

<p align="center">Cours 420-202-RE Traitement de données orienté objet Hiver 2021 Cégep Limoilou Département d'Informatique</p>	<p align="center">Tp3 (6 %) Jeu de Mémoire « Souviens-toi ... »</p>
--	--

OBJECTIFS

- ♦ Utiliser des classes liées par le principe de l'héritage et profiter du polymorphisme.
- ♦ Tester unitairement des classes
- ♦ Utiliser les tableaux dynamiques et des matrices.
- ♦ Utiliser et compléter une classe.
- ♦ Intégrer des classes dans un projet ayant une interface graphique.

DURÉE DU LABORATOIRE :

- ♦ 2 semaines sont consacrées à ce travail. Le travail est fait en équipe de 2.

ACTIVITÉS À RÉALISER

Dans ce laboratoire, vous allez créer une petite application permettant de jouer au jeu de mémoire avec les formes que vous avez déjà créées (référence Tp 2).

IMPORTANT : Avant de commencer votre travail vous allez créer une copie du projet du « Tp 2 » dans un nouveau projet « Tp 3 » et vous allez lui donner le nom suivant « vos noms – Tp3 ».

Dans le nouveau projet « Tp 3 », copier le dossier images et dans le dossier src copier le package gui disponibles sur Léa dans le dossier « TP 3 ».

1. Pour commencer, vous allez créer une classe qui va permettre de placer dans un vecteur dynamique, différentes formes de différentes couleurs. À partir du diagramme de classe fourni en annexe, créez une classe « **VecteurFormes** » dans le package « **formes** » basée sur l'interface « **ManipulerVecteur** ». Cette nouvelle classe possèdera, entre autres, les attributs et comportements suivants :

- Un attribut **vecteur** (ArrayList) de formes qui pourra contenir le nombre de formes voulues;
- Lors de la construction du **VecteurFormes** le vecteur restera vide.
- Une méthode « **remplir** » qui permet de remplir le vecteur avec des formes générées de la façon suivante :

Tant que le nombre de formes voulues (reçu en paramètre) n'est pas atteint on crée une forme à la fois, pour chaque couleur disponible (rouge, vert, bleu, jaune, noir, orange) combiné à chaque type de formes disponibles (Cercle, Rectangle et Triangle). Quand toutes les combinaisons sont épuisées on recommence. (Assurez-vous que le nombre de formes voulu est valide sinon une exception de type `ArrayIndexOutOfBoundsException` sera lancée)

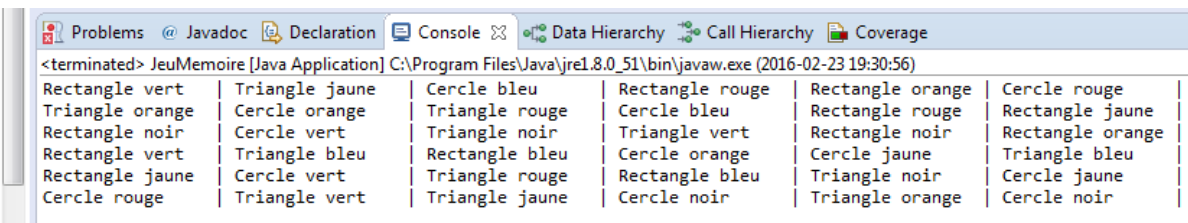
Ex. Cercle rouge, Rectangle rouge, Triangle rouge, Cercle vert, Rectangle vert, Triangle vert, etc.

La taille des formes n'est pas importante.

- Une méthode « **melanger** » pour mélanger aléatoirement le contenu du vecteur de formes. Vous devez concevoir votre propre algo pour mélanger vos formes sans utiliser les fonctions dédiées de l'offre de Java.
- Une méthode « **trier** » pour trier les éléments du vecteur par forme et par couleur (utilisez votre méthode « **compareTo** » déjà développée). Vous devez obligatoirement implémenter un des algos de tri vus en classe (sélection, insertion ou à bulles).
- Une méthode « **getVecteur** » permettant d'obtenir une référence sur le vecteur de formes.
- Redéfinir la méthode **toString** pour vous permettre de visualiser le contenu complet de votre vecteur de formes (une par ligne, nom et couleur seulement). Utilisez toStringCourt de Forme.
- **IMPORTANT** : Cette classe est un contenant pour manipuler des formes et ne doit pas être associée au fonctionnement ou aux comportements directs du jeu final. Elle pourrait très bien servir pour d'autres applications ou jeux qui ont besoin d'un ensemble de formes.

IMPORTANT : Dans la classe Forme vous devez ajouter une nouvelle méthode toStringCourt(). Cette dernière va permettre d'afficher le contenu du vecteur sans modifier la méthode « **toString** » des formes enfants. Retourne le nom de la forme et sa couleur uniquement (**ex.** Cercle bleu, Cercle rouge). Voir le diagramme de classes.

2. Dans la classe **VecteurFormesTest** du package **tests**, vous allez tester unitairement toutes les méthodes publiques de votre classe **VecteurFormes** à l'aide de JUnit.
3. Vous devez documenter votre classe **VecteurFormes** avec la JavaDoc.
4. À partir du diagramme de classes fourni en annexe, créez dans un nouveau package « **jeu** » la classe « **JeuMemoire** » qui implémente l'interface **Memorisable** et qui permet de définir tous les comportements nécessaires à la gestion du jeu de mémoire « **Souviens-toi ...** », complétez la classe selon les instructions suivantes :
 - Penser aux attributs nécessaires pour gérer le jeu. Par exemple, une matrice qui contient des objets « **Forme** ». Le jeu change de niveau à chaque fois que le joueur réussit. Ainsi, au niveau 1, le jeu donne 3 formes à reproduire, au niveau 2, 4 formes et ainsi de suite jusqu'au niveau 6 avec 8 formes. Le nombre de niveau ne dépasse jamais 6. La classe possède aussi une liste de points contenant les coordonnées x,y de chaque forme choisie au hasard à chaque niveau de jeu.
 - Le **constructeur** permet de préparer le vecteur de formes (le remplir et le mélanger) et de préparer la grille du jeu (remplir la matrice à partir du vecteur de formes).
 - **Important** : Une méthode **toString()** permet de retourner une chaîne contenant les éléments de la grille sur 6 lignes et 6 colonnes. On veut ainsi pouvoir afficher la matrice pour faire des tests visuels. Se sert des méthodes ajouterEspaces et toStringCourt. La méthode ajouterEspaces ajoute des espaces à la chaîne reçue pour qu'elle soit de la longueur reçue (ici 17).



Rectangle vert	Triangle jaune	Cercle bleu	Rectangle rouge	Rectangle orange	Cercle rouge
Triangle orange	Cercle orange	Triangle rouge	Cercle bleu	Rectangle rouge	Rectangle jaune
Rectangle noir	Cercle vert	Triangle noir	Triangle vert	Rectangle noir	Rectangle orange
Rectangle vert	Triangle bleu	Rectangle bleu	Cercle orange	Cercle jaune	Triangle bleu
Rectangle jaune	Cercle vert	Triangle rouge	Rectangle bleu	Triangle noir	Cercle jaune
Cercle rouge	Triangle vert	Triangle jaune	Cercle noir	Triangle orange	Cercle noir

- La méthode **getNomForme** permet de retourner le nom et la couleur (sans espace) de la forme située à la position x, y (ex. CercleVert : correspond au nom de l'image)

- La méthode **jouerOrdi** permet de créer une liste de points (voir classe Point dans API) **uniques** pris au hasard dans la matrice. Le nombre de points générés dépend du niveau où le joueur est rendu : (niveau + 2). Se sert de la méthode **choisirForme()** qui retourne un Point dont les coordonnées (x et y) sont pris au hasard dans les limites de la grille de jeu. Pour savoir si un Point existe déjà dans le vecteur (donc pas unique), vous pouvez utiliser une méthode offerte par Java.
- La méthode **jouerHumain** retourne vrai si la coordonnée x, y de la forme à deviner est celle reçue en paramètre.

Astuce : S'il y a des points dans le vecteur, enlever le premier point et vérifier si la coordonnée reçue est égale à la coordonnée de ce point. Ainsi le premier point correspond toujours à celui à deviner. Comme cela vous n'aurez pas à contrôler l'ordre.

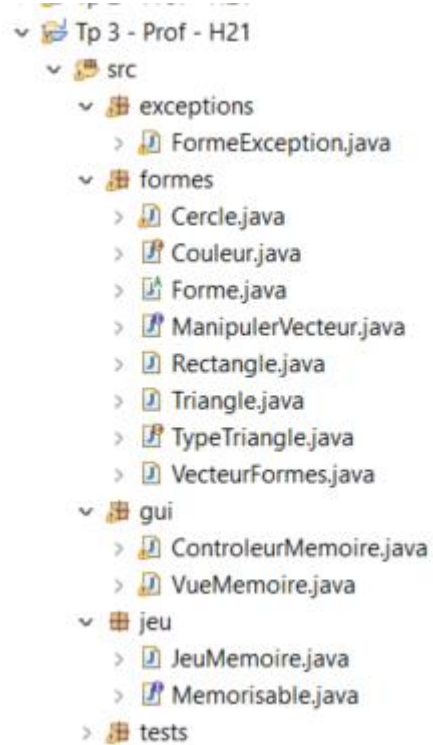
- Il faudra les **accesseurs et mutateurs** nécessaires à la gestion de votre jeu.
 - Un dossier d'images est fourni dans le dossier du Tp. Attention, le nom des images est construit ainsi : **nomFormeCouleurForme.png**, il ne faut surtout pas le changer.
 - Pour vous aider, vous pouvez voir l'exécution de l'application en demandant à l'enseignant.
 - Portez attention au découpage de votre classe, bien respecter 1 méthode = 1 action, si une méthode doit effectuer plusieurs actions, divisez-la en petites méthodes privées.
5. Dans la classe **JeuMemoireTest** du package **tests**, Vous allez tester unitairement toutes les méthodes publiques de votre classe **JeuMemoire** à l'aide de JUnit.
 6. Documentez votre classe **JeuMemoire** avec la JavaDoc.
 7. À partir de la classe **ControleurMemoire** fournie, le point d'entrée du jeu, (il n'y a rien à modifier dans cette classe), vérifiez que votre jeu fonctionne (si vous avez bien respecté les interfaces tout devrait être fonctionnel). Ajustez au besoin votre classe **JeuMemoire** ou **VecteurFormes**. La classe **VueMemoire** permet de créer les éléments nécessaires à l'interface graphique (vous n'avez rien à modifier dans cette classe).

À REMETTRE :

- Code source du projet respectant les consignes, utilisant efficacement les packages, la programmation orientée objet, l'héritage et les exceptions.
- Votre code documenté avec la JavaDoc générée pour les packages jeu et formes (toutes les méthodes : cocher private).
- Les tests unitaires (JUnit) pour les classes demandées (VecteurFormes et JeuMemoire).
- Projet complet compressé (**.zip**) dans un fichier du nom « vos noms – Tp3.zip » et déposé sur Léa.

ANNEXES : PACKAGES et DIAGRAMME DE CLASSES

Packages :



Critères de correction :

Respect des consignes et JavaDoc	/ 10
Fonctionnement du jeu	/ 20
Votre code	/ 40
Vos tests unitaires	/ 20
Tests du profs (conformité et fonctionnement)	/ 20
Total	/ 110

Diagramme de classes :

