Compte rendu

SAE Dev

Fauvet Matthis – TP01

Introduction au projet :

Dans ce projet, où le but était donc d'inviter l'utilisateur à créer ou bien à importer un labyrinthe dans le but de déployer des algorithmes dedans afin que celui-ci soit parcouru. Nous avons dû utiliser toutes les différentes connaissances que nous ayons acquis en cours. Interface graphique, parcours longueurs ou connaissances générales en informatique (telle que la lecture de Byte etc.), ce projet est vraiment une aubaine pour se familiariser avec le Java et donc devenir plus rapide tout en ayant une meilleur façon de penser les différents programmes.

Mon projet :

La classe héritage : Fenetre() ;

Cette class est le fondement graphique de notre projet, cette classe qui initialise une fenêtre instantanément afin de pouvoir la manipuler va être notre support pour afficher les grilles ou bien les panneaux de modifications

Voici les différentes classes héritées de Fenêtre :

-FenetreDefSize ;
-FenetreImpGrille ;
-FenetreInit ;
-FenetreRandomGrille ;
-FenetreVideGrille

A la création de chaque objet de ces classes, une fenêtre va être crée et c'est celle-ci que l'on va manipuler.

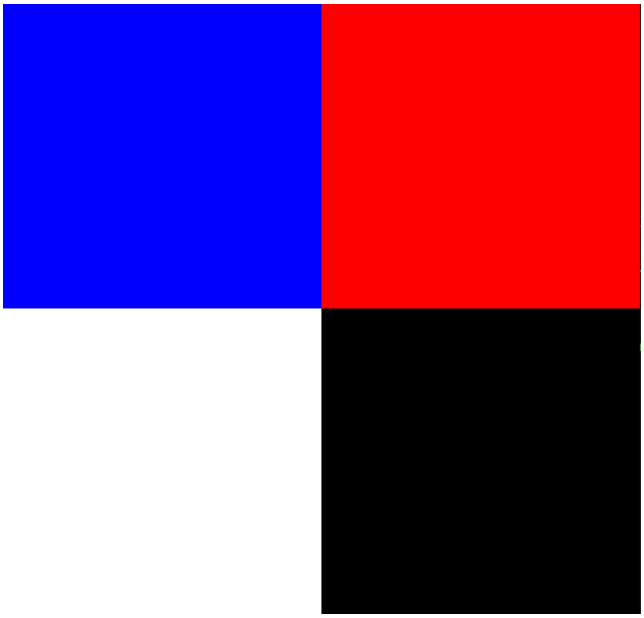
La classe Cellules() :

La classe Cellules est sans l'ombre d'un doute la base même de ce projet, sans cette classe, il ne serai juste pas possible de réaliser certain algo tels qu'ils sont faits actuellement. En effet, la classe Cellules permet de créer des objets « Cellules » qui vont remplir notre grille Graphique, c'est-à-dire la grille qui sera utilisé seulement pour afficher à l'utilisateur le résultat de ses actions.

Les définitions des différentes constantes est primordiale dans ce code, en effet, comme je l'ai dit dans le paragraphe précédent, toute la partie visuelle de la grille va être géré par notre fenêtre mère ainsi que les cellules qui la compose, il est donc primordial d'établir une bonne compréhension du code grâce au nom des constantes. En plus de la lisibilité et la compréhension du code, il est important de pouvoir accéder à chaque modification a ces constantes pour modifier convenablement notre affichage.

Quelques variables static final ou internes à Cellules :

```java
import java.awt.Color;
import javax.swing.JComponent;
import java.awt.Graphics;

public class Cellules extends JComponent{
    public static final int COULOIR=0;
    public static final int MUR=1;
    public static final int ENTREE=2;
    public static final int SORTIE=3;

    public static final int DESSUS=10;
    public static final int VUE=11;

    public static final boolean LIBRE = false;
    public static final boolean OCCUPE = true;

    private Color backgroundColor;

    private int celleLigne;
    private int celleColone;
    private int ceType;
```



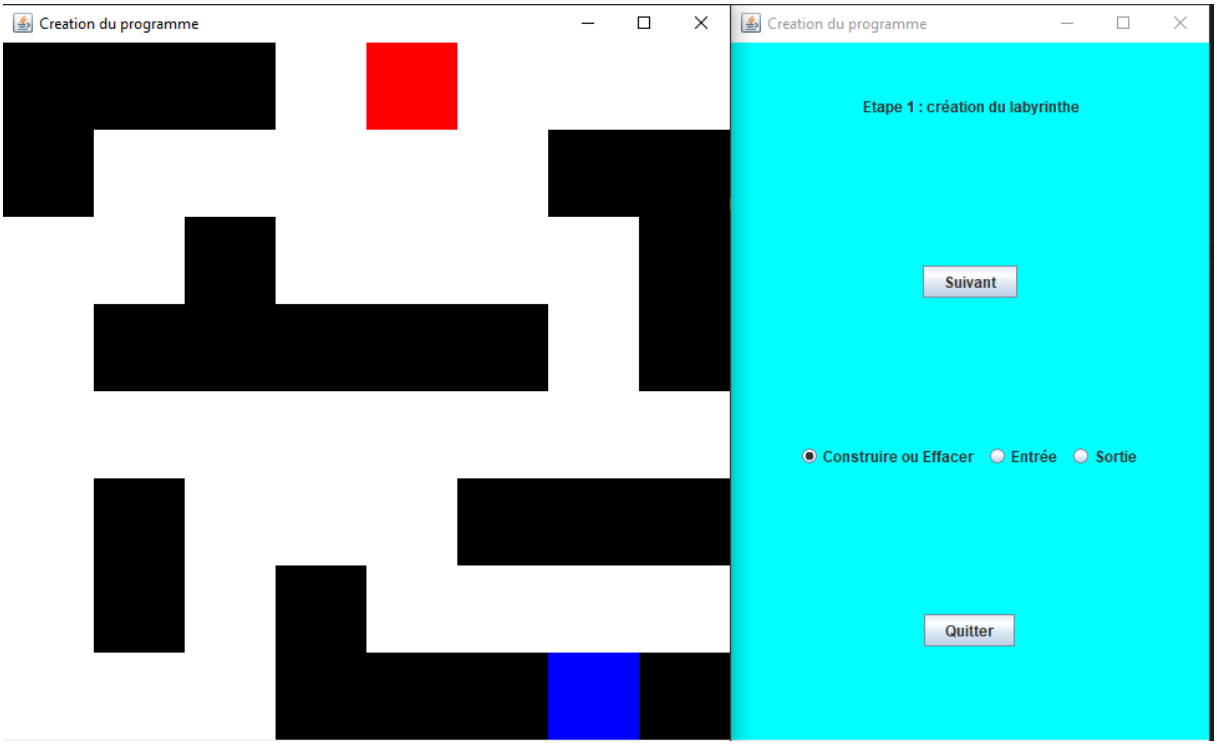
Mais concrètement, qu'est-ce qu'une Cellule ?

Une cellule est tout simplement un JComponent auquel nous allons donner une Ligne et une Colone (afin de le placer et de le modifier convenablement), ainsi qu'un type. Le type, lui, est une constante de notre classe qui permettra **d'afficher** si une case est une entrée, sortie, un mur ou un couloir.

Également, nous pouvons voir la méthode paintComponent de JComponent qui aura pour unique but de changer la couleur de notre cellule si celle-ci devait le faire.

Création et utilisation des fenêtres d'informations :

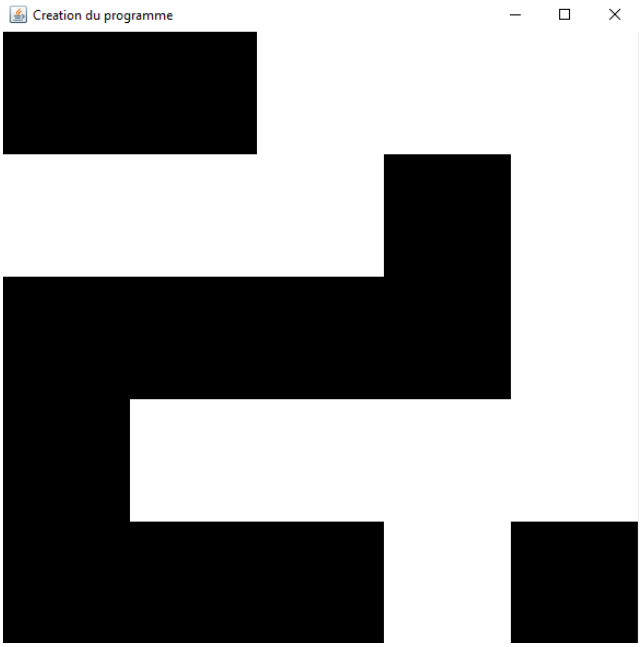
Au cours de cette SAE, j'ai voulu créer un affichage en plusieurs fenêtre qui me paraissait plutôt élégant et utile puisque toutes les options étaient à disposition rapidement. Également, le fait de mettre toutes les étapes en même temps évitait de tout recommencer de 0 en cas d'erreur de l'utilisateur



Génération de grilles aléatoires et importations :

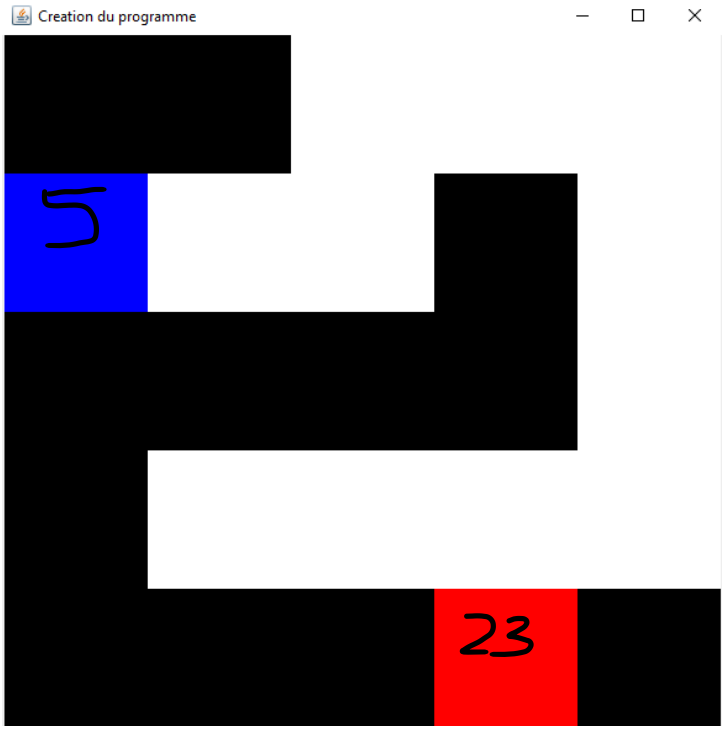
Gestion des case libres ou occupés :

Comme demandé, dans la consigne, un système de création de grille a été ajouté, pour ce faire, un nombre aléatoire est tiré pour chaque case du tableau. Si le numéro vaut 0, alors la case sera un couloir.



Gestion des portes :

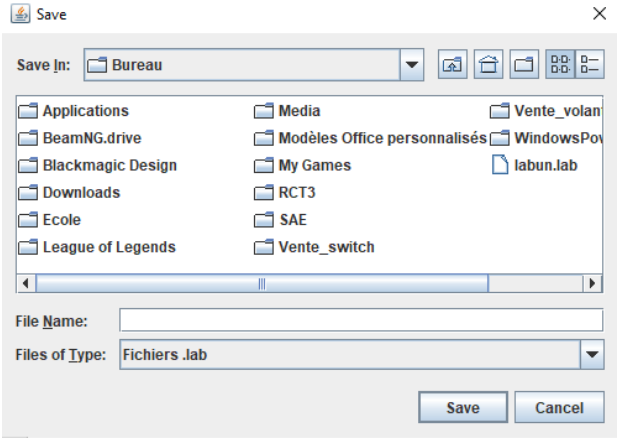
Après avoir posé de case libre et occupe, il fallait poser les deux portes, pour cela on tire un nombre aléatoire entre 0 et la taille max tu tableau et, grâce à un compteur, lorsque celui valait la valeur d'une porte, on déclare cette case comme libre et on pose la porte.

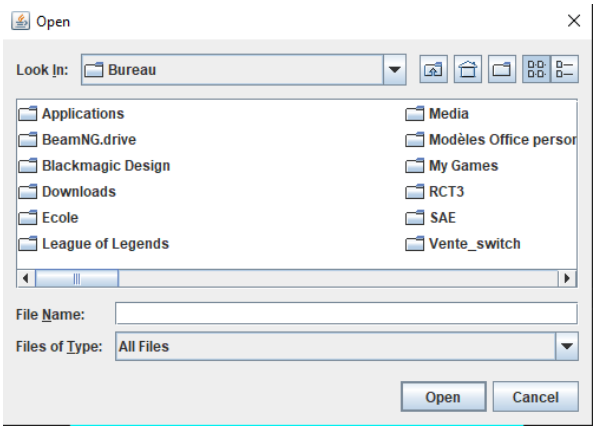


Ecriture et lecture de fichier

Emplacement des fichiers :

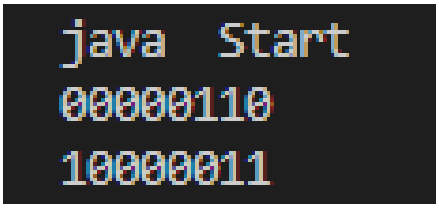
Que ce soit lors de l'ouverture de fichier ou lors de la création d'un nouveau fichier, une fenêtre s'ouvrait pour nous laisser le choix de la position de notre fichier





Encodage des fichiers :

Également, lors de la lecture ou la lecture d'un fichier d'un fichier, puisqu'il fallait utiliser une lecture de flux en lisant les différents octets dans le flux, il était plus ou moins compliqué de comprendre le bon procédé à utiliser au départ.



Les Algorithmes :

Algorithme aléatoire :

Comme son nom l'indique, le chemin que va prendre cet algorithme va être aléatoire. Cependant, s'il est aléatoire, il peut très bien faire des aller et retour a l'infini est c'est le problème principal de cet Algorithme.

Cependant, celui-ci a quand même été ajouté, avec soit l'option pour voir le trajet de celui-ci, soit l'option pour voir la moyenne de coups utilisé pour finir le Labyrinthe :