(int valIncr){
            this.valIncr=valIncr;
        }
        public void actionPerformed(ActionEvent e){
            int tmp = nbLignesTaquin+valIncr;
            if( tmp > 1 && tmp < 9)
                nbLignesTaquin = tmp;
            reInit();
        }
    }
    class bCClick implements ActionListener{
```

```
        private int valIncr;
        public bCClick(int valIncr){
            this.valIncr=valIncr;
        }
        public void actionPerformed(ActionEvent e){
            int tmp = nbColonnesTaquin+valIncr;
            if( tmp > 1 && tmp < 14)
                nbColonnesTaquin = tmp;
            reInit();
        }
    }
    public Taquin(String titre){
        super(titre);
        nbLignesTaquin = 5;
        nbColonnesTaquin = 5;
        setBounds(0,0,largeurFenetre,hauteurFenetre);
        int x_0 = (largeurFenetre - Grille.getLargeurColonne()*nbColonnesTaquin)/2;
        int y_0 = (hauteurFenetre - Grille.getHauteurLigne()*nbLignesTaquin)/2 - 40;
        grille=new Grille(nbLignesTaquin,nbColonnesTaquin,x_0,y_0);
        click = new Click(grille);
        controle = new JPanel();
        bLPlus = new JButton("nb. lig. +");
        bLMoins = new JButton("nb. lig. -");
        bCPlus = new JButton("nb. col. +");
        bCMoins = new JButton("nb. col. -");
        bReInit = new JButton("reinitialiser");
        bReInit.addActionListener(new bRClick());
        bLPlus.addActionListener(new bLClick(+1));
        bLMoins.addActionListener(new bLClick(-1));
        bCPlus.addActionListener(new bCClick(1));
        bCMoins.addActionListener(new bCClick(-1));
        controle.add(bLMoins);
        controle.add(bLPlus);
        controle.add(bCMoins);
        controle.add(bCPlus);
        controle.add(bReInit);
        grille.addMouseListener(click);
        add(controle,"North");
        add(grille,"Center");
        setVisible(true);
    }
    public void reInit(){
        grille.removeMouseListener(click);
        remove(grille);
        repaint();
        int x_0 = (largeurFenetre - Grille.getLargeurColonne()*nbColonnesTaquin)/2;
        int y_0 = (hauteurFenetre - Grille.getHauteurLigne()*nbLignesTaquin)/2 - 40;
        grille = new Grille(nbLignesTaquin,nbColonnesTaquin,x_0,y_0);
        click = new Click(grille);
        grille.addMouseListener(click);
        add(grille,"Center");
        setVisible(true);
    }
    public static void main (String args []){
        new Taquin("Jeu de Taquin");
    }
}
```

1.11 Homework

D'après le chapitre "Construire une interface graphique" de "Java La Synthese" de G. Clavel, N. Mirouze, S. Munerot, E. Pichon, et M. Soukal.

Parcourez l'ensemble du code disponible dans :
`/net/Bibliotheque/AP3/TD04/code/03-editeur/`

Il s'agit de la création d'un petit éditeur de dessin (en 11 étapes. Au final $\pm$450 lignes de code). Tous les concepts (voire plus) y sont repris.

Enfin pour ceux qui auront encore faim de code, vous trouverez une mise en œuvre d'applets dans :
`/net/Bibliotheque/AP3/TD04/code/04-applet/`

2 Applications

Ajoutons une interface graphique à notre labyrinthe ! Le but ici est d'interfacer le labyrinthe du premier TD basé sur une matrice 2D de salles.

1. Récupérer le code de base du labyrinthe dans :
   `/net/Bibliotheque/AP3/TD04/code/05-labyrinthe/01-code-initial/`
2. Écrire la classe **DessinLabyrinthe** étendant **JComponent**. Celle-ci contient un attribut **Labyrinthe** et une méthode `public void paintComponent(Graphics g)` dessinant le labyrinthe (cela à partir de la matrice de salles du labyrinthe).
3. Écrire la classe **LabyrintheGraphiqueApp** qui va construire la fenêtre principale contenant le JComponent DessinLabyrinthe. Testez.
4. Écrire la classe **DessinLabyrintheEtBob** étendant le classe **DessinLayrinthe** permettant d'afficher Bob. Testez.
5. Ajouter la gestion des touches afin de déplacer Bob.
6. Proposer des interfaces graphiques pour les autres labyrinthes.