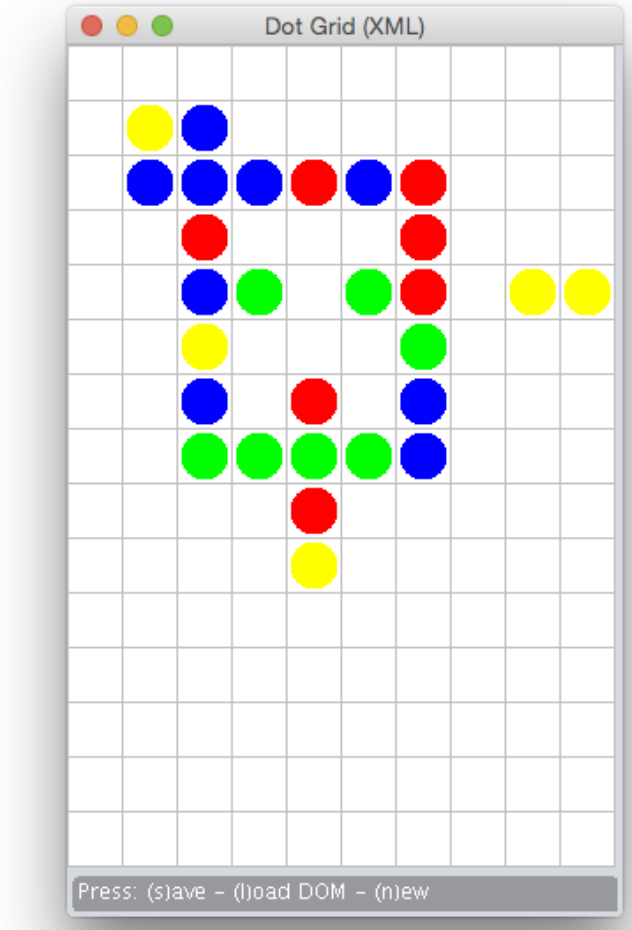
Série D : Traitement de fichiers

Table des matières

Exercice D.1: Dots

Réalisez le petit exercice **Dots** décrit dans la partie XML du cours.



Pour avancés : Utilisez la bibliothèque **Gson** pour sauvegarder et lire les données de la grille dans un fichier **JSON** *(JavaScript Object Notation).*

Exercice D.2: FractionList

Reprenez le code de l'exercice **FractionList** et ajoutez les méthodes suivantes :

`public void saveToDataFile(String fileName)`	sauvegarde dans un fichier de données
`public void loadFromDataFile(String fileName)`	lecture dans un fichier de données
`public void saveToTextFile(String fileName)`	sauvegarde dans un fichier texte
`public void loadFromTextFile(String fileName)`	lecture dans un fichier texte
`public void saveToObjectFile(String fileName)`	sauvegarde dans un fichier d'objets
`public void loadFromObjectFile(String fileName)`	lecture dans un fichier d'objets
`public void saveToXmlFile(String fileName)`	sauvegarde dans un fichier XML
`public void loadFromXmlFile(String fileName)`	lecture dans un fichier XML

Examinez le contenu des fichiers à l'aide d'un éditeur de textes.

Exercice D.3: QuickClick

Reprenez le projet **QuickClick** (Ex. de révision).

- Ajoutez un deuxième **`Timer timeOutTimer`** pour limiter la durée d'un jeu. Et ajoutez à **`DrawPanel`** un bouton pour démarrer un nouveau jeu.
- Veillez à ce que la fiche principale ne puisse pas être redimensionnée (→ propriété **`sizeable`**).
- Ajoutez la possibilité de sauvegarder les 10 meilleurs scores obtenus avec les noms des joueurs qui les ont obtenus.

  Ajoutez la ou les classes nécessaires et sauvegardez les données dans un fichier texte. Le fichier doit être chargé et sauvegardé automatiquement sans intervention de l'utilisateur. Il ne doit pas y avoir de message d'erreur, si le fichier n'existe pas encore.

Exercice D.4: Vigenère

Reprenez le projet **Coder** (Vigenère) et ajoutez une possibilité pour encrypter et décrypter des fichiers binaires octet par octet selon la méthode de Vigenère.

Pour avancés :

Comparez la performance du programme avec et sans mémoire tampon (BufferedInput/OutputStream) en dé-/chiffrant un fichier d'une taille plus grande qu'un Mo.

Exercice D.5: MiniDraw

Reprenez le projet **MiniDraw** (fin du cours de 2GIN). Faites deux copies du projet. Ajoutez aux copies la possibilité de lire et de sauvegarder les dessins :

Copie 1 : dans des fichiers texte,

Copie 2 : dans des fichiers d'objets.

Conseil pour les fichiers texte :

Mémoriser les données des figures ligne par ligne, en écrivant une ligne pour chaque figure (p.ex. à l'aide d'une nouvelle méthode **`toFileString`**). Il peut aussi être utile, d'écrire dans **`Figure`** une méthode statique (p.ex. **`newFromString`**) qui sait créer une figure à partir d'une ligne de texte.

Essayez deux variantes pour lire une ligne de texte dans un fichier (à l'aide de **`BufferedReader`** et à l'aide de **`Scanner`**). Une exception est levée, si le format n'est pas correct.

Utilisez des dialogues de lecture et de sauvegarde. Définissez et utilisez des filtres pour les différents types de fichier.

Définissez et utilisez un attribut '**currentFile**' dans lequel se trouve à tout moment le nom et le chemin du dernier fichier qui a été lu ou écrit. Utilisez **currentFile** pour ouvrir les dialogues toujours en montrant le chemin de **currentFile**. Initialisez **currentFile** par le chemin du programme.