DevineLaCarte

Table des matières

Thème

Développer une logique de jeu mettant en oeuvre de la conception objet et des tests unitaires.

Jeu en mode console. Un début d'implémentation est proposé (`MainPlayConsole.kt` en interaction en mode console/terminal)

Prérequis

Niveau : Deuxième année de BTS SIO SLAM

- Bases de la programmation,
- IntelliJ et kotlin opérationnels sur votre machine de dev
- Avoir fait des premiers pas avec Kotlin (exercices d'initiation)
- Avoir eu une première introduction à la notion de Test Unitaire, distinction entre ***expected value*** et ***actual value***, dans une approche prédictive. En particulier avoir réalisé avec succès l'exercice Faites vos comptes

Objectif

- Conception et mise au point d'une logique applicative avec Kotlin et JUnit
- Structure de données, recherche d'un élément dans une liste

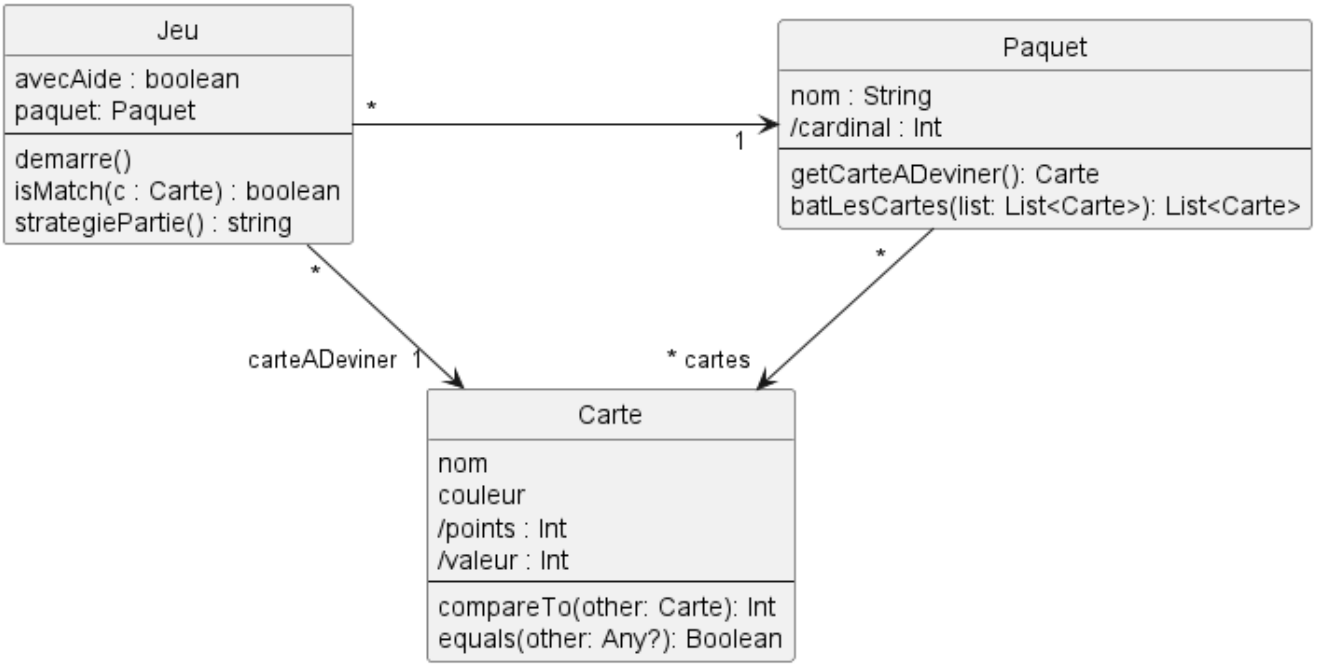
- Analyse des actions du joueur (fonction du nombre de cartes, aides à la décision)

Scénario typique d'une partie

1. (optionnel pour le joueur) paramétrage du jeu (par exemple choix du jeu de cartes, activation de l'aide à la recherche, ...)
2. Lancement d'une partie (le jeu instancie un jeu de carte et tire une carte "au hasard"), que le joueur doit deviner en un nombre de proposition de cartes **optimal**
3. Le joueur propose une carte
4. Si ce n'est pas la bonne carte, alors si l'aide est activée, le joeur est informé si la carte qu'il a soumise est plus petite ou plus grande que celle à deviner. Retour en 3.

   Si c'est la bonne carte alors la partie se termine (passe à l'étape suivante). Le joueur peut choisir d'abandonner la partie (passe à l'étape suivante), ou de retenter sa chance. Retour en 3
5. Le jeu affiche des éléments d'analyse (nombre de fois où le joueur a soumis une carte, ses **qualités stratégiques**, ...)
6. Fin de la partie.

Diagramme de classes (essentiel)



- Une instance de `Jeu` est liée à une instance de `Paquet` (un jeu de cartes) et à une instance de `Carte` nommée `carteADeviner` (la carte à deviner)
- Une instance de `Paquet` est liée à une collection de cartes nommée `cartes`.
- La classe `Jeu` est responsable de la logique du jeu.
- La classe `Paquet` définit la structure d'un jeu de cartes classique (de type jeu de 32 ou 52 cartes) et ses méthodes.

- La classe `Carte` définit la structure d'une carte à jouer et ses méthodes. Une carte à un "nom" (*VALET*, *HUIT*, ect.) associé à un nombre de *points* (*DIX* vaut 10 points par exemple), et une "couleur" parmi *TREFLE*, *PIQUE*, *CARREAU*, *COEUR* ainsi qu'une relation d'ordre (entre carte à déterminer)

En résumé : Une instance de `Jeu` est reliée, à un instant *t*, à

- un paquet de cartes, lui-même relié à un ensemble de cartes (`cartes`), et
- une instance de `Carte` (`carteADeviner` est la carte que le joueur doit deviner)

Travail réalisé

Je vais ici vous presenter et expliquer mon travail comme mes différents tests unitaires et comment mon code fonctionne

Challenge 1

Dans ce challenge nous avons dû rendre opérationel 4 tests unitaires.

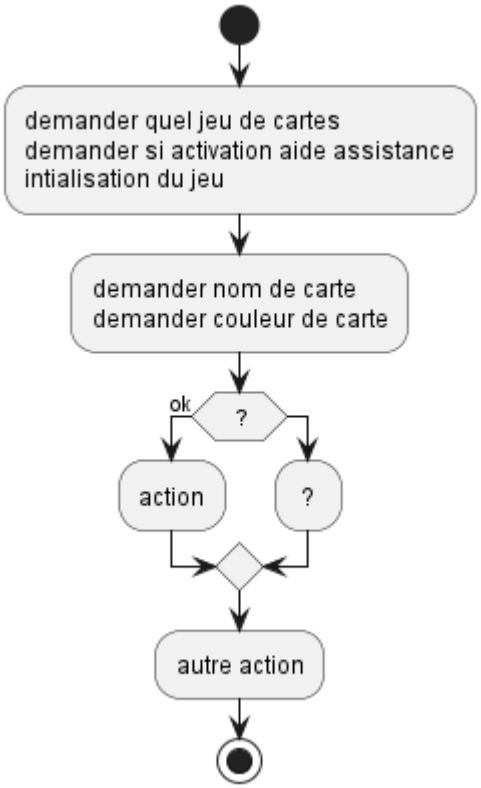
1. `fabriqueDe52Cartes()`
2. `fabriqueDe32Cartes()`
3. `testGetCartes()`
4. `compareCartesDeCouleurDifferenteMaisDeMemeValeur()`
    - Pour les fonctions de test frabique de cartes, je fais appel à ma methode createJeu??Cartes() qui me renvoie une liste de cartes, oû je vais tester si ma carte ici Deux de Trefle se trouve ou pas dans la liste instancié qui ne devrait pas etre dans la liste de 32 cartes mais etre dans celle de 52 cartes.
      Je n'ai pas fais de test testGetCartes() puis'que je test deja dans les tests de fabrique la taille de la liste ainsi qu'une carte dans la liste
    - Pour la fonction compareCartesDeCouleurDifferenteMaisDeMemeValeur()
      ici j'ai du modifier la methode : compareTo(other: Carte): Int
      pour que si les valeurs sont egales elle se compare en fonction de leurs couleurs

Challenge 2

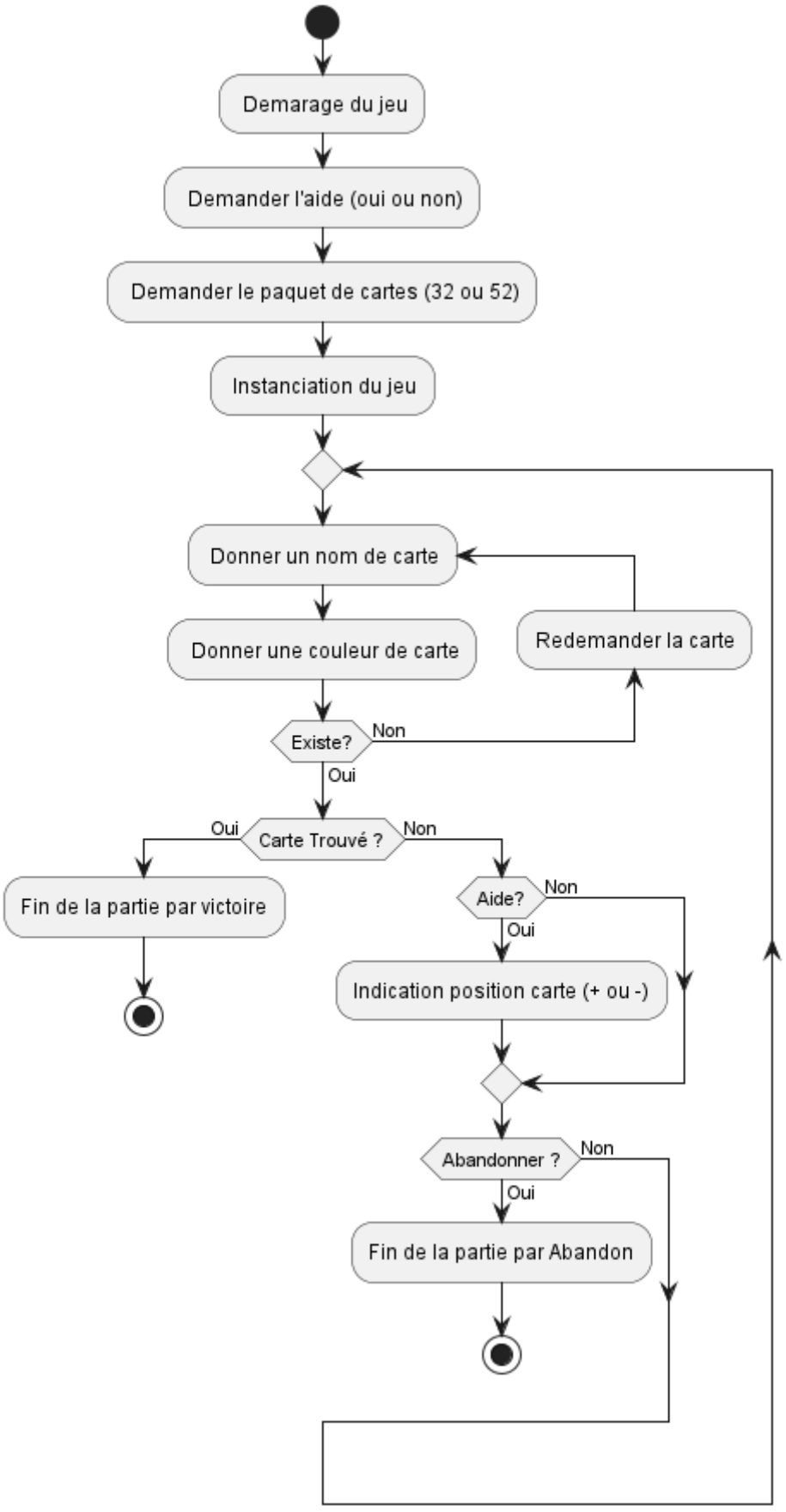
Dans ce challenge nous avons dû realiser tout les ***_TODO*** de la console de jeu pour ensuite le rentranscrire avec un algorithme._

- ***TODO : demander au joueur s'il souhaite avoir de l'aide pour sa partie***
  Pour ce TODO j'ai cree une boucle qui va demander si oui ou non le joueur souhaite de l'aide
- ***TODO : demander au joueur avec quel jeu de cartes 32 ou 52 il souhaite jouer***
  Pour ce TODO j'ai cree une boucle qui va demander quel jeu de carte le joueur souhaite utiliser 32 ou 52 pour ensuite l'instancier dans le jeu
- **_TODO : si l'aide est activée, alors dire si la carte proposée est plus petite ou plus grande que la carte à deviner**
  Pour ce TODO je fais appel à ma methode CompareTo pour indiquer la position de la carte a deviner (+ ou -)
- ***TODO : permettre au joueur de retenter une autre carte (sans relancer le jeu) ou d'abandonner la partie***
  Pour ce TODO j'ai cree une variable nommé repeat: Int qui vaut 0 et qui prendra +1 uniquement si le joueur trouve la carte a deviner ce qui fais que tant que cette option n'est pas realisé le joeur pourra retanter sa chance.
  pour l'option abandonner j'ai cree une boucle ou si le joueur dis oui ma variable repeat prend +1 ou si non le joueur pourra retender une carte
- ***TODO : Présenter à la fin la carte à deviner***
  Pour de TODO j'ai juste recuperer la methode carteADeviner() et je l'ai affiché avec un print

Algo de base



Algo aprés TODO realisé



Challenge 3

Dans ce challenge nous avons dû realiser une fonction batLesCartes() qui melange les cartes d'un paquet.

- pour ceci j'ai utilisé une methode de Kotlin : .shuffled()
  qui ici va recupere une liste de carte la melanger et resortir cette meme liste avec leurs index indexé de facon aleatoir
- Pour la tester j'ai cree une liste de carte allan 2 a 10 que je vais melanger avec ma methode cree juste avant : batLesCartes()
  et ou je vais apres verifier si l'index 1 de mon paquet initial ne correspond pas a l'index 1 de mon paquet melangé

Challenge 4

Dans ce challenge nous devion afficher la srategie du joueur pour cela j'ai elaborer une strategie de facon subjective (selon moi).

- j'ai commencé par recuperer les infomation primordiales pour faire une stragie c'est-à-dire le nombre d'essaie que le joueur a tenté dans sa partie et si il a abandonné
- Ensuite grace a ces informations si le joueur a abandoné il n'a pas de stategie
- Si le joueur a joué avec l'aide je recupere le nombre de carte avec lequel il a joué et grace a la fonction **log2()** de kotlin qui determine de facon dichotomique le nombre d'essaie qui faudrait mathematiquement pour trouver la carte, je vais comparer le nombre d'essaie du joueur et le resultat de l'operation log2(52 ou 32) pour etablir le niveau ou la chance du joueur
- Si le joueur n'a pas activé l'aide il a en theori 1 chance sur 32 ou 52 celon le jeu choisit pour trouver la bonne carte
  dans ma strategie j'ai estimé que 30% de chance etait mon point de chute entre "pas de chance" et "a de la chance"

Conclusion

Ce projet ma beaucoup aider pour l'initiation à Kotlin.