Astéroïde 325

***Objectifs** : expérimenter les notions de délégation, tableaux, classe abstraite, énumération, Vector, méthode statique. Travailler sur un projet de plusieurs classes et paquetages.*

Ce TP vous emmène en voyage sur l'astéroïde 325. Ce petit bout de cailloux fait partie d'un des décors de l'histoire du *Petit Prince* d'Antoine de Saint Exupéry. Il possède un seul habitant : un roi pas très marrant :

> *- Ah ! Voilà un sujet, s'écria le roi quand il aperçut le Petit Prince.*
> *Et le Petit Prince se demanda :*
> *- Comment peut-il me connaître puisqu'il ne m'a encore jamais vu !*
> *Il ne savait pas que, pour les rois, le monde est très simplifié. Tous les hommes sont des sujets.*

Nous vous proposons pour quelques heures de changer un peu l'histoire, en particulier de peupler cet astéroïde de petits princes virtuels, et de les faire interragir entre eux et avec le roi par l'intermédiaire d'un moyen de communication moderne : la console JavaBoy™ !

➨ Importez dans Eclipse l'archive ~`leriche/TPJAVA/tp3.tar.gz`.

1 Les Petits Princes, le Roi et la console JavaBoy™

A. de Saint Exupéry l'a bien compris (sans avoir fait le module 6...), on peut simplifier ce monde et considérer que toute personne marchant sur l'astéroïde 325 est un sujet. Cela permet au roi (qui est en fait un programme serveur) d'interragir avec un sujet quelle que soit son implémentation (vous aurez tous des Petits Princes différents), tant qu'il possède les méthodes spécifiées dans l'interface `Sujet`.

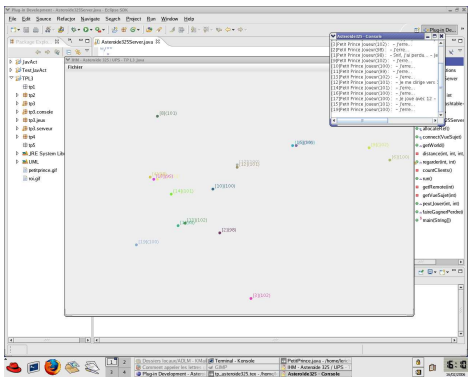
Chaque sujet arrivant sur l'astéroïde 325, peut utiliser les services de la console JavaBoy™ prêtée par le roi (disponible dans le paquetage `console`) et dont les services sont récapitulés en annexe (dernière page).

1.1 Atterrissage du Petit prince

➨ Codez votre Petit Prince qui implémente l'interface `Sujet`. Dans la méthode `run()`, vous pourrez dans un premier temps faire s'exprimer votre sujet en utilisant la méthode `parler()` de la console, puis le faire se déplacer aléatoirement en utilisant la méthode `seDirigerVers(0)`.

➨ Ajoutez une méthode `main`, qui se contente de construire une instance de votre personnage.

1.2 Branchement de la console JavaBoy™



➨ Lancez une exécution de votre sujet. Vous pouvez observer sur la console un message vous donnant un identifiant de console ; ce numéro vous permet de repérer votre sujet par la suite. Pour visualiser l'astéroïde 325 et tous ses sujets, vous avez à votre disposition une classe `IHM`. Lancez-la et observez les activités de votre sujet.

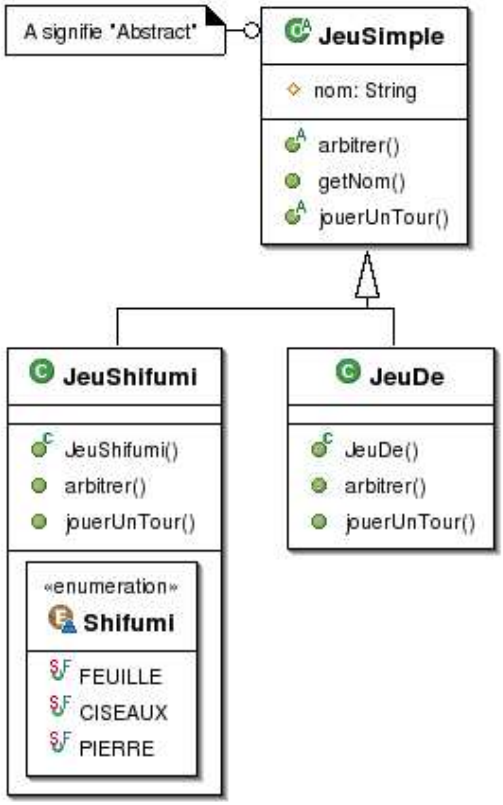
Remarque : en cas de défaillance de votre programme (temps de réponse trop long, inactivité, etc.) et dans certains cas, le roi vous renverra de son astéroïde, en vous donnant la raison. Vous pouvez également changer de planète et partir de l'astéroïde en terminant l'exécution de votre sujet.

2 Le démon du jeu

Votre Petit Prince est maintenant capable de se déplacer sur l'astéroïde 325, mais comme marcher n'est pas son fort, il se dit qu'il programmerait bien sa console JavaBoy™ pour jouer avec les autres sujets du royaume.

2.1 Des jeux simples

Le Petit Prince imagine des jeux simples comme "pile ou face", des "jeux de dé", ou encore jouer avec les mains par exemple à "Pierre-Feuille-Ciseau". En y réfléchissant, il se dit que tous ces jeux ont des points communs et pour factoriser les codes, il dessine vite fait dans le sable la classe abstraite `JeuSimple` ci-contre.



Il se dit ensuite qu'il pourra hériter de cette classe pour écrire les petits jeux auquels il a pensé.

➨ Créez un paquetage `jeu` dans lequel vous allez placer les classes correspondant à cette partie.

➨ Codez la classe abstraite `JeuSimple`. Les signatures des méthodes sont les suivantes (attention, certaines sont abstraites) :
`String jouerUnTour()`
`int arbitrer(String c1, String c2)`
`String getNom()`
La classe contient un champ nom (`String`)

➨ Codez la classe `JeuDeDe`. Pour jouer un tour, vous retournerez une chaîne de caractères contenant un chiffre tiré au hasard entre 1 et 6 (utilisez la classe `java.util.Random`). Pour la méthode `arbitrer`, le gagnant peut être (à votre convenance) le plus grand ou le plus petit chiffre passé en paramètre. Ainsi vous retournerez -1 si `c1` est le gagnant, +1 si c'est `c2`, et 0 en cas d'égalité.

2.1.1 Shifumi

Pour "Pierre-Feuille-Ciseau" (qui s'appelle aussi le jeu de Shifumi), le Petit Prince pense tirer partie des nouveautés de Java 1.5 (en l'occurrence les types énumérés) pour représenter les 3 coups possibles. Une énumération est en Java un objet particulier (voir la javadoc pour `java.lang.Enum`), construit avec le mot-clé `enum`, de la manière suivante :

```
enum Saison{PRINTEMPS, ETE, AUTOMNE, HIVER};
```

Ces objets n'ont pas besoin d'être construits (ils n'ont pas vraiment de constructeurs). On dit qu'ils sont *statiques*, ce qui signifie qu'ils sont partagés par tous les objets et directement visibles. Pour y accéder, on utilise le nom de la classe à la place d'un identificateur d'objet. Les énumérations possèdent plusieurs méthodes prédéfinies. Par exemple : `Saison.values()` qui renvoie un tableau contenant les valeur des éléments ou `toString()`. Les éléments sont accessibles par des champs statiques `Saison.ETE`. Enfin, les types énumérés sont ordonnés et comparables (selon l'interface `Comparable`). On a par exemple : `Saison.ETE < Saison.HIVER`.

➨ Ecrivez la classe `JeuShifumi`, qui contient et exploite une énumération correspondant au diagramme de classe donné plus haut. Pour jouer un tour, vous retournerez une chaîne de caractères correspondant à l'un des coups possible du Sifumi, tiré aléatoirement.

2.1.2 Arbitrage

➨ Ecrivez une interface `Joueur` qui contient les méthodes `void gagner(int argent)` et `void perdre(int argent)`.

➨ Ecrivez une classe de test, qui contient une méthode statique :
`static void Arbitrer(Joueur j1, Joueur j2, JeuSimple jeu)`. Cette méthode utilise l'objet `jeu` pour faire jouer un tour à chaque joueur, et appelle la méthode d'arbitrage de ce jeu sur les deux résultats. En fonction du résultat (utilisez l'instruction `switch`), vous appellerez les méthodes `gagner` et `perdre` des joueurs j1 et j2. Testez vos jeux.

2.2 Jouons ensemble !

Maintenant que les jeux sont en place, vous allez pouvoir les intégrer dans le monde de l'Astéroïde 325. Pour cela, vous devrez d'abord essayer de vous rapprocher d'autres sujets. Ensuite, lorsqu'ils seront distants d'une seule position, vous pourrez essayer de jouer avec eux (via la méthode `jouer(JeuSimple jeu, int ref)`). Le roi se fera un plaisir d'arbitrer la partie, et de déclarer éventuellement le gagnant en appelant les méthodes `gagner` et `perdre` de chacun.

➨ Reprenez le code de votre Petit Prince, et exprimez que celui-ci est un joueur (vous devrez ensuite implémenter les deux méthodes `gagner` et `perdre`).

➨ Les méthodes permettant de faire jouer les sujets ont été mises en commentaires pour vous permettre de faire la première partie du TP sans problèmes. Pour poursuivre, vous devrez décommenter dans la classe `tp3.console.QuasiConsole` :
- les "import" autour de la ligne 15 à 16 ;
- la méthode "jouer" ligne 217 à 246 ;
- la méthode "faireGagnerPerdre" ligne 262 à 265 ;
- la ligne 252, en supprimant la ligne suivante (enlevez l'instruction `return false`).

De même, dans la classe `tp3.console.ConsoleJavaBoy` décommentez :
- les "import" autour de la ligne 4 à 5 ;
- la méthode "jouer" de la ligne 66 à 68.

➨ Pour mesurer la distance qui vous sépare d'un autre sujet, vous pouvez ajouter des méthodes privées. Vous pouvez utiliser la distance de Chebyshev (nombre de cases séparant deux positions) : $d = max(\Delta X, \Delta Y)$.

➨ Vous pouvez utiliser ce genre d'algorithme pour coder le comportement de votre sujet :

```
V = regarder
S = chercher le plus proche sujet dans V

SI (distance(s)<=1) ALORS
  jouer avec S a un jeu (au hasard parmi ceux implantes)
SINON
  SI (s est null) ALORS se diriger au hasard
  SINON se diriger vers S
  FINSI
FINSI
```

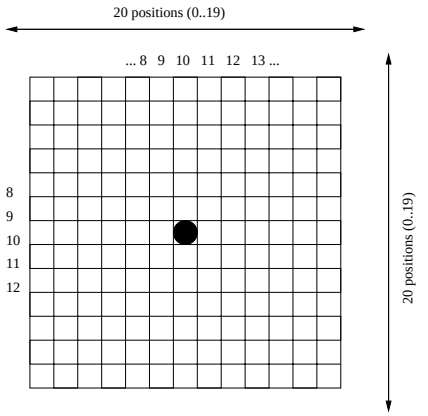
➨ Vous ne pouvez jouer qu'avec des sujets qui n'ont jamais joué avec vous, et qui sont également des joueurs. Pour cela, le Petit Prince vous souffle d'utiliser une `Collection` (par exemple un objet `Vector`) qui contiendrait l'ensemble des identifiants de sujets avec lesquels vous ne pouvez plus jouer. Modifiez le code de votre méthode `run()` pour poursuivre des sujets et leur proposer l'un de vos jeux quand vous serez assez proche.

Annexe

Services de la console JavaBoy™

```
/* Constructeur, prend la reference de votre sujet en parametre */
public ConsoleJavaBoy(Sujet s)

/**
 * Permetà un sujet de percevoir son environnement
 * @return un tableau d'entiers, nuls si il n'y a aucun sujet en cette position, une
 * reference de sujet sinon
 */
public int[][] regarder()

/**
 * éDplace ce sujet d'une case en direction du sujet dont l'identifiant est donnee
 * @param ref l'identifiant d'un sujet, 0=>le sujet va errer aleatoirement
 */
public void seDirigerVers(int ref)

/**
 * Envoie un message qui sera visible sur l'IHM
 * @param s le message
 */
public void parler(String s)

/**
 * Pour jouer au jeu passe en parametre avec un autre sujet dont on precise la reference,
 * @param jeu le jeu auquel on va jouer
 * @param ref l'identifiant du joueur éoppos
 */
public void jouer(JeuSimple jeu, int ref)

 /**
 * Permet d'afficher ou non les messages envoyes via la console sur le terminal
 * Valeur a faux par defaut, il peut etre utile (pour tester) de la mettre a vrai
 * @param e vrai si on fait l'echo sur le terminal
 */public void setEcho(boolean e)
```

Vision du Petit Prince



La vision d'un Petit Prince est donnée par la console JavaBoy™ . Celle-ci retourne un tableau d'entiers (20x20) contenant des numéros identifiants les sujets. Deux sujets ne peuvent jamais occuper une même case.