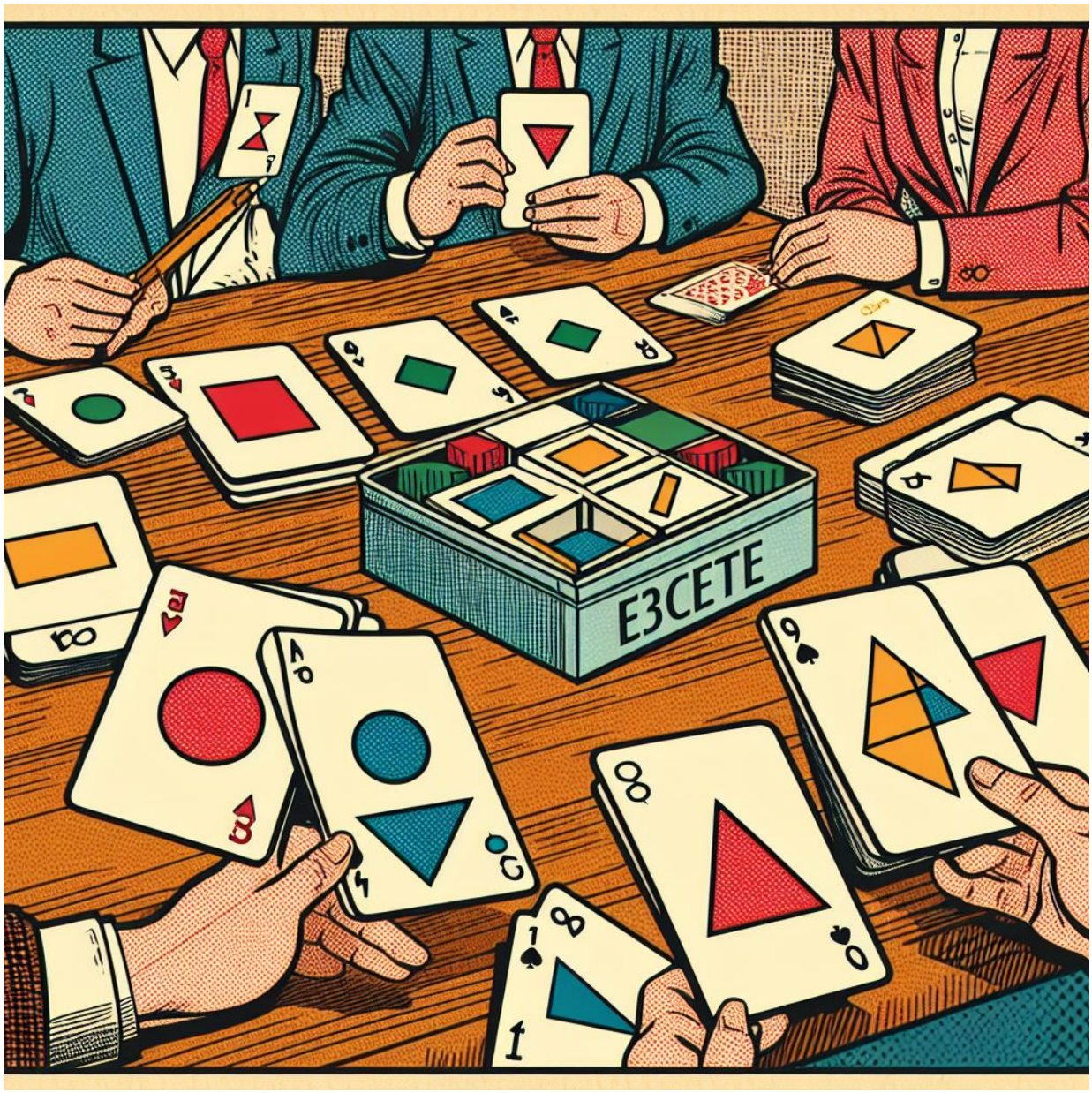
Développement Initiatique

SAE 1.02 : E3Cète



IUT de Montpellier-Sète - Année 2023 - 2024

Dans cette SAE nous vous proposons de nous intéresser à un problème que tout informaticien se doit de connaître : les tris !
Dans une première partie nous allons modéliser une jeu de cartes (au sens physique du terme). L'objectif de cette partie est de réaliser différents tris et de les comparer. Enfin dans une seconde partie, nous allons utiliser ce paquet de cartes pour en faire un vrai jeu , auquel il est possible de jouer. Inutile de chercher un lien entre les deux parties de cette SAE, vous risquez de ne pas le trouver ;)

1 Modélisation et comparaison d'algorithmes de tris

1.1 Pratique

Dans cette partie nous souhaitons construire un $Paquet$ de $Carte$s afin de pouvoir les trier.

Exemple 1.

Cette $Carte$ est :
de $Couleur$ verte
ayant une $Texture$ pleine
et comme $Figure$ un losange
repété 3 fois.

1.1.1 La classe $Carte$

La classe $Carte$ représente une carte possédant une $Figure$ répétée un certain nombre de fois avec une $Texture$ et une $Couleur$.

Pour cette classe, on a besoin de connaître :
- La $Figure$ représentée
- Le nombre de fois où la $Figure$ est représentée.
- La $Couleur$ de la $Figure$.
- La $Texture$ de la $Figure$ (pleine, hachurée,...).

1.1.2 La classe $Paquet$

La classe $Paquet$ représente un ensemble de $Carte$s.
Elle est similaire à la classe EE vue en TD, en remplaçant les entiers par des $Carte$s : les $Carte$s sont stockées dans un tableau dont la longueur est le nombre de $Carte$s du jeu complet, et un indice (entier) permet de stocker le nombre de $Carte$s restantes (non piochées). Dans cette partie de la SAE (et uniquement), toutes les $Carte$s doivent toujours être disponibles (aucune $Carte$ n'est retirée/ignorée) du $Paquet$.

Pour cette classe, on a besoin de connaître :
- Le tableau stockant les $Carte$s.
- Le nombre de $Carte$s restantes dans le $Paquet$.

1.1.3 Une énumération : qu'es aquo ?

Une énumération ou liste énumérative est un ensemble de valeurs constantes. En java on utilise le mot clé $enum$. Par exemple, voici une énumération des civilités usuelles en Java.

Exemple 2.
```
public enum Civilite {
   MONSIEUR,
   MADAME,
   MADEMOISELLE;
}
```
Il est possible d'accéder aux valeurs de cette énumération depuis une autre classe en utilisant $Civilite.values()$; qui retourne une tableau de $Civilite$.

De plus, ces valeurs sont ordonnées par l'ordre d'énumération donné dans la définition. Il est possible de connaître leur "rang" avec l'expression suivante (où c est de type Civilité) : $c.ordinal();$

Il est également possible d'enrichir une énumération comme dans l'exemple suivant (que vous êtes invités à tester).

```
public enum Civilite {

    MONSIEUR("MR"),
    MADAME("MME"),
    MADEMOISELLE("MLLE");

     private String abreviation ;

     private Civilite(String a) {
         this.abreviation = a ;
    }
     public String getAbreviation() {
         return  this.abreviation ;
    }
}
```

1.1.4 Un affichage en couleur sur le terminal

Dans cette SAE, il n'est pas question de proposer une interface graphique (c'est au programme du S2), cependant cela ne nous interdit pas d'avoir une expérience avec le terminal qui soit agréable à l'utilisateur.

Exemple 3. Il est possible de colorer du texte affiché dans un terminal avec le code suivant (que vous êtes invités à tester).

```
String texteColore = "\u001B[31mCoucou!";
System.out.println(texteColore); //Affiche le texte ("Coucou!") en rouge
```

Il est également (fort) possible que vous ayez besoin de réinitialiser la couleur du texte en utilisant un autre code couleur particulier. On ne vous en dit pas plus, on vous laisser chercher.

1 Écrire les énumérations correspondants à $Couleur$, $Figure$ et $Texture$ en donnant 3 valeurs possibles pour chaque caractéristique pour l'instant. Vous pourrez augmenter cette valeur pour le rapport demandé dans la suite.

2 Implémenter les méthodes spécifiées dans les classes $Carte$ et $Paquet$ (piocher et peut piocher serviront dans la seconde partie de cette SAE, vous pouvez donc les implémenter dans un second temps) sans oublier de définir les attributs.

1.2 Théorie

Créer un document pdf "SAE1.02-Maths-NOM1-NOM2" que vous compléterez au fur et à mesure de cette SAE. On souhaite comparer les trois algorithmes de tris que vous avez implémenté dans la parie précédente.

Vous pouvez utiliser l'instruction suivante pour obtenir le temps de tri d'un $Paquet$ avec la méthode $triSelection()$ par exemple

```
long tempsExec = Ut.getTempsExecution(paquet::triSelection);
```

La méthode $getTempsExecution$ est donnée dans Ut.
Le temps d'exécution étant très rapide sur des $Paquet$s ayant peu de $Carte$s, ajouter un compteur permettant de compter approximativement le nombre d'opération élémentaires nécessaires à chaque tri(affectation/ lecture d'une variable, test d'une conditionelle..).

Ecrire une méthode $testTri()$ qui génère un $Paquet$ (réaliser les copies nécessaires), trie le $Paquet$ avec les trois versions et affiche le temps nécessaire ainsi que le nombre d'opérations élémentaires approximatif nécessaires à chaque tri.

3 Réaliser un graphique pour chaque tri faisant apparaître le temps et le nombre d'opérations élémentaires approximatif nécessaire pour trier un $Paquet$ de $Cartes$ en fonction du nombre total de cartes.
Vous devez également proposer une analyse du graphique obtenu afin de comparer les différents tris. Faire des commentaires et émettre des conjectures quand aux "performances" de chaque tri et essayer les comparer.

Vous détaillerez tout le protocole de test de manière précise :
- nombre total de $Carte$s, noté n, à faire évoluer en modifiant la cardinalité des caractéristiques possibles pour une $Carte$ tel que $10 < n < 500$ environ ,
- le nombre de $Paquet$ différents triés pour chaque valeur de n (choisir une valeur assez importante pour que la moyenne du temps nécessaire soit significative, 1000 environ),
- machine utilisée ,
- timeout eventuel ...

Vous pouvez utiliser le logiciel de votre choix pour réaliser ce graphique.

4 Au premier semestre vous avez vu ce qu'est une combinaison, un arrangement ainsi qu'un n-uplet. D'un point de vue mathématiques, qu'est ce qu'une $Table$? Justifier votre choix.

Pour les questions suivantes, la $Table$s contient 9 $Carte$s et le jeu en contient 81.

5 Combien y-a-t-il de $Table$s différentes possibles ?

6 Combien y-a-t-il de $Table$s contenant exactement 3 $Carte$s rouges ?

7 Déduire, des questions, précédentes, la probabilité d'avoir une $Table$s contenant exactement 3 $Carte$s rouges.

8 Ecrire une fonction $proba3CR$ (comme probabilité 3 $Carte$s rouges) qui permette de "retrouver" cette probabilité théorique en faisant augmenter le nombre d'essai.

9 Combien y-a-t-il de $Table$s contenant exactement 3 $Carte$s rouges et 2 $Carte$s ayant au moins un losange ?

10 En utilisant la méthode $estUnE3C$, faire une estimation (étude expérimentale) de la probabilité d'avoir un $E3C$ sur une $Table$, en faisant augmenter le nombre d'essai.

Pour les questions 8 et 10 vous ferez une courbe montrant la fréquence en fonction du nombre d'essai. En utilisant ces courbes, vous préciserez le nombre d'essai qui vous semble être "nécessaires" pour avoir un résultat expérimental au plus proche du théorique .

Pour toutes ces questions, vous pouvez vous rapprocher de votre enseignant Maths Discrètes ou de Mr MARIE-JEANNE (par mail) qui a participé à l'élaboration de ce sujet.

2 E3Cète

Dans cette partie nous allons pouvoir jouer au célèbre jeu "E3Cète". L'objectif est de réaliser des ensembles de trois $Carte$s, notés $E3C$. Un ensemble de trois $Carte$s est un $E3C$ si et seulement si :

Pour chaque caractéristique, l'ensemble des trois $Cartes$ doivent être toutes égales ou toutes différentes .
.

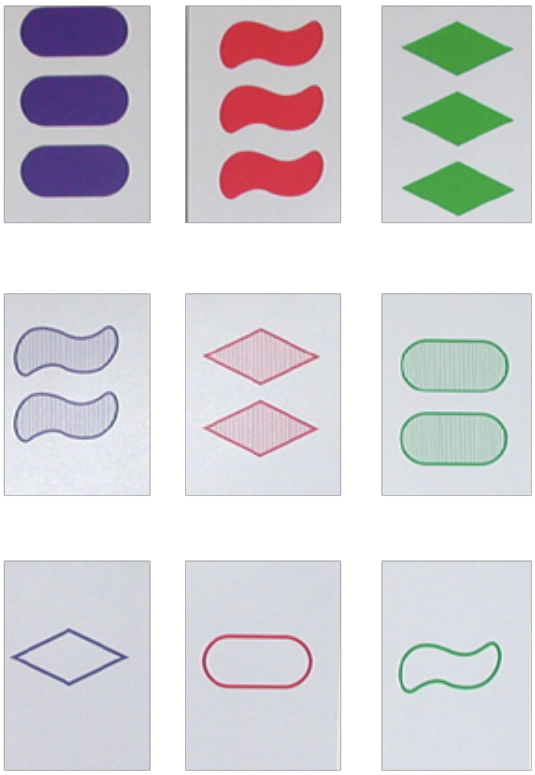
Les règles sont les suivantes :
- On possède un $Paquet$ de $Carte$s ou chaque $Carte$ est unique et représente une $Figure$ répétée (entre 1 et 3 fois) avec une $Texture$ et une $Couleur$.
- Une $Table$ permet de stocker les 9 cartes qui sont faces visibles. Au début de la partie, on dispose 9 cartes sur la $Table$, piochées sur le dessus $Paquet$.
- A chaque tour, le joueur doit essayer de trouver un ensemble de trois $Carte$s, noté $E3C$. S'il n'en trouve pas, il doit éliminer des cartes.
- Le joueur doit désigner trois cartes par leurs $Coordonnees$. Si ces cartes forment un $E3C$, il gagne trois points sur son score.
Si ce n'est pas un $E3C$, il perd 1 point sur son score.
- Les trois $Carte$s sont remplacées par de nouvelles $Carte$s piochées dans le $Paquet$.
- La partie se termine quand il n'y a plus de $Carte$s dans le $Paquet$ (même s'il reste des $Carte$s sur la $Table$).

La classe $Table$ représente une table de jeu contenant sur laquelle on dépose des $Carte$. Les $Carte$ sont affichées sous forme matricielle mais sont stockées dans un tableau. Quand elle est initialisée, la $Table$ est vide.

Pour désigner une carte sur la $Table$, on utilise des coordonnées de la forme (l,c) ou l représente un numéro de ligne et c un numéro de colonne .
On mémorise les cartes sur les cases de la première ligne, puis on continue sur celles de la seconde, etc...
Les cartes sont donc indexées dans le tableau en les parcourant de gauche à droite, puis de haut en bas sur la représentation matricielle.

La classe Jeu permet de faire des parties du jeu "E3Cète" soit avec un humain, soit avec un ordinateur,
Pour cette classe, on a besoin de connaître :
- Un $Paquet$ pour stocker toutes les cartes et avoir une pioche.
- Une $Table$.
- Le score du joueur (humain ou ordinateur).

Exemple 4. Voici un exemple de $Table$



Les $Cartes$ ayant les numéros suivants forment un $E3C$:
1 , 2 et 3 (toutes égales pour le nombre de répétitions et la $Texture$; toutes différentes pour la $Couleur$ et la $Figure$)
4 , 5 et 6
7 , 8 et 9
1 , 4 et 7 (toutes égales pour la $Couleur$; toutes différentes pour le nombre de répétitions, la $Texture$ et la $Figure$)
2 , 5 et 8
3 , 6 et 9
1 , 5 et 9 (toutes différentes sur toutes les caractéristiques)
1, 6 et 8 ... et il y en a encore pleins d'autres
(Carte 1 en haut à gauche , Carte 3 en haut à droite et Carte 9 en bas à droite)

11 Implémenter les méthodes spécifiées dans $Coordonnees$, $Table$ et Jeu sans oublier de définir les attributs

Vous pouvez ajouter des méthodes que vous jugez nécessaire pour un meilleur découpage et/ou pour respecter le principe DRY.

3 Extensions

Une première partie des extensions est disponible. Vous êtes assez libres sur celles-ci et c'est un choix volontaire de notre part. Ainsi n'attendez pas d'être guidé sur le choix à faire d'un point de vue conception / implémentation. Cependant si l'objectif de l'extension n'est pas clair, nous sommes là pour apporter des précisions.

3.1 Fin de partie

Dans la version de base la partie se termine lorsque le $Paquet$ est vide. Dans cette extension, nous vous proposons de terminer la partie lorsque le $Paquet$ est vide et que la $Table$ ne contient plus d'$E3C$. La $Table$ va donc se retrouver avec des emplacements vides, à vous de gérer ça, on ne vous en dit pas plus ;) .

3.2 Saisie simplifiée pour l'utilisateur

Afin de simplifier la saisie pour l'utilisateur et éviter les erreurs entre la ligne et la colonne, nous vous proposons d'améliorer la saisie des coordonnées en utilisant une lettre pour la ligne et un chiffre pour la colonne. Ainsi, l'utilisateur pourra saisir A,1 au lieu de 1,1.
Dans cette extension (uniquement !), la hauteur et la largeur de la $Table$ seront au maximum égales à 20 (ouf, il y a 26 lettres dans l'alphabet).
Vous devez mettre à jour l'affichage console afin qu'il fasse désormais apparaître cette amélioration.

Il est toujours interdit d'utiliser des Maps ou d'autres structures de donnés. Vous avez (encore et toujours) le droit d'utiliser un tableau, on ne vous en dit pas plus non plus.

3.3 Ensemble de x Cartes : ExC

Dans cette extension nous vous proposons de mettre 20 valeurs différentes dans chaque caractéristiques (vous pouvez mettre forme1, forme2 ... si vous n'avez pas d'imagination). L'objectif est de pouvoir jouer en sélectionnant un sous-ensemble de x valeurs (parmi les 20) pour chaque caractéristique.
Dans cette extension, vous devrez également modifier la méthode $estUnE3C$ par UNE des méthodes suivantes en fonction de votre choix pour x qui retourne vrai si $cartes$ est un ensmeble de x $Carte$s :

```
public static boolean estUnExC(Carte[] cartes, int x)

public static boolean estUnExC(Carte[] cartes)
```

Il y a bien sur d'autres modifications à faire mais on ne vous en dit pas plus non plus.

3.4 Taille de la table

Dans cette extension, nous vous proposons de permettre à l'utilisateur de jouer sur une table qui n'est pas carrée, c'est-à-dire telle que largeur ne soit pas égale à hauteur. Le joueur doit pouvoir préciser la largeur et la hauteur qu'il souhaite.
Il faut penser à vérifier que tous les paramètres soient compatibles et qu'il est réellement possible de jouer une telle partie ou à contrario, que certains ne sont pas compatibles et donnent lieu à une situation impossible (qu'on va bien sur oublier de vous préciser ;))

4 Programme du contrôle du 8 janvier

Est au programme de ce contrôle tout ce qui apparaît dans la version de base : partie mathématiques + E3Cete . Vous devez être capable de répondre aux questions mathématiques posées, de refaire le tris ou encore d'écrire toutes les méthodes nécessaires à la version de base.

5 Modalités du rendu

Deadline : 14 Janvier 23h59

Votre travail de programmation est à poussé sur Gitlab. Il sera partiellement noté par tests automatiques.

Nous vous demandons de faire un package $E3CeteBase$ pour la version de base, puis un package $E3CeteExt12$ si vous décider de faire les extensions 1 et 2 par exemple. Nous vous demandons de regrouper au maximum les extensions dans un même package (si elles ne sont pas en conflit les unes avec les autres).

Le dossier sera "mathématiques" à déposer sur Moodle (à confirmer).