

Démineur en mode graphique



1 – Présentation du TP

Objet du TP

Ce TP a pour objectif d'implémenter, avec la technologie Swing, une interface graphique pour le jeu de démineur implémenté dans la première partie.

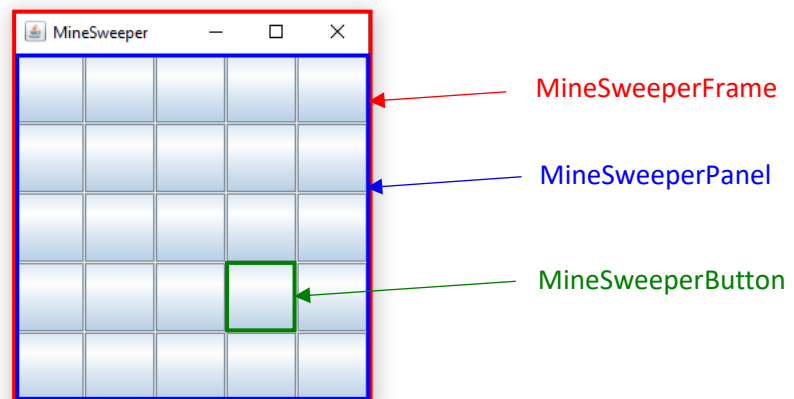
Ensuite, vous pourrez améliorer cette interface en ajoutant des icônes, des menus, etc.

2 – Démarrage du TP

Ouvrez le projet créé dans la première partie de ce TP.

3 – Interface minimale

L'objectif de cette partie est d'implémenter l'interface minimale suivante :



Nous vous conseillons d'implémenter les classes suivantes :

Classe	Superclasse	Description
MineSweeperFrame	JFrame	Fenêtre principale
MineSweeperPanel	JPanel	Panel principal représentant la grille du démineur.
MineSweeperButton	JButton	Bouton associé à une case de la grille.

Création de l'interface graphique

Créez les différents composants graphiques décrits ci-dessus, et disposez-les les uns dans les autres pour obtenir une apparence conforme à l'illustration.

Indications :

- Le programme principal doit créer deux instances : une de la classe `MineSweeper` implémentée dans le TP précédent et une de la classe `MineSweeperFrame`
- La grille du panel principal `MineSweeperPanel` doit s'adapter aux dimensions de celles de l'instance de `MineSweeper`

Affichage des coordonnées d'un bouton au clic

Faites en sorte que le jeu affiche les coordonnées d'un bouton (ligne, colonne) lorsqu'on clique dessus.

Éléments de réflexion :

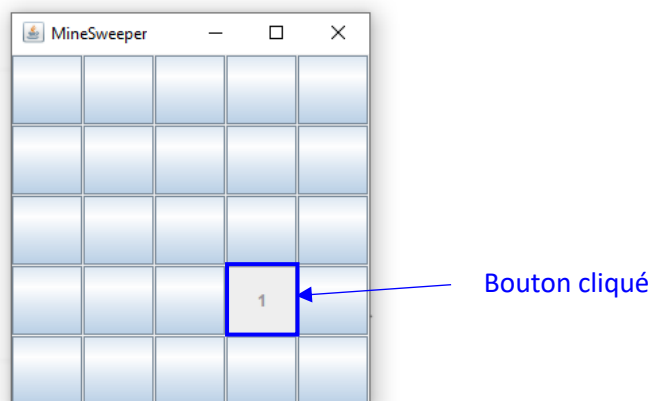
- Comment identifier le bouton sur lequel on a cliqué sans les parcourir tous ?
- Comment retrouver les coordonnées associées à un bouton ?

Dévoilement d'une case au clic

Faites en sorte que le clic sur un bouton dévoile la case correspondante.

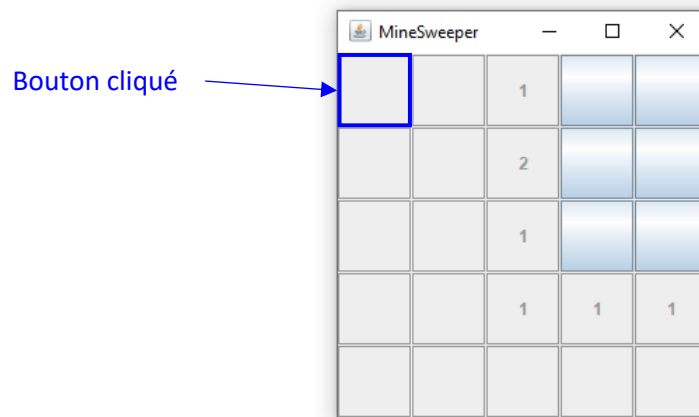
Lorsqu'une case est dévoilée :

- On fixe le texte du bouton à la représentation sous forme de chaîne de caractères de la case associée (donnée par la méthode `getCellSymbol()` de la classe `MineSweeper`)
- On désactive le bouton avec la méthode `setEnabled()`, ce qui lui donne un aspect grisé.



[Optionnel] Dévoilement en cascade

Si vous avez implémenté le bonus « Dévoilement en cascade des cases » dans le TP précédent, alors vous devez mettre à jour toutes les cases dévoilées dans l'interface, comme illustré ci-dessous :



[Optionnel] Marquage des cases

Si vous avez implémenté le bonus « marquage des cases » dans le TP précédent, faites en sorte que l'utilisateur puisse changer le marquage d'une case en réalisant un clic droit.



A chaque clic, le marquage de la case sera modifié selon le cycle suivant :

aucun marquage → marquage « mine » → marquage « indéci » → aucun marquage.

On reprendra les conventions de la vue texte, sauf pour l'absence de marquage où l'on ne mettra aucun texte (au lieu de « # »)

4 – Améliorations de l'interface

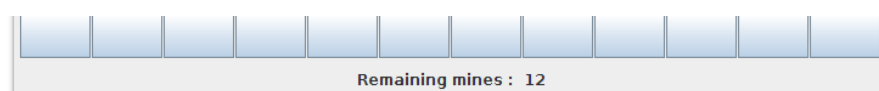
Icônes

Remplacez le texte des boutons par des icônes, par exemple  pour une mine,  pour le marquage d'une mine, etc.

Pour ce faire, vous devrez utiliser la classe [ImageIcon](#). Ce [tutoriel](#) sur les icônes peut vous aider.

Nombre de mines

Ajoutez une barre de statut en bas de la fenêtre qui indique le nombre de mines restantes.

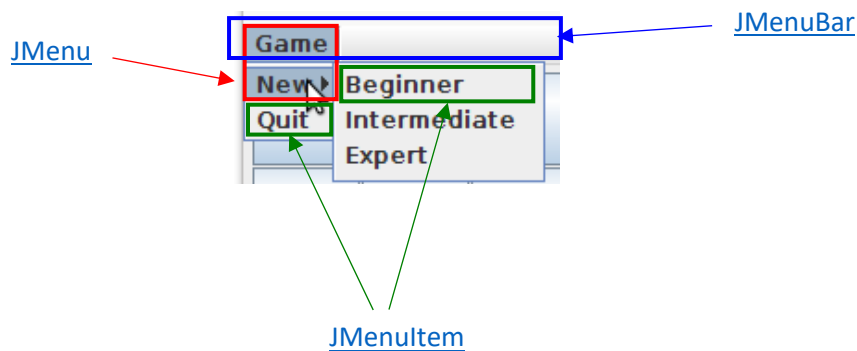


Menu

Créez un menu qui permet de quitter le jeu, ou démarrer une nouvelle partie avec différents niveaux de difficulté :

- « Beginner » : 10 mines dans une grille 9 x 9
- « Intermediate » : 40 mines dans une grille 16 x 16
- « Expert » : 99 mines dans une grille 16 x 30

On vous donne ci-après un aperçu du menu et les classes à utiliser :

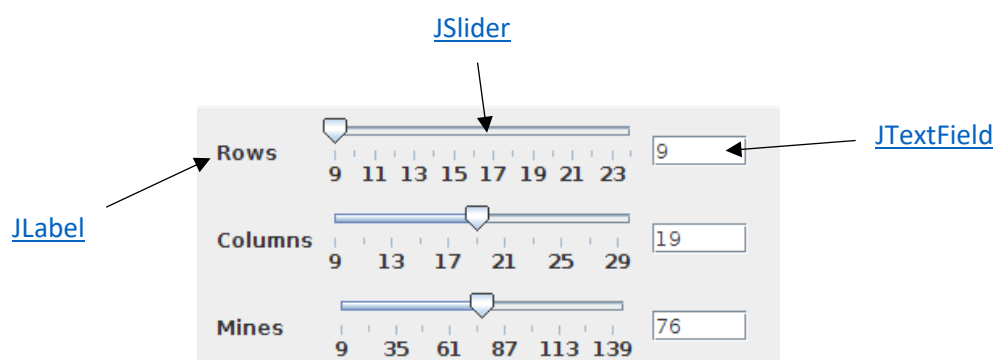


Ce [tutoriel](#) sur les menus peut vous aider.

Partie personnalisée

Ajoutez un item « Custom » au menu « New » pour permettre de définir une partie personnalisée.

Lorsque l'utilisateur clique sur cet élément, une fenêtre affichera le panneau de configuration illustré ci-après. Les noms des classes associées à chaque composant graphique est donné.



Les contraintes pour les différents paramètres sont les suivantes :

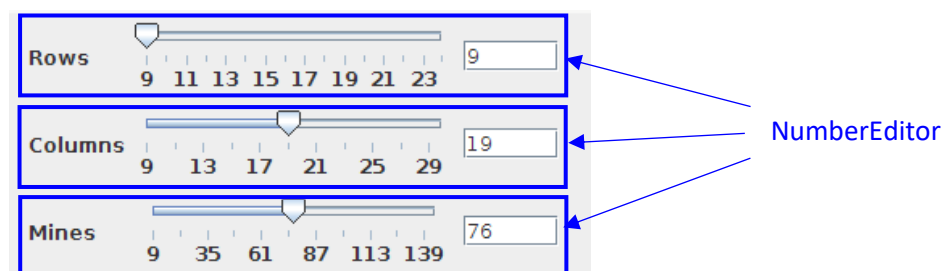
- Le nombre de lignes L doit appartenir à l'intervalle [9, 24]
- La nombre de colonnes C doit appartenir à l'intervalle [9, 30]
- Le nombre de mines M doit appartenir à l'intervalle [10, 0.85 * L x C]
- Les valeurs par défaut sont : L = 9, C = 19, M = 76

Faites en sorte que les valeurs représentées par le JSlider et le JTextField soient cohérentes, pour cela, vous devrez gérer les événements émis par ces composants :

- A chaque déplacement du curseur, un JSlider émet un [ChangeEvent](#) (que l'on peut écouter avec un [ChangeListener](#))
- A chaque validation du texte, un JTextField émet un [ActionEvent](#) (que l'on peut écouter avec un [ActionListener](#))

Pour aller plus loin : un composant générique

Comme vous pouvez le constater, le panneau de configuration est composé de 3 lignes similaires, chaque ligne permettant d'entrer la valeur d'un nombre.



Dans votre programme, le code associé à chacune de ces lignes est également similaire, que ce soit pour l'organisation des composants, la réaction aux événements, etc. En fait, c'est le même au nom des variables près.

On peut voir une ligne comme un composant graphique, que l'on pourrait appeler « NumberEditor ».

Plutôt que de copier/coller du code, on pourrait instancier 3 composants de type NumberEditor, avec des paramètres différents.

Ceci présente entre autres les intérêts suivants :

1. Ceci limite la redondance de code, ce qui le rend plus maintenable.
2. Le code est plus clair et plus concis : le lecteur n'est pas noyé dans les détails. Quand on lit le code du panneau principal, on voit l'instanciation de 3 NumberEditor et c'est tout !
3. Le composant NumberEditor est générique et répond à un besoin commun : il pourra être réutilisé dans d'autres projets. C'est la philosophie de la programmation orientée objet !