**Projet Tetris L3 MIAGE 2021**

**Notion Technique : Projet Tetris Java**

Team



Réalisé par l’équipe Skull Striker1 constituée de :

Mr Vincent PISANO Mr Oudeinguene Seycha Mr Jeancy Mupuanga Lumbanzila

**Sommaire**

[**Graphique AWT/SWING/Paint**](#_u4cvwj9q4k7u) **3**

[**Événement Listener**](#_mgzbvqmvrqzj) **6**

# **Graphique AWT/SWING/Paint**

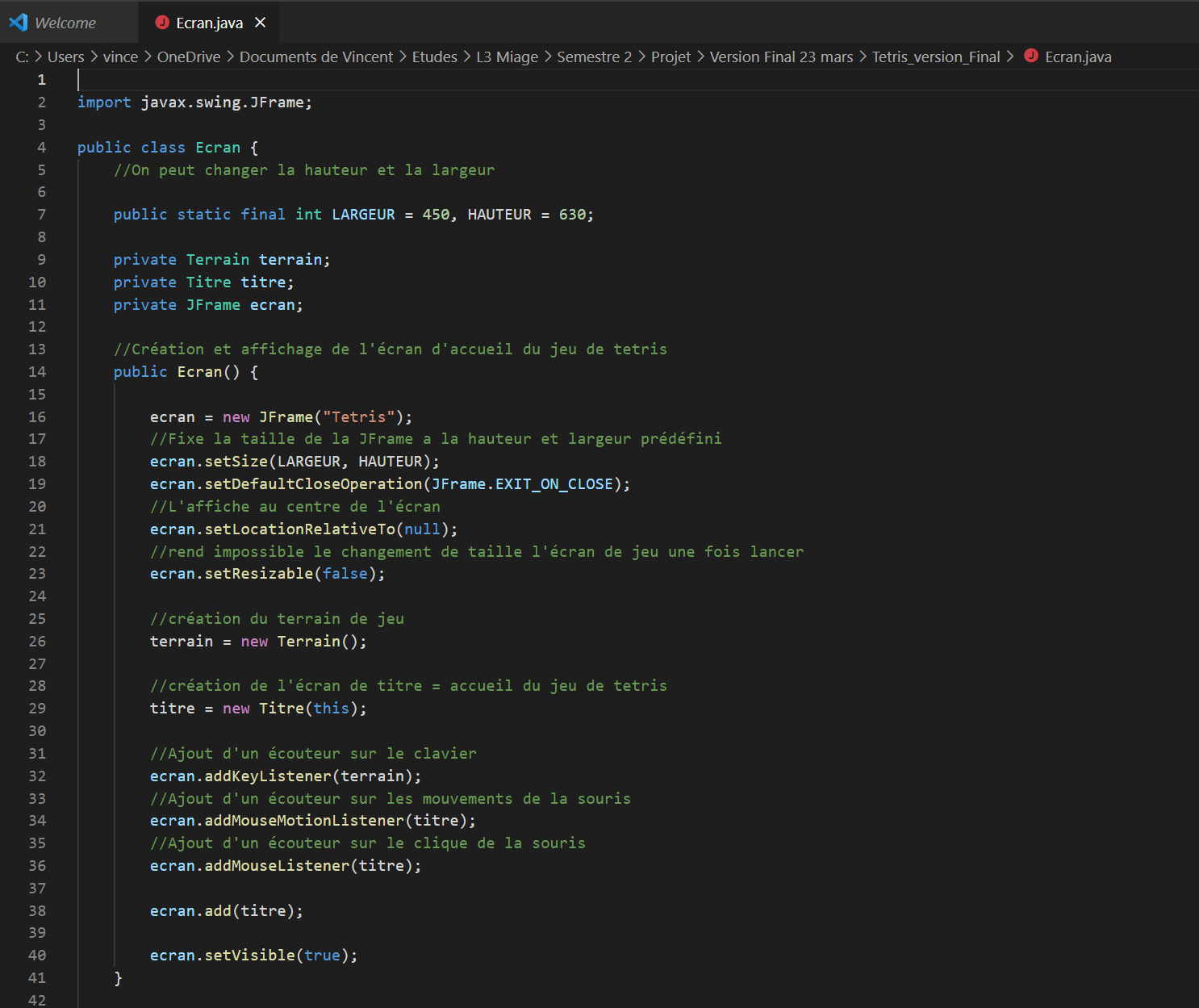
Dans notre projet Tetris nous avons commencé par créer une JFrame.

Qu’est-ce que JFrame?

JFrame est une classe qui se trouve dans le package javax.swing qui hérite de java.awt.frame, il ajoute la prise en charge de l’architecture des composants SWING.

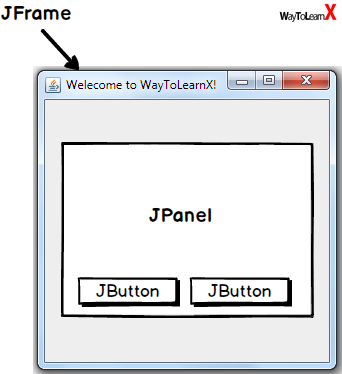
Il s’agit d’une fenêtre de niveau supérieur, avec une bordure et une barre de titre. La classe JFrame possède de nombreuses méthodes qui peuvent être utilisées pour la personnaliser.

Voici la JFrame que nous avons créé :



Après cela, nous avons intégré le fichier Titre.java dans notre JFrame. Cela est possible car Titre est un JPanel.

Un JPanel est une partie du package Java Swing qui est un conteneur qui peut stocker un groupe de composants. La tâche principale de JPanel est d’organiser les composants, diverses dispositions peuvent être définies dans JPanel qui offrent une meilleure organisation des composants, mais il n’a pas une barre de titre comme JFrame.



Voici comment est déclaré le fichier Titre.java afin qu’il soit un JPanel :

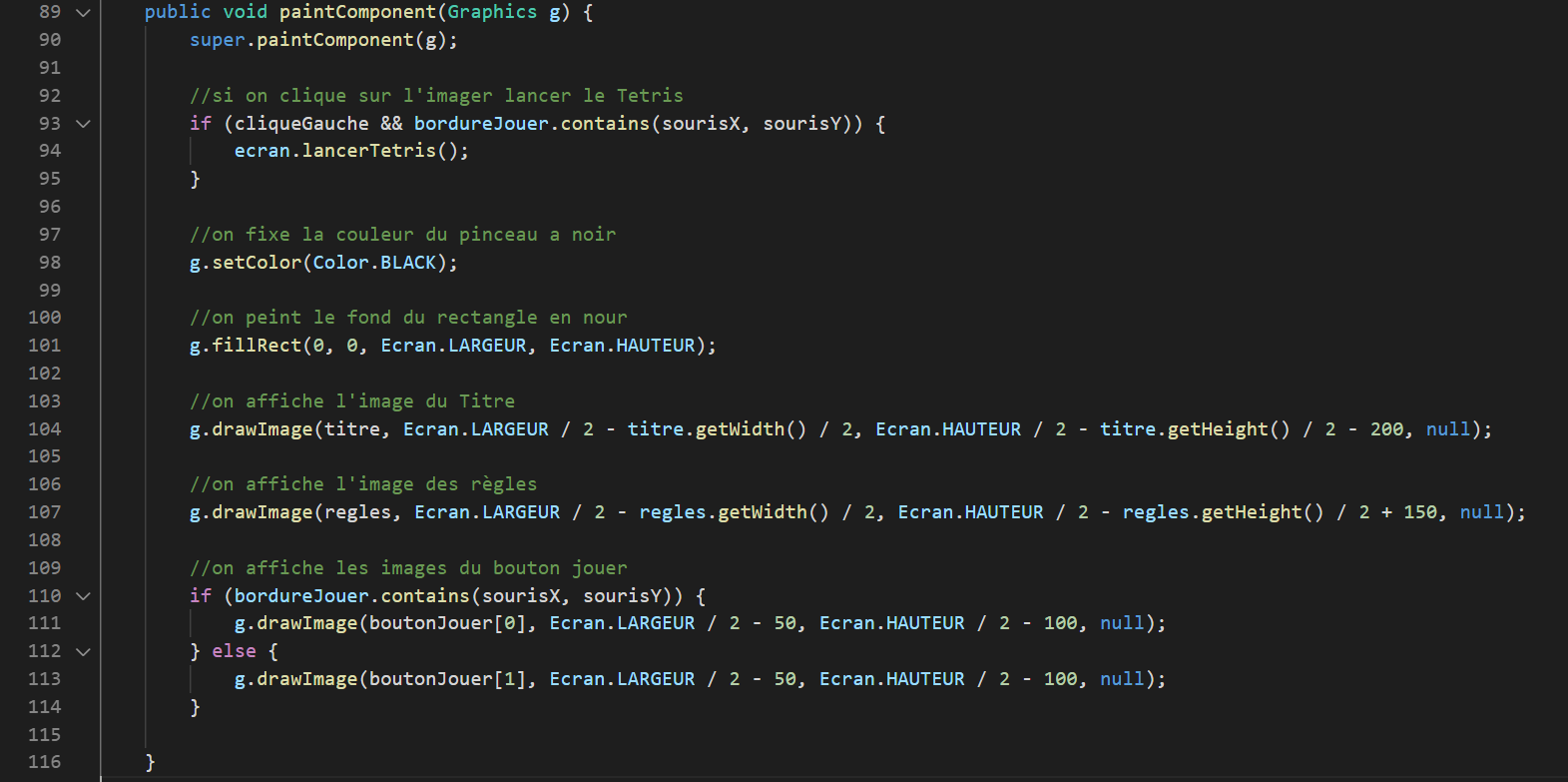
public class Titre extends JPanel

En plus de JFrame et de JPanel nous avons aussi employé la méthode paintComponent.

En Swing, c'est dans cette méthode que doivent être faits les dessins d'un composant. Cette méthode est définie dans la classe JComponent. C'est en re-définissant cette méthode que l'on peut ajouter des instructions de dessin.

Par défaut, la méthode paintComponent appelle la méthode ComponentUI.update() qui efface et redessine le fond si le composant est opaque (comme JPanel par défaut).

Voici ce que réalise cette méthode après que nous l’ayons redéfinit pour la page d’accueil du jeu Tetris :



# **Événement Listener**

Au sein d'une interface graphique (Swing, AWT … ), les listeners(= écouteurs) permettent au programmeur de réagir suite aux actions de l'utilisateur ( exemples d'événements : clic gauche de la souris, appui sur touche du clavier ...).

Les « listeners » sont des interfaces. Ces interfaces fournissent une ou plusieurs méthodes qui peuvent donc être implémentées différemment selon les cas et les besoins, pour répondre aux événements.

Par exemple pour implémenter le listener pour détecter un événement lié à la souris sur un écran graphique, vous devez définir la class comme ceci :

public class MaClasse extends JPanel implements MouseListener

Les interfaces « listener » sont présentes principalement dans le package java.awt.event, mais également dans le package javax.swing.event. Dans notre projet Tetris nous avons principalement utilisé les listeners AWT afin notamment de :

**Détecter le clic gauche sur la souris :**

@Override//détection d'une presse sur le clique gauche

public void mousePressed(MouseEvent e) {

if (e.getButton() == MouseEvent.BUTTON1) {

cliqueGauche = true;

}

}

**Détection des clics sur les touches directionnelles :**

@Override

public void keyPressed(KeyEvent e) {

if (e.getKeyCode() == KeyEvent.VK\_UP) {//flèche du haut

formeCourante.rotationForme();

}

if (e.getKeyCode() == KeyEvent.VK\_RIGHT) {// flèche droite

formeCourante.setDeltaX(1);

}

if (e.getKeyCode() == KeyEvent.VK\_LEFT) {//flèche gauche

formeCourante.setDeltaX(-1);

}

if (e.getKeyCode() == KeyEvent.VK\_DOWN) {//on appui sur la flèche bas

//Donc on accèlère la vitesse de descente de la pièce

formeCourante.augmenterVitesse();

}

}