Simonin Enzo Simon Pereaux

**Compte Rendue SAE JAVA :**

**Premières séances de 2 heures :**

*-Création de la classe Carte et de son constructeur (en prenant un paramètre de type String) :*  
 🡪 il fallait utiliser la méthode ‘char charAt(int i)’ de la classe String pour convertir un élément d’une chaine de caractère à un indice i en Char.  
 🡪 et la méthode ‘int Integer.parseInt(String s)’ pour convertir la chaine de caractère (avec seulement des chiffres) en un entier pour devenir la Date de la carte.

*-Création de la Méthode toString de la classe Carte avec la condition de savoir si la carte et sur la face recto ou verso :*  
 🡪 Créer une variable de type String qui est vide  
 🡪 On affiche oui ou non la Date de la carte si la carte est sur la face verso(non) ou recto(oui)

*-Création de la classe MainCarte pour tester le fonctionnement de la classe Carte :*  
 🡪 Tout d’abord nous avons recréer une méthode dans la classe Carte appelé retournerCarte qui permet de retourner la Carte pour la passer de recto à verso ou inversement, elle permet de voir la date de la carte ou non

🡪 Dans ce main nous avons créés un objet LectureFichier avec comme parametre notre fichier texte contenant les cartes puis nous stockons les cartes dans un tableau de chaine de caractère nommé fich.  
 🡪 Ensuite pour chaque ligne de ce tableau on :

* créé une carte avec le contenu de la ligne
* affiche grâce à la méthode String de la classe Carte
* la retourne grâce à la méthode retournerCarte de la classe Carte
* affiche la carte qui est maintenant retourné donc nous pouvons voir la date de la carte

*-Faire les tests de la classe Carte*

🡪 Nous avons écrit plusieurs test pour tester les différentes méthodes de la classe Carte :

* Test méthodes carte qui permet de tester si les méthodes de paquet sont utilisables et présentes
* test1 : permet de tester que le constructeur de la classe Carte fonctionne correctement
* test2 : permet de tester la méthode toString, que la carte soit face verso ou non
* test3 : permet de tester la méthode retournerCarte

🡪 Ces tests ont tous été validé du premier coup

*-Création de la Classe Paquet*  
 🡪Nous avons repris la classe Paquet du TP11 comme demander et l’avons adapté à la SAE.  
(nous n’avons eu aucun problème pour cette étape)

*-Ajout d’un constructeur supplémentaire à la classe Paquet*

🡪 Ce constructeur permet grâce à un nom de fichier en paramètre de lire ce fichier et de transfomer chaque ligne en Carte grâce au constructeur de la classe Carte.

(nous avons rencontré quelques soucis pour comprendre le fonctionnement de la classe lectureFichier )

*-Création de la méthode piocherHasard*  
 🡪Création de la méthode piocherHasard dans la classe Paquet qui permet de piocher une carte au hasard dans un paquet et de la retirer du paquet

*-Complétion de la méthode toString de la classe Paquet*  
 🡪 Nous avons complété la méthode toString de la classe Paquet du tp11 pour lui permettre d’afficher un tableau de carte, cette méthode lis le tableau et afficher les cartes au fur et à mesure

**Deuxième séance de 2 heures :**

*-Complétion et ajout de test dans la classe Paquet*

🡪 Nous n’avons pas eu besoin de modifie les tests :

* test1 qui test le constructeur vide
* test3 getCarte qui permet de retourner la carte à l’indice en paramètre
* Nous avons du créer  les tests :
* Test méthodes paquet qui permet de tester si les méthodes de paquet sont utilisables et présentes
* test2 qui permet de tester le constructeur qui prend un nom de fichier en paramètre
* test4 qui permet de tester ce que fait getCarte lorsque la carte est en dehors du tableau
* le test7 qui permet de tester si la méthode piocherHasard
* le test8 qui permet de tester la méthode toString
* Nous avons du adapter les tests :

- test5 qui permet de tester ajoutCarteFin a été adapté pour nos cartes qui non plus seulement une date mais également un évènement et un côté

- test6 qui permet de tester retirerCarte a été adapté pour nos cartes qui non plus seulement une date mais également un évènement et un côté

(Pour cette partie nous avons rencontré quelque problème avec le test8 de la méthode toString causé par des références à des objets et nous les avons réglés grâce à des variables tampons)

*-Création de la classe Frise et de son constructeur*

🡪 Le constructeur de la classe Frise prend un tableau de carte en parametre qu’il trie en fonction de la date des cartes

*-Création de la méthode ajouterCarteTrie*

🡪 Cette méthode prend une carte en paramètre et l’ajoute au bon endroit dans la frise en fonction de sa date, elle :

* crée un nouveau tableau de carte avec une case en plus
* remplis le tableau jusqu’à tombé sur une case où la date de la carte est supérieure à la date de la carte en paramètre
* ajoute la carte en paramètre
* remplis le tableau avec toutes les valeurs suivantes

*-Création de la méthode vérifierCarteApres*

🡪 Cette méthode prend en paramètre une carte et un entier p elle permet de vérifier si la carte en paramètre peut être placé après la place p en comparant la date de la carte en paramètre et c’elle après la place p et renvoie si oui ou non elle peut être placé

*-Création de la méthode insererCarteApres*

🡪 Cette méthode permet d’insérer une carte c après la position p si cela est possible et renvoie si oui ou non elle a placé cette carte

**Fin de la première semaine (envoie du rendu intermédiaire)**

**Deuxième semaine :**

**Troisième séance de 2 heures :**

*-Création des tests de la classe Frise*

🡪 Nous avons crée les tests :

* Test méthodes frise qui permet des tester si les méthodes de la classe frise sont éxistante et utilisable
* test1 constructeurParamOK qui permet de testé le deuxième constructeur
* test2 constructeurSansCarteOK qui permet de tester les constructeur vide
* test3 verifierCarteApresOK qui permet de tester que verifierCarteApres fonctionne correctement
* test4 verifierCarteApresIndiceTropEleve qui test la méthode verifierCarteApres lorsque l’indice donné en paramètre est trop élevé
* test5 insererCarteApresOK qui permet de tester que insererCarteApres fonctionne correctement
* test6 insererCarteApresIndiceTropEleve qui test la méthode insererCarteApres lorsque l’indice donné en paramètre est trop élevé

(Nous avons rencontré des difficultés pour réussir les tests sur ces 2 méthodes)

**Quatrième séance de 2 heures :**

* test7 qui test la méthode ajouterCarteTrie qui test que la méthode ajouterCarteTrie fonctionne correctement
* test8 qui permet de tester la méthode toString
* test9 qui permet de tester l’ajout d’une carte trie dans une liste vide
* test10 qui permet de tester la verification de verifierCarteApres dans une frise vide
* test11 qui permet de tester insererCarteApres dans une liste vide

(Nous avons rencontré de difficultés lorsque l’on souhaite ajouter une carte en -1 et pour faire fonctionner les tests de la méthode ajouteCarteTrie)

**Cinquième séance de 2 heures :**

*-Création de la classe Jeu*

🡪 Nous avons créée la classe Jeu et son constructeur qui prend en paramètre un entier tailleMain et une chaine de caractère nomFicher

* Ajout d’un constructeur vide dans la classe Frise qui permet de créer une file vide
* Ce constructeur permet de créer un Paquet pioche en y insérant les valeurs du fichier portant le nom donné en paramètre et d’ajouter un nombre tailleMain de carte du fichier dans le Paquet mainJ qui représente la main du joueur

*-Création de la méthode fin*

🡪 la méthode fin renvoie un booléen qui indique si oui ou non le jeu est fini

* Le jeu ce termine lorsque la main du joueur ou la pioche est vide

*-Création de la méthode fonctionnement­*

🡪 la méthode fonctionnement permet le déroulement du jeu

* Cette méthode commence par afficher la main du joueur et lui demande d’y choisir une carte ensuite elle retourne la carte et l’ajoute à la frise, retire la carte de la main

**Sixième séance de 2 heures :**

*-Continuation de la méthode fonctionnement*

🡪 Cette méthode fait ensuite plusieurs instructions en boucle dans que la méthode fine ne soit pas vérifiée, elle :

* Affiche la frise
* Affiche la main du joueur
* Demande quelle carte de la main à ajouter
* Demande derrière qu’elle carte l’ajouter
* Si la carte peut être ajouter, elle :
  + - La retourne
    - L’ajoute
    - La retire de la main
    - Pioche une carte
* Et si elle ne peut pas être ajouté elle rajoute une carte dans la main
* Lorsque la partie est terminé cette méthode affiche victoire si la main du joueur est vide ou défaite si la pioche est vide

*-Création de la classe main ProgJeu*

-Cette classe créer un objet jeu et utilise la méthode fonctionnement pour lancer le jeu

*-Génération de la javadoc*

🡪 javadoc -d javadoc src/\*.java