Derrière la bataille

Dans ce TP nous allons révisiter le jeu de bataille que vous avez vu pendant le premier TP. Au lieu d'implémenter le jeu lui-même, vous allez implémenter les classes qui "font le travail".

Les classes que vous avez implémenté la dernière fois, `JeuBataille.java` et `Main.java`, sont disponible pour téléchargement sur eCampus.

Objectif(s)

- ✯ correctement implémenter des classes qui sont utilisée par d'autres.
- ✯ lire et comprendre la JavaDoc des classes du JDK (Java Development Kit)

Exercice 1 – The paquet de cartes

Pour le premier exercice, vous implémentez le paquet de cartes, dans la classe `PaquetCartes`.

Question 1

Les cartes seront stockées dans une `ArrayList`, qui est, évidemment, privé. Une `ArrayList` est comme un tableau qui peut dynamiquement changer sa taille. La JavaDoc de cette classe est disponible à `https://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html`.

Quand vous déclarez l'attribut, vous devez spécifier le type d'éléments qui seront stockés dans la liste utilisant la syntaxe `ArrayList<Carte>`.

Le constructeur de la classe ne prend pas de paramètre, mais instancie l'`ArrayList`.

Question 2

La méthode `ajouter()` prend une Carte en paramètre et l'ajoute à la liste.

La méthode `getNbCartes()` retourne le nombre de cartes restant dans le paquet.

La méthode `estPaquetVide()` retourne "vrai" si le paquet est vide, non sinon.

Question 3

La méthode `tirerCarteDessus()` retourne la première Carte de la liste et la **supprime**.

La méthode `mettreCarteDessous()` prend une Carte un paramètre et l'ajoute à la fin de la liste.

Question 4

La méthode `toString()` construit au fur et à mesure la représentation textuelle du paquet. Pour rendre la représentation plus lisible, nous passons à la ligne prochaine après quatre cartes.

Question 5

La méthode `melanger()` mélange le carte en prenant la carte à une position aléatoire, la supprimant, et l'ajoutant à la fin de la liste.

Pour générer un nombre aléatoire, vous utilisez la méthode `nextInt()` de la classe `Random` dont la JavaDoc se trouve à `https://docs.oracle.com/javase/8/docs/api/java/util/Random.html`.

On répète cette opération dix fois le nombre de cartes dans le paquet.

Exercice 2 – Les cartes

Question 1

Les cartes sont représentées par deux entier : un pour sa hauteur dont les valeur vont 7 (por le 7) à 14 pour l'as, l'autre pour la couleur dont les valeur sont 0 : trefle, 1 : carreau, 2 : coeur, 3 : pique.

Pour aider l'implémentation de la méthode `toString()`, la classe contient aussi deux tableaux de chaîne de caractères qui sont des **attributs de classe**. Ces tableaux traduisent des entiers en chaîne de caractères, pour les hauteurs 0 – 7 sont traduis en "inconnu".

Question 2

La classe ne contient pas de setter, mais de getter.

Question 3

Le seul constructeur de la classe prend deux paramètre de type `int`. Il valide les valeurs passées en paramètre et s'ils sont illégaux, il indique l'erreur en utilisant l'instruction `throw new IllegalArgumentException("Carte illegale");`

Question 4

La méthode `toString()` exploite les tableaux mentionnés plus haut afin de générer une chaîne de caractères lisible par des humains.

Exercice 3 – Les joueurs

Chaque joueurs a deux attributs : un `PaquetCartes` et une chaîne de cartes qui représente son nom.

Question 1

Le seul constructeur de la classe prend un paramètre du type chaîne de caractères.

Question 2

La classe n'a pas de setter, mais des getters.

Question 3

La méthode `nbCartesRestantes()` retourne le nombre de cartes qui reste dans le paquet.
La méthode `prendreCarteDessus()` retourne la première carte du paquet.
La méthode `mettreSousLePaquet()` prend une Carte en paramètre et la met sous le paquet.

Question 4

La représentation textuelle d'un joueur contient son nom et la représentation textuelle du paquet.