# Coding dojo

TP personnel (environ 2 h)

Source « https://www.math.univ-paris13.fr/~chaussar/ »

Le but général de l'exercice est de proposer un code flexible permettant d'afficher un personnage de jeu vidéo à l'écran.

## 1 Instructions générales

1. Téléchargez l'archive zip avec le code de base donné dans le dossier de la plateforme pédagogique.
2. Ouvrez le projet dans IntelliJ IDEA. Nul besoin de faire des paquets pour cet exercice. Tout est codé à la racine projet.
3. Exécutez le programme. Si le projet ne s'exécute pas réglez la variable `projectPath` de la classe `SystemValues` avec le dossier courant de votre projet : là où le programme trouvera le dossier `img` contenant les images.

## 2 Premier exercice

Dans un premier temps, on s'occupe de la partie multijoueur du code. Dans la classe `Main1`, trois `Player` sont créés et exécutés en concurrence. Ces trois `Player` tentent de se connecter à un `Server` (en le créant) puis demandent au `Server` de leur envoyer un `Character` afin de pouvoir joueur au jeu. Si tout fonctionnait bien, l'exécution de la fonction `start()` des `Player` (voir le Main) devrait afficher autant de `Character` qu'il y a de `Player` (quatre `Character` au maximum peuvent être créés par le `Serveur`), chaque `Character` ayant une couleur différente :

Si vous exécutez le `Main` (qui créé trois `Player`), vous obtiendrez bien l'affichage de trois `Character`, mais tous de même couleur. En effet, chaque `Player` créé son propre `Server`, chose que l'on souhaite éviter : il faut que chaque `Player` se connecte au même `Server`. À vous de modifier le code afin qu'un seul et même `Server` puisse être créé par tous les `Player` (même si quelqu'un récupère votre code, il ne devrait pas pouvoir tricher pour créer plusieurs `Server`), et qu'au final, le `Main1` affiche bien trois `Character` de couleurs différentes.

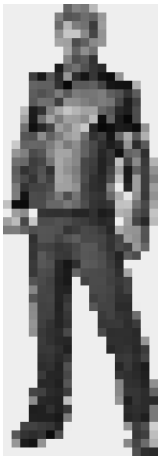
**Contraintes d'implémentation** : Vous ne pouvez modifier que les classes `Player` (une seule ligne de code pourra être modifiée dans ce fichier) et `Server`. Dans cette dernière, vous ne devez pas rendre la variable de classe `numberOfPlayers` statique, ni modifier le contenu de la fonction `getNewCharacter` (par contre vous pourrez modifier son prototype). Si vous pensez avoir tout modifié comme il faut, mais que vous avez toujours trois personnages de la même couleur qui apparaissent, regardez justement si vous ne devez pas faire quelque chose dans la déclaration du prototype de cette dernière fonction.

## 3 Second exercice

Utilisez maintenant le `main()` de la classe `Main2`. Lorsque l'on souhaite créer un GraphicMultiPlatform (à partir d'un `Character`) et l'afficher, beaucoup de fonctions doivent être appelées selon que l'on souhaite créer un `GraphicMultiPlatform` pour l'afficher sur PC, sur Iphone, ou sur GameBoy. Si l'on n'appelle pas les bonnes fonctions, ou que l'on appelle trop de fonctions, une erreur d'affiche à l'écran. Testez le `Main2` directement : une erreur devrait s'afficher.

Dans le Main, vous verrez, en commentaires, les différentes instructions à exécuter pour afficher un `GraphicMultiPlatform` en version PC, en version Iphone, ou bien en version Gameboy. Décommentez successivement les trois blocs de code afin d'afficher un `GraphicMultiPlatform` pour le PC, pour l'Iphone ou pour la GameBoy.





On souhaiterait obtenir un système assez simple pour créer des `GraphicMultiPlatform` qui peuvent être directement affichés, sans qu'il soit nécessaire d'appeler toutes ces fonctions dans le `Main`.

**Contraintes d'implémentation :** Si l'on modifie (ajouter ou retirer des fonctions) la façon de créer des `GraphicMultiPlatform` pour le PC, l'Iphone ou la Gameboy, votre code doit facilement s'adapter. Si par exemple, on décide que pour créer un `GraphicMultiPlatform` pour Iphone, il ne faut pas appeler la fonction `load3DEnvironments()`, un minimum de code devra être modifié.

De plus, il faudra avoir, dans le `Main2` ou dans une autre classe, un code générique pour la création des `GraphicMultiPlatform` pour PC, des Iphone, ou des Gameboy (le même code pour les trois). Si, par exemple, la façon de créer des `GraphicMultiPlatform` pour IPhone est changée en retirant la fonction `load3DEnvironments()`, votre code générique ne devra pas changer.

**Indice :** vous aurez au moins besoin de créer trois classes spécialisées.

# 4 Troisième exercice

Les classes `Knife`, `Mask`, `Sabre` et `Club` permettent d'ajouter des accessoires à votre `Character`. Dans le `Main3`, vous trouverez une façon d'afficher un accessoire sur votre personnage. On souhaiterait une solution élégante afin de superposer les accessoires sur un `Character`.

**Contrainte :** Pour afficher un `Character` à l'écran (avec ou sans accessoires), il faudra passer à travers un `GraphicMultiPlatform` (même code que pour l'exercice précédent). L'affichage d'un personnage avec accessoires devra se faire à travers un `GraphicMultiPlatform`. Aucun appel à la fonction `display()` de `MyImage` ne devra être effectué en dehors de `GraphicMultiPlatform` (les effacer des fonctions `paintOnCharacter()` des accessoires).

**Indice :** Pour créer un personnage avec un sabre et un masque, on fera :

```
Character character = new Sabre(new Mask(new ChuckGreene(Colour.YELLOW)));
```

# 5 Quatrième exercice

On souhaite ajouter un menu permettant de créer différents personnages à la volée. Ce menu sera représenté par une machine à états.

Dans le premier état, on demande à l'utilisateur de commencer une création de personnage ou de quitter (fin du programme).

Si l'utilisateur choisit la création, on passe au second état qui consiste à demander la couleur du personnage (quatre couleurs disponibles, regardez la classe `Color`) ou bien de revenir au premier état.

Une fois la couleur choisie, on passe au troisième état : on peut ajouter un accessoire, revenir au premier état, ou bien afficher le personnage. Si on décide d'afficher le personnage, on l'affiche et on revient au premier état. Si on décide d'ajouter un accessoire, on reste dans le troisième état afin de pouvoir ajouter plus d'accessoires.

Dans chacune des étapes, il sera proposé à l'utilisateur un menu, et chacun des choix sera entré par un numéro (par exemple, dans l'état numéro deux, le choix 1 serait la couleur rouge, le 2 serait le vert, le trois serait le bleu, le quatre serait le jaune, et le cinq serait revenir au menu principal).

**Indice :** Une classe abstraite `Etat` pourrait être créée, avec les fonctions `choix1`, `choix2`, `choix3`… Si on appelle `choix3` alors qu'on est dans le premier état, rien ne se passe.

Les différents `Etat` pourront s'occuper d'afficher des informations à l'utilisateur et de rediriger vers l'état suivant.

# 6 Cinquième exercice

Ce dernier exercice consiste à rajouter des fonctionnalités dans votre code et à mesurer la facilité avec laquelle ces ajouts peuvent être faits.

Dans le dossier `img`, vous trouverez un personnage appelé `Crystal` (en rouge, vert, bleu et jaune). Intégrez ce personnage à votre code afin que l'on puisse produire des `ChuckGreene` ou des `Crystal` (un pattern bien connu peut être utile). Une partie de la classe `Crystal` est donnée ci-dessous :

```
public class Crystal implements Character {
    private int[] _rightHand = {94, 260};
    private int[] _leftHand = {185, 260};
    private int[] _head = {137, 70};
    private int[] _torso = {140, 183};
}
```

Le code suivant permet d'afficher une guitare dans le dos d'un personnage. Ajoutez la guitare aux accessoires que l'on peut rajouter à un personnage (attention, la guitare apparaît derrière le personnage).

```
public class Guitar {
    private final MyImage _image;
    private final int _xShift;
    private final int _yShift;

    public Guitar() {
```

```
        _image = new MyImage("img/guitare.png");
        _xShift = -110;
        _yShift = -110;
    }

    public MyImage paintOnCharacter( Character c ) {
        MyImage image = c.getImage();
        image.paintUnder(_image, c.getTorso()[0] + _xShift,
                                 c.getTorso()[1] + _yShift);
        return image;
    }}
```

Intégrez Crystal et la guitare à votre menu (il faudra créer un nouvel état pour sélectionner le personnage à construire).