Cégep André-Laurendeau

# TP4 : Space Invaders

## Algorithmique et programmation
## 420-235-AL

---

## 1. Objectifs

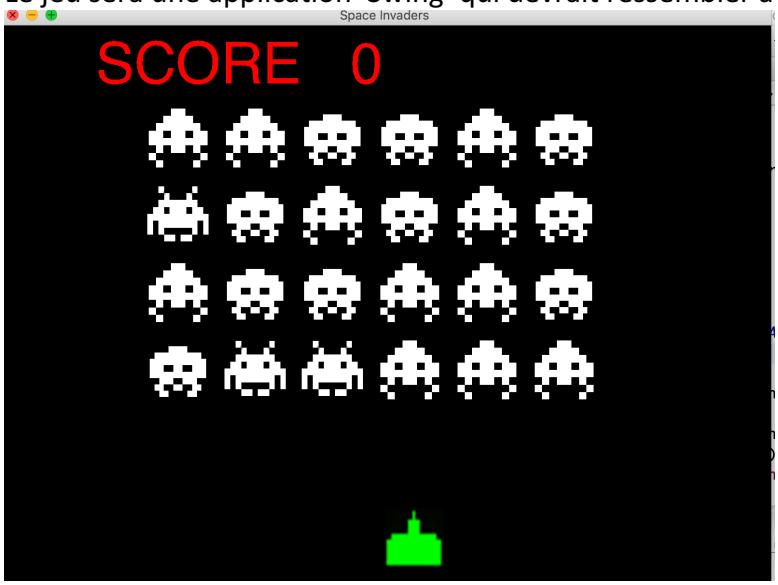
- Approfondissement de la compréhension de la programmation orientée objet
- Utilisation du concept d'héritage, d'interfaces, de classes abstraites, de classes internes
- Continuer de s'amuser!

---

## 2. Présentation du problème

Vous allez construire un jeu de 'space invaders'. Ce travail vous permet de mettre en pratique les acquis que vous aurez appris en classe notamment l'utilisation adéquate des concepts suivants : L'héritage, les interfaces, les classes abstraites, les classes internes, les différents mots-clés utilisés en java (public/private/protected, static, final)

Le jeu sera une application 'Swing' qui devrait ressembler à l'image ci-dessous



Voici la fonctionnalité attendue de votre jeu

- Avoir au moins 2 rangées de monstres
- Le déplacement des monstres doit se faire de gauche à droite puis de droite à gauche de quelques pixels à la fois.
- Quand les monstres changent de direction (de droite vers la gauche et vice-versa) ils descendent d'une ligne.
- Un canon doit être dessiné sur la ligne du bas. Celui-ci doit pouvoir se déplacer latéralement soit à l'aide du clavier. (Les flèches gauches et droites)
- Vous devez permettre au canon de tirer des obus. Les obus seront dessinés sur l'écran du jeu. (La barre d'espacement devrait être utilisée pour tirer les obus)
- Si un obus touche à un monstre celui-ci disparait.
  - Moins il reste de monstres, plus la cadence de déplacement latéral des monstres accélère. (Optionnel)
  - Si le joueur réussis à détruire tous les monstres, il a gagné la partie.
  - Si un monstre se rend à la ligne où se trouve votre canon, le joueur a perdu.
  - Vous devez afficher le score à l'écran au fur et à mesure du déroulement du jeu.
  - À la fin de la partie, vous devez sauver le 'score' dans un fichier.
  - Les 'high scores' devraient apparaitre à la fin de chaque partie. (Optionnel)

## 3. Critères de correction

Vous êtes libre d'implanter le jeu comme vous le désirez. Par contre, j'aimerais voir une utilisation adéquate des différentes façons de faire de Java. (Classe abstraite, interface, bonne utilisation de static, protected, finale). Évidemment, le tout doit être fait 'clean-code'.

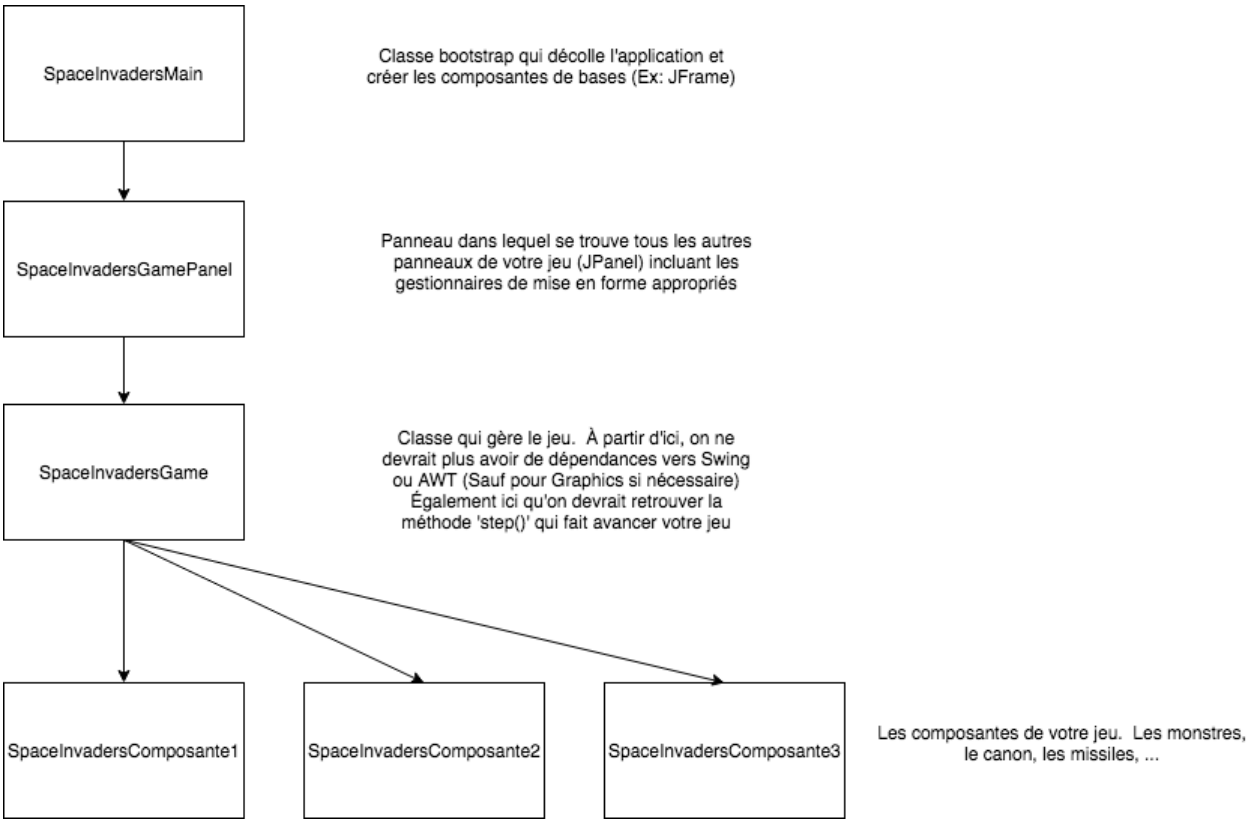
Voici le barème de correction qui sera utilisé

| | |
|---|---|
| **Qualité du code** | |
| Utilisation des concepts orientés-objet | /40 |
| Clean-code (bon découpage, noms significatifs, courtes méthodes) | /20 |
| **Fonctionnement** | |
| Déplacement et affichage des monstres | /8 |
| Déplacement et affichage du canon | /5 |
| Tirer des missiles du canon | /6 |
| Tirer des missiles des monstres | /6 |
| Conclure la partie (gagnant/perdant) | /5 |
| Afficher le score | /5 |
| Sauver le score dans un fichier | /5 |
| **Total** | /100 |

## 4. Suggestion d'architecture

Comme cela a été expliqué en classe, il est important que les dépendances de votre application soient unidirectionnelles, c'est-à-dire qu'elles aillent toutes dans le même sens. Par exemple, vos classes comportant la logique de votre jeu devrait être agnostiques de la présentation de votre jeu. Dans notre cas, cela veut dire qu'on ne devrait pas retrouver de classes appartenant à AWT ou à Swing dans vos classes d'objets métier. Une exception pourrait être d'avoir une méthode de 'paint()' qui recevrait un 'Graphics' en paramètre.

Suite à ces suggestions, je vous donne un exemple sur la façon dont vous pourriez structurer votre jeu.




Les flèches indiquent le sens des dépendances. Ainsi, la classe 'SpaceInvadersGame' ne devrait pas dépendre de Swing ou de AWT sauf exception pour Graphics si nécessaire.

## 5. Rappel des concepts Swing

Voici un rappel des concepts importants de Swing.

- Un JFrame, JDialog, JWindow est un 'container' de haut niveau. On ne devrait qu'en trouver 1 seul pour votre application
- Un ou plusieurs JPanel peuvent se trouver dans votre 'container' (JFrame) de haut niveau. N'oubliez pas d'utiliser les gestionnaires de mise en forme tel que montré en classe.
- Les composantes de plus bas niveau sont ensuite insérées dans vos JPanel.
- C'est dans la méthode 'paintComponent()' de vos panneaux qu'il faut penser à afficher vos dessins (rectancles ou images).
- C'est avec la méthode 'repaint()' que vos dessins (ou images) se font redessiner. Si on ne met aucun paramètre à 'repaint()', tout le canvas se retrouve alors dessiné. Vous devriez appeler 'repaint()' après l'appel de votre méthode 'step()' dans le panneau approprié.
- Un 'timer' avec une action devrait être le mécanisme privilégié pour cadencer l'exécution de votre jeu. La méthode 'step()' de votre jeu devrait y être appelée.
- Les intéractions avec l'usager (les touches de clavier pour diriger le canon et lancer des missiles) sont effectuées à l'aide des 'listeners' appropriés. N'oubliez pas que ces 'listeners' appartiennent au 'container'
- Les détections de collisions peuvent être faite à l'aide de la méthode 'intersects() de la classe Rectangle

## 6. Remise

N'oubliez pas que ce TP est à remettre le 15 Mai à 23h55. Comme il y a beaucoup de travail (je m'attends à ce que ça vous prenne autour de 20 heures pour le compléter), n'attendez pas à la dernière minute pour le faire. C'est votre TP certificatif, vous devez donc avoir plus de 50% pour ne pas échouer votre cours

POLITIQUE DE RETARD

Tout retard dans la remise d'un travail entraîne une pénalité de 10% par jour de retard jusqu'à concurrence de 5 jours de calendrier scolaire (donc, fin de semaine et congé non compris). Après cette date, la note de 0 sera attribuée au travail.

## 7. Mot de la fin

Bonne chance à tous! N'oubliez pas de vous amuser lorsque vous ferez ce travail. Il s'agit de faire un jeu après tout.