## 

## **INF-111 Travail pratique #2**

## **Automne 2014 Travail en équipe de quatre(4)**

**Méthode : Par courriel**

C’est quoi ton dessin (jeu de dessin caché) ?

# Projet

Réaliser et implémenter la conception d’un jeu de dessin caché. Ce travail est nécessaire à une application de style graphique (GUI) qui sera réalisé dans le prochain travail pratique.

# Objectfs du tp2

* Mettre en application les concepts orientés-objet de classe et d’encapsulation dans un projet concret de programmation.
* Utiliser une collection de type liste implémentée avec chaînage dynamique.
* Développer des compétences transversales :
  + Utilisation de fichiers binaires entre applications.
  + Développement de méthodes de résolution de problèmes.

1. **Description du problème**

Nous allons reproduire un jeu papier qui consiste à découvrir un dessin caché sous une grille carrée[[1]](#footnote-1) avec des indices sur le nombre de blocs coloriés. Un bloc étant une ou plusieurs cases consécutives.

Pour le dessin original suivant :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Le programme offre les indices suivants :

2 2 2

3 1 2 1 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

1

3

2 2

1 1 1

5

L’utilisateur doit sélectionner des cases, une par une, et trouver le dessin. Il dispose de trois vies pour un 5 X 5, quatre vies pour un 10 X 10 et cinq vies pour un 15 X 15. Il perd une vie s’il clic sur une case qui ne fait pas partie du dessin original. S’il clic sur une bonne case, elle est dessinée. Lorsqu’un bloc a été complètement découvert, l’indice affiché correspondant (en ligne et/ou en colonne) devient négatif.

Exemple :

2 2 2

3 -1 2 -1 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

1

3

2 2

1 1 1

-5

La partie se termine si l’utilisateur n’a plus de vie, que le dessin est découvert en entier ou que l’utilisateur annule en entrant 0 pour la ligne ou la colonne.

1. **Déroulement du programme**

Le programme offre de récupérer un fichier avec une extension .dsn. Ce fichier doit être généré par l’application creerDessin.jar fourni avec cet énoncé.

Votre application calcule les indices pour chaque ligne et chaque colonne à partir du fichier .dsn et les affiche à l’écran avec une grille vide. L’utilisateur entre le numéro de ligne et de colonne qu’il veut découvrir. Si la case ne fait pas partie du dessin original, l’utilisateur et avisé qu’il perd une vie sinon la case est coloriée et l’indice ajusté selon qu’un bloc est complété ou non. Si l’utilisateur entre 0 pour la ligne ou la colonne, c’est qu’il veut annuler. Le programme se termine alors exactement comme si l’utilisateur n’a plus de vie ou qu’il trouve le dessin.

1. **Conception**

Pour que l’interaction fonctionne entre les différentes applications (la vôtre et la mienne), vous devez respecter la description des API que je vous donne. Cependant, certains choix d’implémentation sont laissés à votre discrétion. Vous recevrez les consignes à mesure de la description des différentes parties à réaliser.

1. **creerDessin.jar**

Idéalement, cette application se trouve dans le même répertoire que votre projet. Double‑cliquez dessus pour la démarrer. Elle vous permet de créer des dessins. Vous ne pouvez qe créer un nouveau dessin ou à en récupérer un déjà existant. Vous cliquez sur les cases jusqu’à ce que vous ayez le dessin voulu et vous le sauvegardez (ou non).

1. **Remise et évaluation**

C’est un travail à remises multiples mais seule la dernière remise sera notée. Vous devez faire valider votre travail à chaque semaine en m’envoyant votre code commenté (pas besoin de faire les commentaires Javadoc. Les commentaires de stratégie seulement sont exigés).

Vous êtes en équipe de quatre étudiants et c’est la responsabilité de chaque membre d’être impliqué et de participer à chacune des étapes. D’ailleurs la note de chaque membre ne sera pas nécessairement égale à la note du travail selon l’évaluation des pairs qui devra être remise avec chaque étape. Un membre qui ne s’implique pas peut se faire exclure de l’équipe par les autres membres. Il devra réaliser le travail seul ou avec un autre membre exclus d’une autre équipe. Aucune prolongation n’est possible.

Des questions précises sur les travaux 2 et 3 feront partie de l’examen final. À ces questions, tous les membres recevront la moyenne de l’équipe. En d’autres termes, tout le monde doit comprendre.

**PREMIÈRE PARTIE**

**Remise : jeudi 6 novembre (13h00 max)**

1. **Introduction**

Nous commençons le projet en récupérant un dessin sur disque. Vous devez écrire la classe GrilleDessin avant de pouvoir écrire la procédure de récupération de la classe UtilitaireFichier.

1. **GrilleDessin**

Cette classe vous est partiellement fournie. Elle contient deux attributs (interdit d’en ajouter ou en enlever):

* + Un tableau 2D de booléens (si une case est à ***true***, c’est qu’elle fait partie du dessin)
  + Le nom du dessin (sert à reconnaitre le dessin).

Voici les méthodes fournies (voir le code pour détails) :

* Le constructeur : Il reçoit la taille de la grille et une chaîne représentant le nom qui identifie le dessin.
* Les accesseurs : estColorie, getTaille
* Le mutateur : colorieCase

Vous devez écrire la méthode suivante :

* + estValide() : Retourne si le dessin est valide. Un dessin est invalide s’il a des lignes ou des colonnes avec aucun bloc sauf sur le contour. Autrement dit, entre les lignes vides du haut et du bas et les colonnes de gauche et droite, il n’y a pas d’espace sans au moins une case coloriée. Même chose en colonne. Les cases vides du contour ne font pas partie du dessin à découvrir.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Dessin valide Dessin invalide Dessin invalide

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

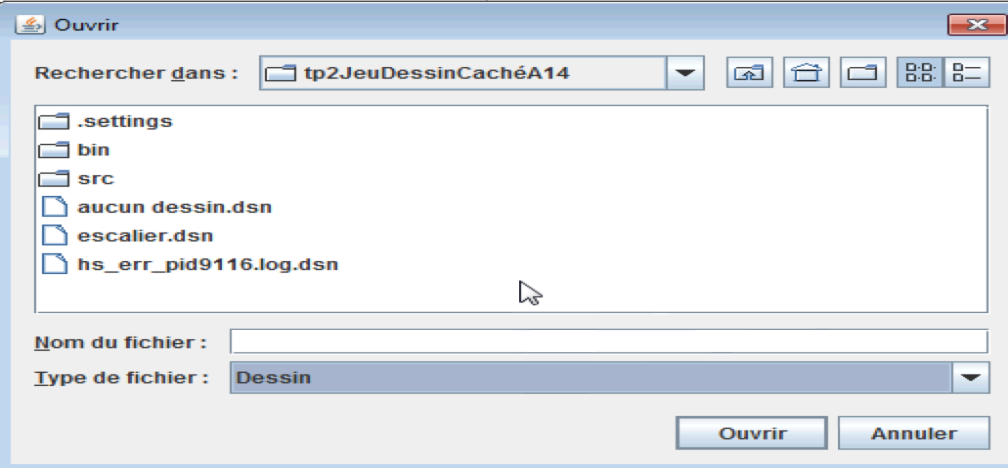
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

1. **UtilitaireFichier**

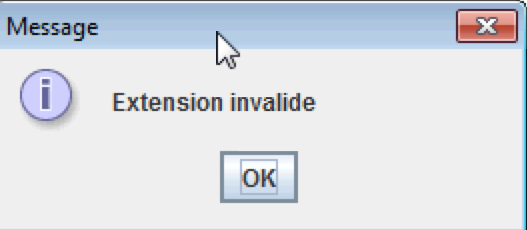
Vous devez écrire une méthode static dans une classe utilitaire qui permet de récupérer un fichier avec l’extension .dsn créé par l’application creerDessin.jar. Consultez les API des classes :

* JOptionPane
  + showMessageDialog
* FileNameExtensionFilter (constructeur seulement)
  + C’est pour filtrer les choix de fichiers offerts
* JFileChooser
  + Constructeur, addChoosableFileFilter, showOpenDialog, getSelectedFile et getAbsolutePath.
* FileInputStream (constructeur)
* ObjectInputStream
  + Read()[[2]](#footnote-2) et close().
* Utilisez les constantes static des différentes classes lorsque c’est possible et pertinent (exemple : JFileChooser.APPROVE\_OPTION*,* OptionPane.CANCEL*\_*OPTION, JOptionPane.NO\_OPTION, …)

Vous devez voir cette fenêtre à l’aide de JFileChooser (les noms de fichier peuvent varier):



Si l’utilisateur n’annule pas et choisis un fichier, vous devez l’aviser si l’extension n’est pas valide (.dsn) et réafficher la fenêtre de saisie tant que ce n’est pas valide ou que l’utilisateur quitte. Exemple de message :



**Général** : Pour changer le répertoire courant par celui du fichier qui vient d'être lu (sauve du temps lors des tests) : System.*setProperty*("user.dir",fc.getSelectedFile().getParent());

1. **UtilitaireAffichageConsole**

Je ne vous demande pas de remettre le code d’affichage du dessin mais si vous voulez tester facilement la validité de vos méthodes, écrivez une méthode static qui affiche une GrilleDessin. Si une case est à true , vous mettez un X sinon vous mettez un espace. Autrement, il faut se servir du débogueur pour valider votre code.

1. **Déroulement du travail (idéalement)**

Vous déterminez en équipe quelle est la stratégie pour valider le dessin dans GrilleDessin. Lorsque vous êtes d’accord, chaque membre fait une version béta. Les versions sont présentées aux autres membres de l’équipe et la meilleure solution est retenue pour approbation. Vous répétez le tout pour la procédure de récupération dans UtilitaireFichier.

Vous me remettez la version d’équipe dans un fichier compressé. Pas besoin de .pdf, juste GrilleDessin.java, UtilitaireFichier.java et DemarrerJeuCache.java (contient seulement le main).

Je m’attends à au moins 6 heures de travail par semaine par membre de l’équipe pour ce travail. Cela n’inclut pas les exercices de laboratoire qu’il y aura à réaliser en préparation à la réalisation du dernier travail pratique qui sera totalement orienté‑objet.

1. **Critères d’approbation**

* Efficacité des algorithmes (pas de code inutile).
* Découpage en sous-programmes.
* Commentaires de stratégie clairs et pertinents.
* Conformité avec l’énoncé (exécution).
* Respect des normes Java (sans le Javadoc).

**Conseils :**

* Écrivez le commentaire de stratégie avant d’écrire le code.
* Écrivez le code correctement du premier coup comme si c’était juste avant la remise (identificateurs significatifs, utilisation de constantes, aération, indentation, 80 col. Max et découpage en SP, …). Cela vous sauvera du temps si votre version est choisie. Formatez le code à l’aide de ctrl-i (sous Eclipse).
* Ne commencez pas à coder si vous n’avez pas une stratégie implacable.

Tant que vous n’avez pas l’approbation, vous ne recevez pas la partie suivante. Il y a trois parties comptant pour 33.3% de la note chacune. Seule la version finale, comptant les trois parties, remise le 20 novembre sera notée.

VOUS NE DEVEZ PAS CONSULTER LES AUTRES ÉQUIPES.

Les seules questions auxquelles les auxiliaires et moi répondront sont sur les concepts de programmation vus en classe, la qualité de votre code et non sur le fonctionnement des classes de Java. **Vous devez apprendre à utiliser les API de Java sans aide.**

Bon travail!

1. Les gilles peuvent être de 5, 10 ou 15 lignes (décision arbitraire). [↑](#footnote-ref-1)
2. Il n’y a qu’un objet de la classe GrilleDessin dans un fichier [↑](#footnote-ref-2)