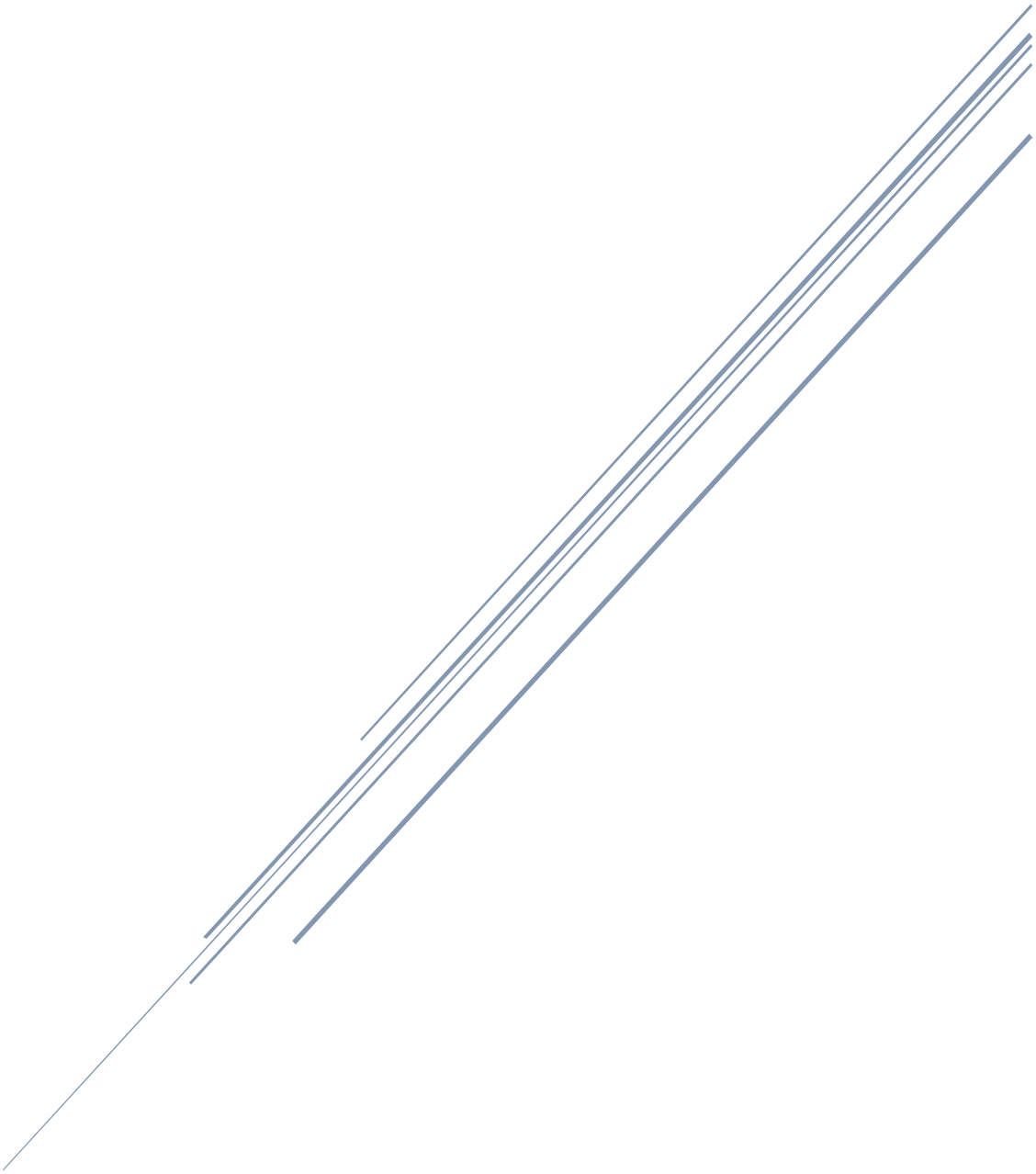
RAPPORT SAMEGAME

Projet tutoré APL 2.1



Quentin LACOMBE & Adam MEDDAHI

Table des matières

Introduction :

Pour ce projet, notre mission fut de réaliser un jeu de SameGame. Ce jeu devait être développé en JAVA sans emprunt extérieur, exception faite pour l'API officielle. Ce jeu devait être accompagné d'un rapport, ce dernier étant celui que vous lisez actuellement. La date limite de ce projet était le mardi 6 avril 2021 à 8h00. La partie logicielle de notre projet fut développée en utilisant le serveur GOGS du département.

Principe du jeu :

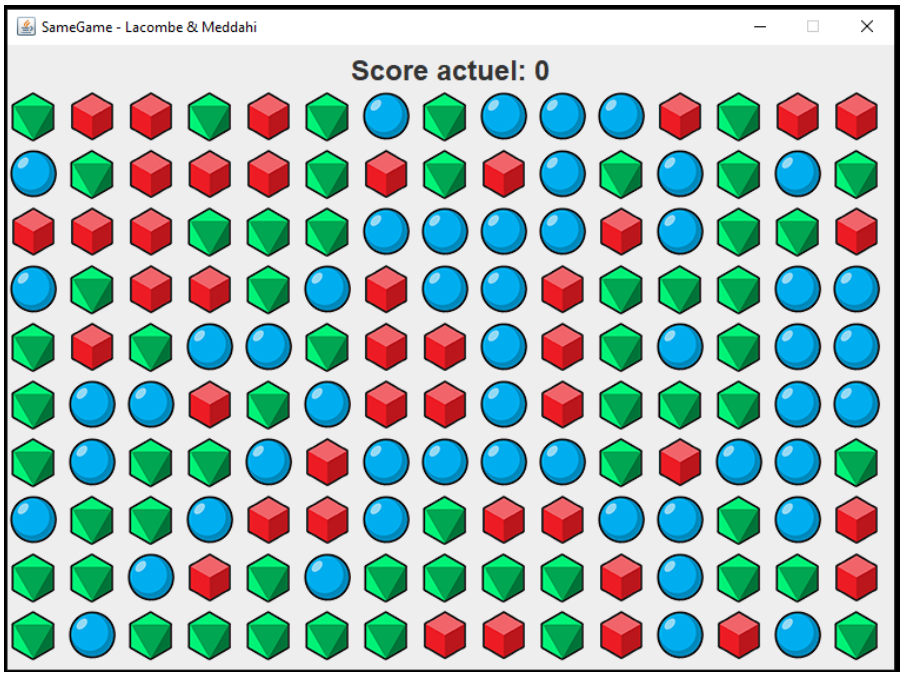
Une grille est remplie de blocs, ces derniers appartiennent à un type parmi les trois types existants. Le but du joueur est de vider cette grille. Le joueur ne peut que retirer des groupes de blocs adjacents (d'un minimum de deux blocs) en cliquant sur le groupe. Le nombre de points gagnés augmente en fonction du nombre de blocs composant le groupe. Les blocs doivent pouvoir se réarranger de telle sorte à boucher les trous de la grille. Lorsque plus aucun groupe n'est présent, la partie est terminée.

Fonctionnalités du programme :

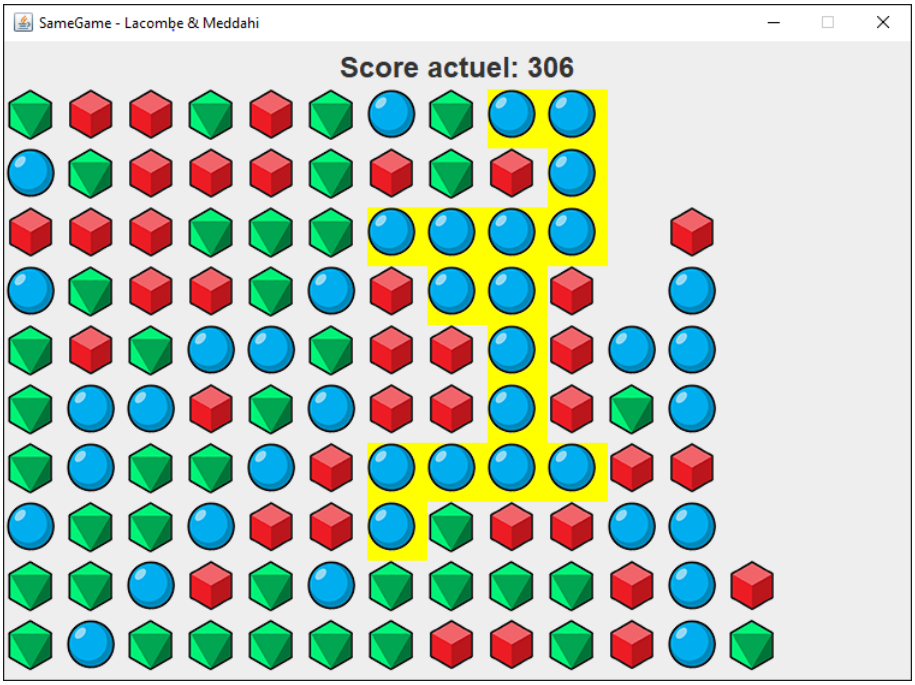
Lorsque l'on lance l'exécutable du jeu, une première fenêtre s'ouvre, il s'agit du menu de début de jeu. Celui-ci est simplement composé du titre du jeu ainsi que les deux boutons qui définiront sur quelle base commence la partie : une grille aléatoire ou une grille définie par un fichier.

Si on clique sur le bouton « Aléatoire », la fenêtre de menu se ferme et se fait remplacer par la fenêtre de jeu.
Si on clique sur le bouton « fichier », la fenêtre du gestionnaire de fichiers se lance pour laisser l'utilisateur sélectionner le fichier de configuration de son choix. Une fois le fichier validé, les fenêtres du gestionnaire de fichier ainsi que du menu de début se ferment, pour être remplacées par la fenêtre principale de jeu.

La fenêtre principale du jeu est constituée de deux panneaux : un qui contient le score et se situe en haut et l'autre qui contient la grille de jeu et se situe en dessous du score. Le score est juste un nombre qui varie en augmentant au fur et à mesure de la partie tandis que la grille est constituée de blocs de couleurs rouges, verts et bleus.



Lorsque cette fenêtre apparait, la partie est déjà lancée et l'utilisateur peut jouer à l'aide de la souris. Si le joueur survole un groupe de blocs de même couleur, le groupe est surligné en jaune et s'il clique dessus, alors le groupe disparait, le score est mis à jour et la grille est réorganisée pour combler l'espace crée par la suppression du groupe. Un groupe est constitué d'au moins deux blocs ; si un bloc indépendant est survolé par la souris, ou même cliqué, rien ne se passe. Lorsqu'une colonne de la grille est complètement vide, alors toutes celles situées à sa droite se voient décalées d'un cran vers la gauche.



Lorsque la partie arrive à son terme et que plus aucun coup n'est possible, la fenêtre de jeu se ferme pour faire place au menu de fin de jeu.

Le menu de fin de jeu est une fenêtre composée de la mention « Fin de la partie ! », ainsi que du score final et de deux boutons, « Rejouer » qui ferme le menu de fin pour faire place au menu de début et « Quitter » qui ferme juste le menu de fin.

Structure du programme

Le programme est constitué de **11 classes** :

- Main : Qui permet d'appeler la classe Menu pour la première fois grâce à la méthode main

-Menu : Qui affiche une fenêtre de menu de début avec comme choix de jouer à partir d'une grille aléatoire ou à partir d'un fichier

-BoutonMenu : Qui gère les actions liées à l'actionnement des deux boutons du menu de début grâce à la méthode actionPerformed héritée d'ActionListener

-Grille : Qui gère le panneau de grille, entre autres son initialisation aléatoire grâce à initGrille ou en se basant sur un fichier grâce à lireFichier et dessinerGrille, ainsi que sa mise à jour après chaque clic grâce à dessinerGrille l'organisation des espaces blancs et la fin de la partie grâce à la méthode reorgaGrille

-GestionnaireSouris : Qui gère le fonctionnement du jeu et les actions de la souris grâce aux méthodes mouseEntered, mouseClicked, mouseExited héritées de MouseListener mais aussi la gestion des groupes grâce à trouveGroupes et testGroupe

-Samegame : Qui affiche une fenêtre contenant le score en haut et le menu en dessous

-Score : Qui gère l'affichage et la mise à jour du panneau de score avec la méthode majScore

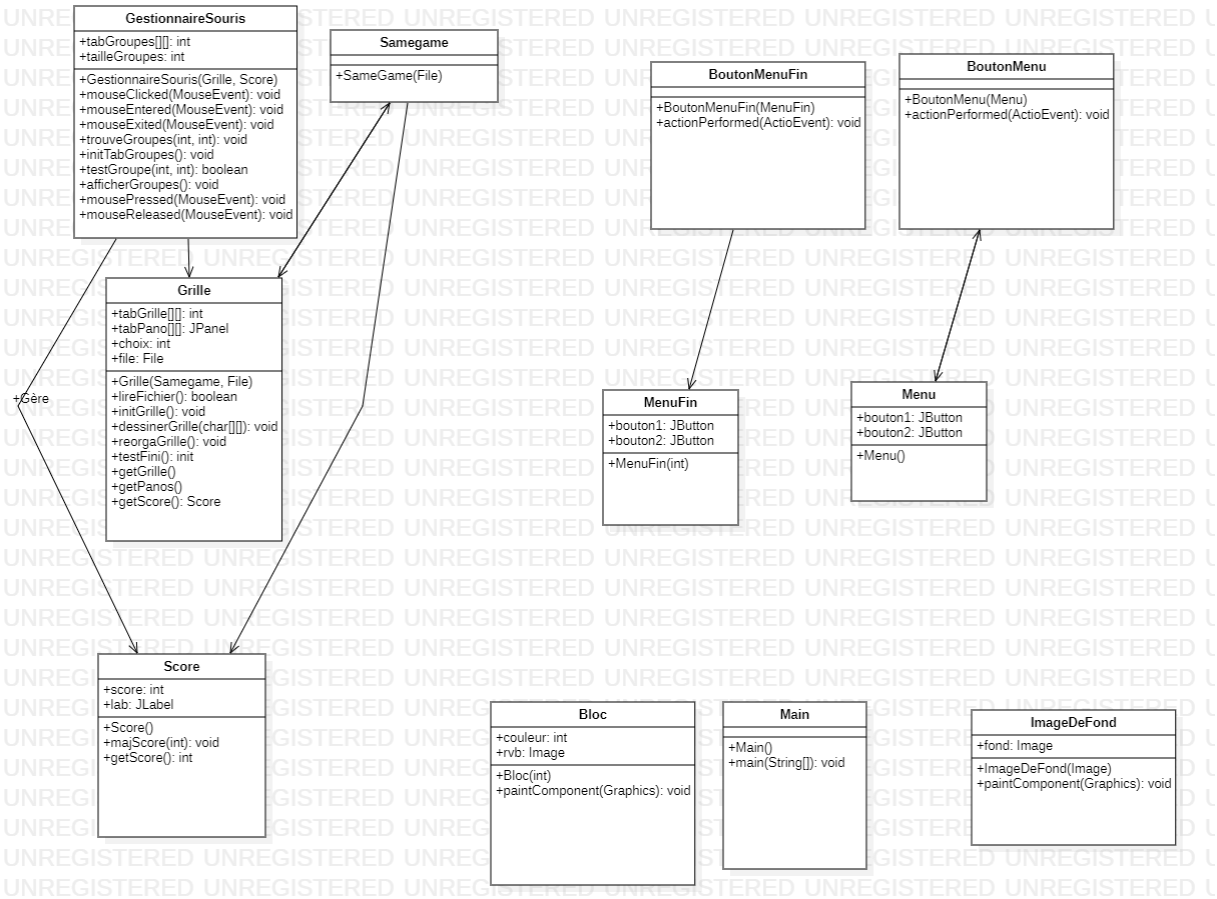
-Bloc : Qui génère des images des blocs Rouge, vert et Bleu grâce à paintComponent héritée de JComponent

-ImageDeFond : Qui génère les images de fond du menu de début et du menu de fin avec paintComponent héritée de JComponent

-MenuFin : Qui affiche une fenêtre de menu de fin avec comme choix de rejouer ou de quitter le jeu

- BoutonMenuFin : Qui gère les actions liées à l'actionnement des deux boutons du menu de fin grâce à la méthode actionPerformed héritée d'ActionListener

Diagramme UML :



Algorithme définissant les groupes :

Le fonctionnement de l'algorithme qui gère les groupes est plutôt simple.
Lorsqu'un un bloc est survolé par la souris, si ce dernier n'est pas « blanc », alors on regarde si chacun des blocs a ses extrémités est de la même couleur que lui, et si c'est le cas on note la position de chacun de ces blocs (s'ils n'ont pas déjà été répertoriés) dans un tableau à deux dimensions puis pour chaque bloc qui est de la même couleur, on regarde ses blocs adjacents et ainsi de suite grâce à la méthode trouveGroupes qui est récursive.
A la fin le tableau à deux dimensions contient les coordonnées du groupe survolé et est remis à 0 lorsqu'il ne l'est plus.

Conclusions personnelles :

Adam MEDDAHI :

La réalisation de ce projet m'a permis, d'une part, de consolider des compétences tel que l'utilisation de Git, la communication avec mon binôme ou la réalisation d'un Makefile, et d'autre part, de mettre en pratique de toute nouvelle connaissance, notamment dans le langage JAVA. J'ai pu me familiariser avec la réalisation de menus. En effet, la création des menus de ce jeu m'a poussé à d'abord réaliser un travail de graphiste afin de réaliser l'arrière-plan des menus dans un style minimaliste et moderne. J'ai ensuite pu mettre en place un code permettant de rendre ces menus interactifs. La mise en place d'autre composantes du projet m'a permis de me familiariser avec le langage JAVA, notamment du fait de la recherche des solutions envisageable afin de mener à bien ce projet.

Quentin LACOMBE :

La réalisation en binôme de ce projet m'aura permis de progresser dans la programmation en Java, et surtout plus largement, de progresser dans la programmation orientée objet. J'ai pu me familiariser avec ce concept, mais aussi renforcer mes compétences et notamment dans l'utilisation de plusieurs classes, l'utilisation de l'héritage (avec lequel je n'étais pas assez à l'aise), leur instanciation et leur « cohabitation ». Le Makefile étant bien différent de celui du projet en C89, sa réalisation m'a permis de voir une autre façon de réaliser ce fichier plus qu'utile. Mais tout comme le projet précédent, je pense que le fait de travailler en équipe m'a surtout permis de me rapprocher encore plus des conditions de travail en entreprise.