TP4 - Barre de chargement et jeu intégré

Le but de ce TP est de mettre en place une barre de chargement basique qui simulera le temps de préparation du café, puis d'ajouter un jeu Morpion orienté IHM SAÉ.

Ajout de la barre de chargement sur les machines à café

Dans l'interface de machine à café, ajoutez les attributs suivants :

- `loadingBar` de type `JProgressBar`
- `loadingTimer` de type `Timer`
- `loadingPourcentage` de type `Integer`

Rajoutez la barre de chargement à votre interface dans `initComponents`, elle affichera un chargement en pourcentage donc de 0 à 100, et elle sera cachée par défaut.

Ajoutez une méthode `loading` qui se chargera de simuler le temps de préparation du café :

- initialisez `loadingPourcentage` à 0
- instanciez `loadingTimer`
- ajouter au timer une tâche à répéter :
  - utilisez la méthode `scheduleAtFixedRate` avec `delay` à 10 et `period` à 100
  - lui transmettre une instance de `TimerTask` dont vous surchargez la méthode `run` (`@Override public void run…`)
  - pour chaque exécution du `run` :
    - mettez la valeur de `loadingPourcentage` dans la barre de chargement
    - si le chargement est à 100 %, exécutez la méthode `cancel` afin de mettre fin au timer
    - incrémentez le pourcentage de chargement

Testez.

Ajout du jeu Morpion

Pour simuler le jeu nous allons utiliser les notions de **tours** et de **parties**. Une partie est une suite de plusieurs tours qui commence sur un plateau vide, un tour est le fait de générer l'affichage du plateau et de cliquer sur une case. De cette façon vous allez redessiner chaque case à chaque tour, ce qui permettre d'y dessiner des croix (avec des traits) et des cercles, éléments qui permettent aussi de dessiner des graphs.

Créez une nouvelle classe `Morpion` qui étend de `JFrame`.

Cette classe contient un `JButton` "Réinitialiser le jeu", une liste de `JPanel` appelée `cells`, une liste de `Boolean` appelée `states` et un boolean appelé `cross`.

Ajoutez un constructeur vide ainsi que les méthodes `beforeInitComponents`, `initComponents` et `initEventListeners`.

Le constructeur se chargera de mettre un titre à la fenêtre, d'appeler `DSIPOSE_ON_CLOSE` sur la croix, de centrer la fenêtre, d'appeler `beforeInitComponents`, `initComponents`, `initEventListeners`, `pack` puis d'afficher fenêtre.

`beforeInitComponents` doit initialiser `cross` à true pour indiquer que le premier joueur joue la croix, initialiser `states` avec une liste vide puis ajouter la valeur null 9 fois dedans afin d'indiquer qu'aucune case ne contient une croix ou un cercle. Cette fois le `for (int i = 0; i… )` est autorisé.

`initComponents` instancie le bouton "Réinitialiser le jeu", l'ajoute dans un nouveau panel, ajoute un peu d'espace autour de ce panel puis l'ajoute en haut de l'écran. Ensuite iel instanciez `cells` ainsi qu'un nouveau panel avec un `GridLayout` de 3 lignes, 3 colonnes, 25 de `hgap` et 25 de `vgap` pour les cellules. Puis pour chaque élément de `states`, récupérez un `JPanel` avec une fonction `buildCell` qui prend en paramètre l'indice de la case (dans `states`), ajoutez ce panel à vos cellules, ajoutez-le à `cells`, et ajoutez-le au milieu de votre fenêtre. Enfin, ajoutez un peu d'espace sur les bordures de votre fenêtre.

`buildCell` prend en paramètre l'index de la case et retourne le `JPanel` qui correspond à la croix, au cercle ou à la case vide en fonction de l'état du plateau. Si la case est vide (null), instanciez un nouveau `JPanel`. Si la case contient une croix, récupérez le `JPanel` avec une fonction `buildXorOCell` qui prendra en paramètre true, sinon false. Mettez une bordure et une dimenstion de 150*150 et retournez la cellule.

`buildXorOcell` retourne un `JPanel` et prend en paramètre un booléen `modeX`. Il retourne un nouveau `JPanel` dans lequel vous devez surcharger la fonction `paintComponent`. Après le `super`, récupérez les coordonnées du centre du composant puis calculez les coordonnées pour tracer une croix ou un rond au milieu de la case, de largeur 40. Si vous êtes en mode X, tracez 2 lignes qui se croisent, sinon tracez un cercle et retournez le panel.

`nextRound` enlève tous les composents de la fenêtre puis appel `initComponents`, `initEventListeners` et `pack`.

`playAgain` appel `beforeInitComponents`, réinitialise `cross` et appel `nextRound`.

`initEventListener` ajoute un listener sur le bouton "Réinitialiser le jeu" et appel `playAgain` en cas de clique, puis pour chaque cellule dont le state est null, ajoute un listener sur le clique qui met `cross` dans `states` à l'indice de la case, puis inverse `cross` et appel `nextRound`