MANZANO Lilian
LAFAURE Robin

ie2-021

RAPPORT FINAL PROJET

Sommaire

Introduction

Nos objectifs

Lors de ce projet, nous devons réfléchir au développement d'un programme permettant de jouer à un jeu de société se basant sur le jeu ''Kartel'', et réaliser le plus possible ce programme.

Kartel est un jeu de société où tous les joueurs incarnent le même détective dont l'objectif est de mettre en prison les membres des différents gangs et parfois (pas trop souvent) d'accepter les pots-de-vin. Mais attention, il faut choisir judicieusement ses alliés (car un Boss en prison refusera de payer ses pots-de-vin) et ses ennemis (en effet, s'ils parviennent à s'enfuir, les Boss sauront venger leurs gangsters incarcérés) .

Notre travail

Pour réaliser ce projet, nous sommes restés en duo, même si la situation s'y prête moins, duo formé de MANZANO Lilian et LAFAURE Robin.

Il est important de préciser que nous avons réalisé tout le programme. Il est ainsi fonctionnel et il est possible de jouer à notre jeu avec toutes les fonctionnalités que nous allons décrire par la suite.

Notre code

En ce qui concerne le code de notre programme, il y a quelques éléments à savoir que nous n'allons pas forcément préciser plus tard:

- Tout d'abord, nous avons importé la classe ''Lire'' à notre programme, classe qui vient d'internet. Nous ne l'avons donc pas programmée nous-même. Cette classe permet de récupérer les données entrées par l'utilisateur.
- Ensuite, il faut savoir que nous avons programmé pour laisser la possibilité de facilement ajouter de nouvelles fonctionnalités, c'est pour cela que certaines parties de notre code comme par exemple certains accesseurs ne serviront pas. Nous nous permettons de faire cela puisque ce n'est pas un programme très lourd et qu'il n'est pas précisé qu'il doit être optimisé.
- Enfin, tous les menus de notre jeu sont sécurisés, c'est-à-dire que lorsqu'il sera demandé de saisir un nombre pour faire un choix, si l'utilisateur entre des caractères non proposés le programme n'aura pas d'erreur, il se contentera de réafficher le menu en indiquant ce qu'il n'allait pas dans le choix précédent.

Le jeu et ses fonctionnalités

Dans cette partie nous allons retracer le parcours de joueurs lorsqu'ils lancent le programme, cela va nous permettre de voir toutes les fonctionnalités intégrées dans le jeu.

Avant la partie :

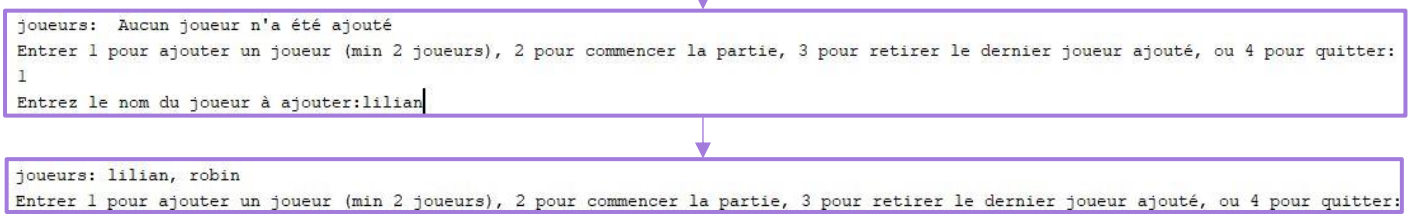
Lorsque l'utilisateur lance le programme, une phrase d'accueil s'affiche ainsi que le premier menu. Celui-ci permet de lancer une partie, d'afficher les règles, ou de juste quitter :

```
bienvenu pour jouer à Kartel
_____
Entrer 1 pour lancer une partie, 2 pour voir les règles, 3 pour quitter:
```

Dans le cas où le joueur veut quitter, le programme se ferme directement. S'il veut voir les règles, celles-ci sont affichées, ainsi que la façon de calculer les points et la correspondance des jetons aux caractères utilisés dans ce jeu (ex : <>=détective), puis le premier menu est affiché de nouveau. Et enfin si l'utilisateur veut lancer la partie un nouveau menu s'affiche.

Le nouveau menu est précédé par la liste des joueurs déjà ajoutés. Il donne la possibilité d'ajouter un joueur, de commencer la partie, de retirer le dernier joueur ajouté en cas d'erreur, ou encore une fois de quitter . Pour la troisième et quatrième option, l'action se fait directement. En revanche pour ajouter un joueur il est demandé son pseudo, une fois celui-ci rentré, il est ajouté à la liste de ceux ajoutés et le menu est affiché de nouveau :

```
joueurs:  Aucun joueur n'a été ajouté
Entrer 1 pour ajouter un joueur (min 2 joueurs), 2 pour commencer la partie, 3 pour retirer le dernier joueur ajouté, ou 4 pour quitter:
```




```
joueurs: lilian, robin
Entrer 1 pour ajouter un joueur (min 2 joueurs), 2 pour commencer la partie, 3 pour retirer le dernier joueur ajouté, ou 4 pour quitter:
```

Pour pouvoir commencer la partie il faut au minimum deux joueurs. Une fois ceux-ci ajoutés, la deuxième option du menu peut être utilisée, lorsqu'elle l'est, le jeu commence et le premier tour s'effectue directement.

Pendant la partie :

À chaque tour de jeu, l'affichage est construit de la même manière, la première ligne correspond au plateau et la seconde à la prison où sont mis les boss lorsqu'ils sont récupérés. Les lignes suivantes ont les pseudos de chaque joueur suivis entre parenthèses de leur score actuelle en fonction de leur réserve et de la prison. Il y a également sur la même ligne leur réserve avec les pots-de-vin ou les gangsters qu'ils ont récupérés ; il y a autant de lignes comme celles-ci qu'il y a de joueurs, soit une par joueur.

Ensuite, il y a indiqué le nom du joueur qui doit jouer, suivi en dessous du résultat du dé, dé qui a été ''lancé'' automatiquement. Et enfin un menu est affiché, demandant au joueur de combien de cases il souhaite avancer, les possibilités dépendent bien-sûr du résultat du dé :

```
2G - 1G - 2B - 2C - [A] - $E - 2B - $F - $G - $B - 2E - 3D - <> - 1E - [B] - 2D - ∞ - [D] - 2A - 3B - $D -
Prison: [E] - [F] - [G]
lilian (6) : 2G - $C - 1A - 2E
robin (1) : 1C - 2C - 1F - 3F

C'est à robin de jouer.
Le dé a été lancé, le résultat est :3
Tu peux donc choisir entre avancé de 2 cases, 3 cases, selon ton choix rentre le chiffre correspondant:
```

Lorsque le joueur a choisi un chiffre, le tour suivant s'affiche directement avec les modifications (détective déplacé, jeton récupéré placé au bon endroit, scores recalculés, indication du nom suivant qui doit jouer, dé relancé et menu correspondant au résultat de celui-ci).

Il y a un seul cas où le tour suivant n'est pas lancé directement, c'est lorsque le joueur récupère le jeton ''évasion''. C'est une nouvelle fonctionnalité que nous avons décidé d'ajouter, ce jeton est représenté par le signe hash (#)* dans notre jeu, et permet si le joueur le souhaite de faire sortir un boss de prison et le faire disparaître dans la nature.

Lorsque ce jeton est récupéré, un menu est affiché seulement s'il y a au moins un boss en prison (sinon le tour suivant s'affiche), ce menu permet au joueur de décider s'il veut faire sortir un boss, et si oui lequel.

* Anciennement nous notions l'évasion avec le signe infini (∞), c'est ce signe qui est présent sur les captures d'écran de ce rapport.

```
C'est à lilian de jouer.
Le dé a été lancé, le résultat est :2
Tu peux donc choisir entre avancé de 2 cases, selon ton choix rentre le chiffre correspondant:
2

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
Vous avez récupéré le jeton d'évasion, il permet si vous le souhaiter de faire s'évader un Boss de prison,
pour ça entrer 1 pour libérer le boss de la cellule 1, 2 pour libérer le boss de la cellule 2, 3 pour libérer le boss de la cellule 3, 4 pour libérer le boss de la cellule 4,
si vous ne voulez pas libérer de boss il faut entrer 5
```

Une fois le choix fait, le cours de la partie reprend normalement en passant au tour suivant, et le jeton ‘‘évasion’’ disparait et devient inutilisable qu’il ait ou non été utilisé.

Après la partie :

La partie se finit lorsque la prison est pleine, c’est-à-dire quand il y a 5 boss emprisonnés. Dès lors, une phrase indiquant ceci est affichée ; les joueurs avec leur score et leur réserve sont eux aussi indiqués comme pendant le reste de la partie ; le ou les gagnants (en cas d’égalité) sont nommés ; et un nouveau menu nous proposant de rejouer est affiché :

```
La partie est terminé car la prison est pleine, voici les boss qui sont en prison:
Prison: [E] - [G] - [B] - [D] - [C]

Et voici les joueurs avec leurs score et leurs réserves
lilian (26) : 2G - $C - 1A - 2E - $D - $A - 2F - 1D - 2C - $F - 3D - 3B - 3G - 2B - 2E - 3E - $G
robin (5) : 1C - 2C - 1F - 3F - 2A - 1B - 3C - 2A - 1G - $E - $B - 2D - 2D - 3A - 2B - 2G

Ainsi le(s) gagnant(s) de cette partie est/sont: lilian
Vous avez la possibilité de rejouer

_____________________________________
Entrer 1 pour rejouer, 2 pour voir les règles, 3 pour quitter:
2
```

Le menu pourrait être le même que le premier que nous voyons en lançant le programme, car celui-ci possède les mêmes possibilités, c’est-à-dire jouer, afficher les règles, ou quitter. Cependant pour faire plus proche de l’utilisateur nous utilisons ce second menu qui remplace juste ‘‘jouer’’ par ‘‘rejouer’’.

Par contre ici, si le joueur souhaite rejouer, un nouveau menu apparaît, donnant la possibilité de rejouer avec les mêmes joueurs :

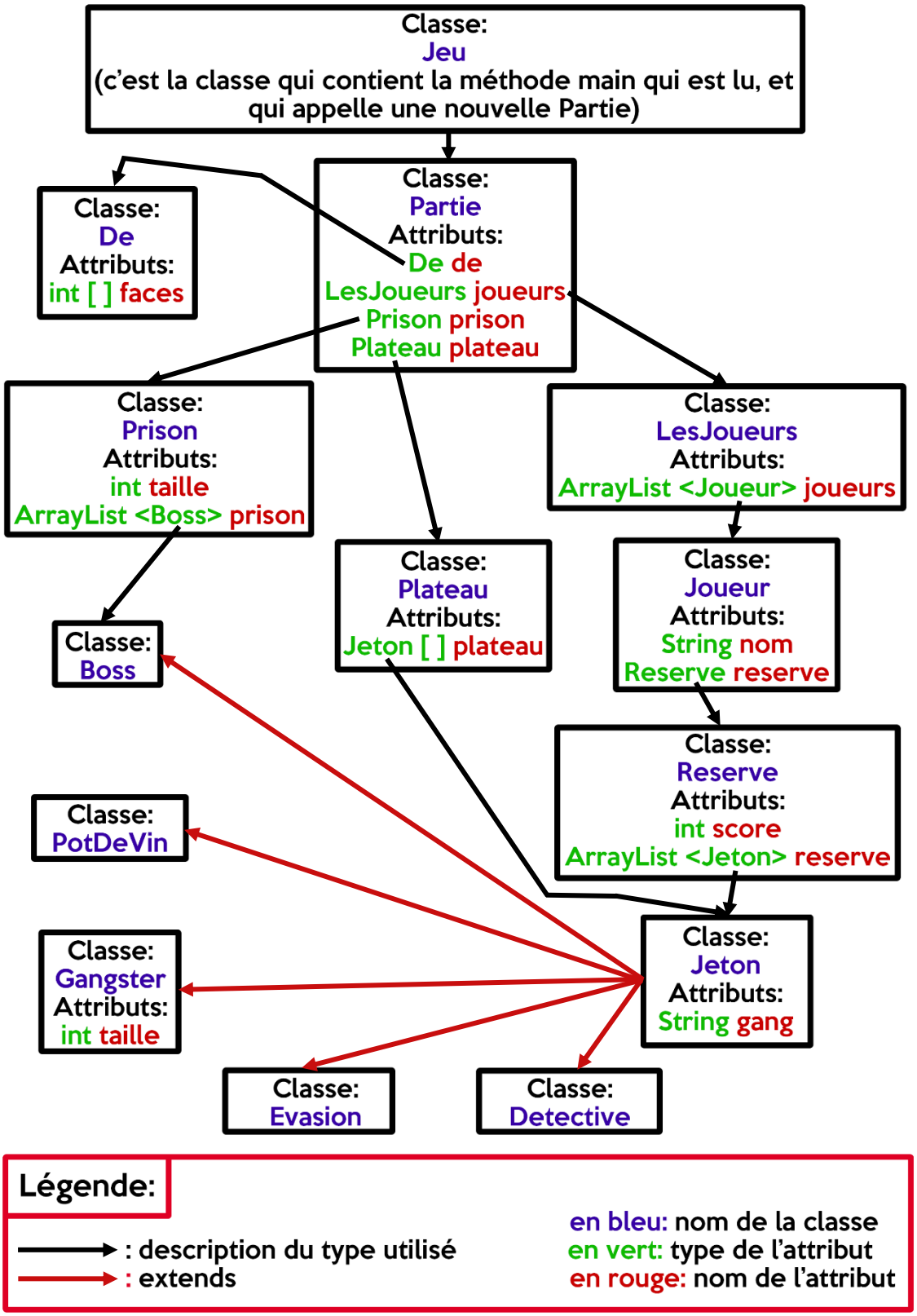
```
Entrer 1 pour rejouer, 2 pour voir les règles, 3 pour quitter:
1

Entrer 1 pour rejouer avec les mêmes joueurs, 2 pour changer de joueurs:
```

Si l’utilisateur décide de rejouer avec les mêmes joueurs, la partie se lance, et si en revanche il veut de nouveaux joueurs il se retrouve devant le menu permettant d’en ajouter puis de lancer la partie (menu vu précédemment).

Tant que l’utilisateur ne décide pas de quitter le jeu, il a la possibilité de refaire des parties de la même façon.

Diagramme de classes



Pour mieux comprendre et surtout pour donner plus de précision, nous allons reprendre classe par classe afin d'étudier leur fonction et les liens entre elles.

Description du programme

Dans la suite nous allons parler du code, cependant nous n'allons pas forcément nous attarder sur tous les détails de la «cuisine Java», c'est-à-dire que nous n'allons pas mentionner tout ce qui est évident pour un programme java (comme par exemple la présence des accesseurs classiques) si ceci n'est pas important, et pourtant ceux-ci sont présents dans le programme.

classe «Jeu»

La classe «Jeu» représente l'application. C'est cette classe qui possède la méthode ''main'' qui est exécutée, donc ici ce qui est programmé est en fait le menu principal, et c'est depuis ce menu que l'on peut choisir de créer une nouvelle instance de la classe «Partie».

classe «Partie»

Une partie est représentée par la classe «Partie». Pour qu'une partie soit possible, elle doit posséder plusieurs éléments, qui sont ici représentés grâce à ses attributs. La partie a besoin de la prison, du plateau, du dé, et enfin de la liste des joueurs qui elle, est évolutive selon les joueurs que l'utilisateur ajoute.

La classe partie possède 2 constructeurs : le premier sans paramètre, où une fonction permettant d'afficher le menu d'ajout de joueur sera appelée ; et le second qui possède en paramètre une liste de joueur, cette liste sera donc appliquée directement pour créer la partie, ce second constructeur permet donc de proposer à l'utilisateur de rejouer avec les mêmes joueurs que la partie précédente.

De plus, la classe «Partie» a deux fonctions, celle qui est appelée ponctuellement et qui permet d'ajouter des joueurs, et la seconde qui est appelée à chaque partie, et qui permet de jouer.

classe «LesJoueurs»

Voyons maintenant comment fonctionne une liste de joueur. Celle-ci gère juste une ArrayList contenant des types «Joueur». En revanche elle possède de nombreuses méthodes permettant de gérer cette ArrayList. Tout d'abord il y a les méthodes qui permettent d'ajouter ou retirer des joueurs, ensuite il y a des méthodes qui permettent d'afficher ces joueurs de différentes façons, et enfin il y a les méthodes les plus intéressantes.

Il y a la méthode ''initialiser()'' qui est appelée uniquement lorsque le joueur décide de refaire une partie avec les mêmes joueurs. Cette méthode permet d'initialiser les joueurs de la liste à zéro, par là on parle de mettre leur score à zéro et vider leur réserve.

Et pour finir avec cette classe, il y a la méthode ''calculScores()'' qui permet de calculer le score de chaque joueur de cette liste. Pour ceci elle a un paramètre de type «Prison» qui lui permettra de savoir les boss qu'il y a en prison. Cette méthode sera utilisée à chaque tour de jeu depuis la fonction qui représente le jeu de la classe «Partie».

classe «Joueur»

Nous avons vu comment est gérée une liste de joueurs, voyons maintenant comment est géré le joueur en lui-même. Celui-ci a sa propre classe, cependant il n'y a pas grand-chose à l'intérieur. Le joueur a deux attributs, son nom et sa réserve de jetons que nous allons voir juste après. De plus, le joueur a une seule méthode qui permet juste de récupérer sous forme de caractère lui, son score, et sa réserve (ce qui est affiché lors d'une partie pour chaque joueur).

classe «Reserve»

Étudions la réserve, celle-ci à deux éléments : une liste de jetons qui permet de les stocker lorsqu'un joueur en récupère, et le score de cette réserve, c'est-à-dire le score des jetons en fonction des boss présents dans la prison.

Il y a 3 méthodes dans cette classe, la première permet juste d'ajouter un jeton à la réserve ; la seconde permet d'afficher la réserve, c'est-à-dire afficher les jetons séparés par des tirets ; et la dernière permet d'initialiser cette réserve (mettre le score à zéro et effacer les jetons qu'elle possède), celle-ci est utilisée dans la méthode ''initialiser()'' de la classe «LesJoueurs».

classe «Prison»

La façon dont sont gérés les jetons est un peu plus particulière nous y reviendrons donc plus tard. Pour le moment, revenons aux attributs de la classe «Partie» que nous n'avons pas traités, ainsi parlons de la prison.

La prison est caractérisée par sa taille et la liste de boss qu'elle contient. Sa taille est pour le moment toujours définie sur 5, ce qui signifie que dès qu'elle contient 5 boss elle devient pleine et donc la partie est finie (l'attribut taille existe même si c'est par défaut 5, ceci car on n'exclut pas la possibilité de rajouter une fonctionnalité permettant de changer celle-ci).

La prison possède des méthodes pour rajouter ou retirer un boss, mais aussi des méthodes permettant de savoir son état. Ainsi, il y a une méthode permettant de savoir si celle-ci est pleine, de même pour savoir si la prison est vide, et même une méthode pour vérifier qu'un boss est en prison ou non (permet de calculer les scores). Et enfin bien entendu, il y a la méthode qui permet d'afficher la prison sous forme de caractère (similaire à la méthode de la réserve).

---

classe «De»

Nous reparlerons des boss plus tard, en même temps que nous évoquerons les jetons. Pour le moment revenons sur un élément indispensable du jeu, le dé.

Le dé possède un nombre de face défini, ici prédéfini sur 6 (encore une fois il y a quand même un attribut pour les raisons déjà citées) et avec 2 faces de 2, 2 faces de 3, et 2 faces de 4; il possède également un attribut face courante qui correspondra à la face du dé après qu'il aura été jeté.

Le dé possède 2 méthodes, la première qui est uniquement utilisée dans son constructeur, elle permet d'initialiser les faces du dé ; et la seconde qui sera appelée à chaque tour de jeu, elle permet de ''jeter le dé'', en fait elle se contente de prendre un entier au hasard entre 1 et 6 compris.

---

classe «Plateau»

Il nous reste un élément indispensable à une partie de jeu qu'on va appeler le plateau. C'est en fait une liste de jeton dont l'ordre définit le placement de ceux-ci. Cette classe possède une fonction qui permet d'initialiser le plateau avec les 44 jetons (1 détective, 1 évasion, 7 boss, 7 pots-de-vin, 28 gangsters) dans un ordre aléatoire.

De plus, il y a différentes méthodes dans cette classe il y a les méthodes qui permettent de déplacer le détective, la première permet de récupérer la place du détective, ainsi cette méthode est utilisée dans la seconde qui permet de le faire avancer. Cette seconde méthode possède un entier en paramètre qui sera en fait le nombre de jetons que souhaite avancer le joueur, ainsi cette méthode récupère la place du détective et rajoute l'entier récupérer à celle-ci, le détective se retrouvera donc à la place d'un autre jeton, il faut donc retourner ce jeton pour pouvoir l'enlever du plateau et le placer en prison ou dans la réserve du joueur.

Ensuite, il y a une méthode qui permet de diminuer le plateau en enlevant les places vides. Ceci permet 2 choses, l'affichage propre du plateau, et éviter qu'en faisant un tour le détective tombe sur ces places.

Enfin il y a la méthode qui permet d'afficher le plateau (similaire à celles des classes «Reserve» et «Prison»), méthode qui est donc utilisée à chaque tour de jeu.

---

classe «Jeton»

Maintenant nous allons étudier la façon de programmer les jetons, nous les avions laissés de côté plus tôt et nous avions dit que ceux-ci sont un peu particuliers, mais pour quelles raisons ?

Un type «Jeton» est créé grâce à une classe du même nom, ce qui permet de mettre les jetons dans le plateau où encore la réserve d'un joueur. C'est donc pratique d'utiliser une même classe s'il y a une utilisation commune entre tous les jetons. Cependant il y a aussi besoin de différencier les types de jetons, car il y a besoin de différents éléments pour chaque type, par exemple toute les méthodes ''toString()'' des jetons seront différentes car il faut pouvoir différencier le détective d'un gangster lorsque l'on joue, autre exemple, la méthode permettant d'afficher un menu qui est utilisée uniquement pour le jeton «Evasion», ou encore juste la caractéristique de taille qui est uniquement utile pour les jetons «Gangster».

Ainsi pour résoudre ceci, nous avons créé une classe «Jeton» représentant tous les jetons, et d'autres classes ''extends'' pour apporter des précisions pour chaque type de jeton. Utiliser des classes ''extends'' permet d'avoir des types avec de nouveaux éléments propres à eux tout en étant référencés par un type commun qui est le type «Jeton».

classe «PotDeVin»

Le jeton pot-de-vin n'a pas de caractéristiques particulières, c'est pour cela que sa classe utilise uniquement l'attribut de sa classe parent (la classe «Jeton»), attribut qui définit le gang. Cette classe existe uniquement pour sa méthode ''toString()'' qui est essentielle.

classe «Boss»
classe «Detective»

La classe «PotDeVin» n'est pas la seule à être comme ceci puisque c'est aussi le cas des classes représentant les boss, et le détective.

classe «Gangster»

La classe représentant les jetons de type «Gangster» est assez ressemblante aux précédentes, cependant elle possède un attribut ''taille'', qui permet de définir le nombre de gangster qu'il y a sur un jeton (ils peuvent être 1, 2, ou 3).

classe «Evasion»

Pour finir, voyons la classe représentant le jeton «Evasion», celle qui est la plus représentative de l'intérêt d'utiliser l' ''extends''. Cette classe a une méthode en plus des autres, c'est la méthode ''evenement()'' qui représente tout ce qui se passe à partir du moment où le jeton ait été récupéré, c'est-à-dire l'affichage du message proposant un choix, la récupération du choix, et le retrait du boss choisi de la prison.