PROJET CPO

Dossier de conception

-Etape 1-

STRI
INFORMATIQUE - RÉSEAUX - TÉLÉCOMS

UNIVERSITÉ TOULOUSE III PAUL SABATIER
Université de Toulouse

L3 IRT-STRI 1A 2021/2022

DUGUAIT Nicolas

KHEMIRI Rania

YAICI Celina

Sommaire:

1 Introduction:

Le sujet de notre projet de CPO (Conception et programmation objet en Java) est une forme de jeu d'échecs personnalisée par chaque groupe. Le jeu est composé d'un tableau avec des cases sur lesquels se déplacent des protagonistes de différents types.
Dans le cas de notre jeu nous avons décidé de créer deux équipes qui s'affrontent. Les sorcier.e.s et les fées de l'univers WINX.
La première étape est de créer un plateau de jeu avec nos protagonistes et des cases. Le programme à cette étape devra permettre de gérer les déplacements des protagonistes selon les coordonnées des cases du plateau de jeu. L'application générale est détaillée à travers des diagrammes UML.

2 Objectif:

Les deux équipes ont pour objectif de se combattre et de gagner des points afin de remporter la partie. Les protagonistes s'affrontent sur l'échiquier créé en premier lieu.
Notre groupe doit donc créer un échiquier et des protagonistes en utilisant le langage de programmation JAVA avec un interface graphique via un IDE (netbeans dans notre cas)

3. Contraintes techniques

Le jeu est associé au cours CPO, il doit donc être codé en utilisant le langage vu en cours qui est Java.
Les notions de programmation orientée objet ayant été abordées en cours, le programme devra essentiellement s'appuyer sur le paradigme de la programmation POO.
L'interface sera réalisée en mode texte dans un terminal

4 Diagrammes UML

3.3 Diagrammes de séquence

Diagramme de séquence :

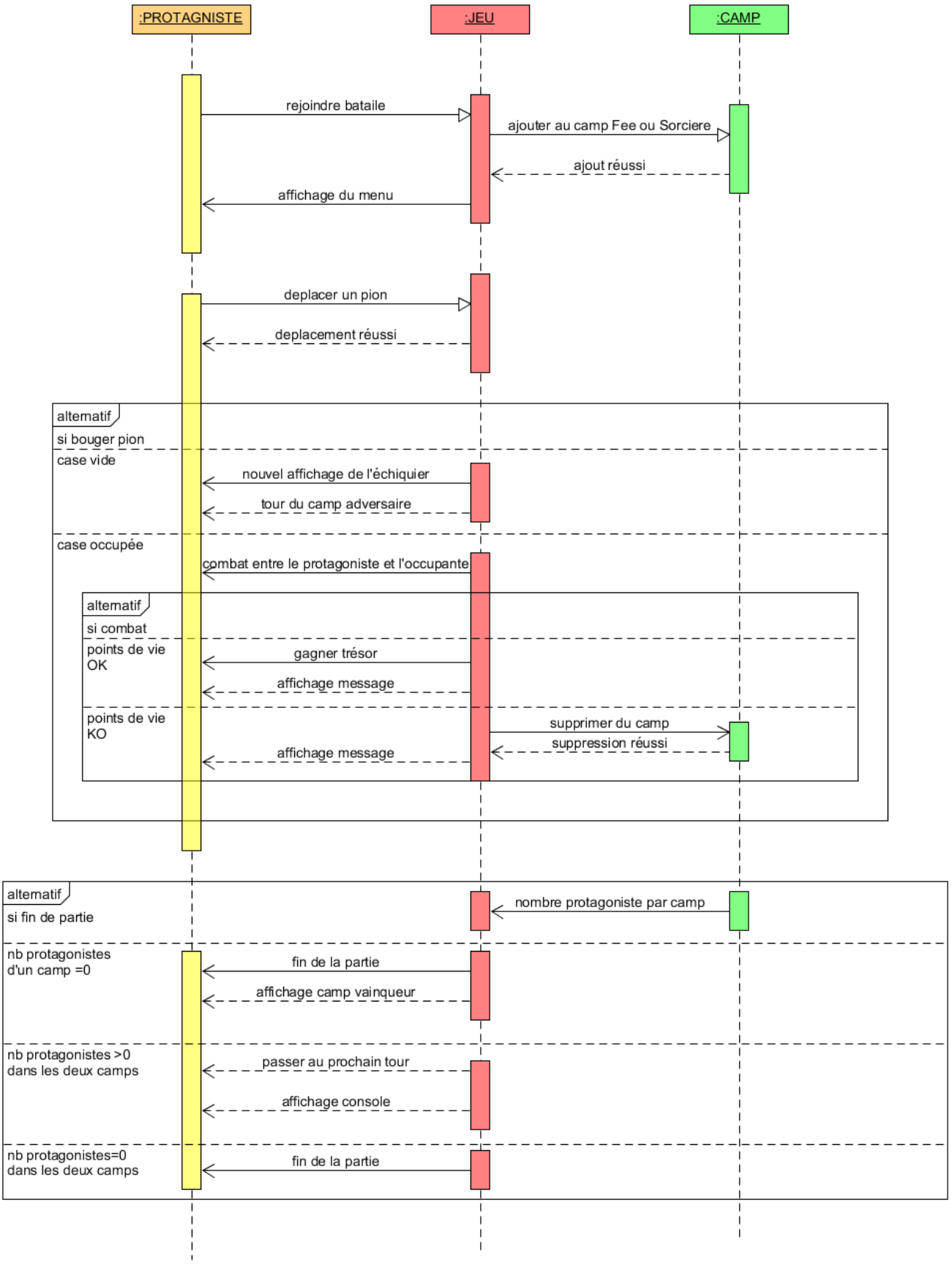


Diagramme de séquence système:

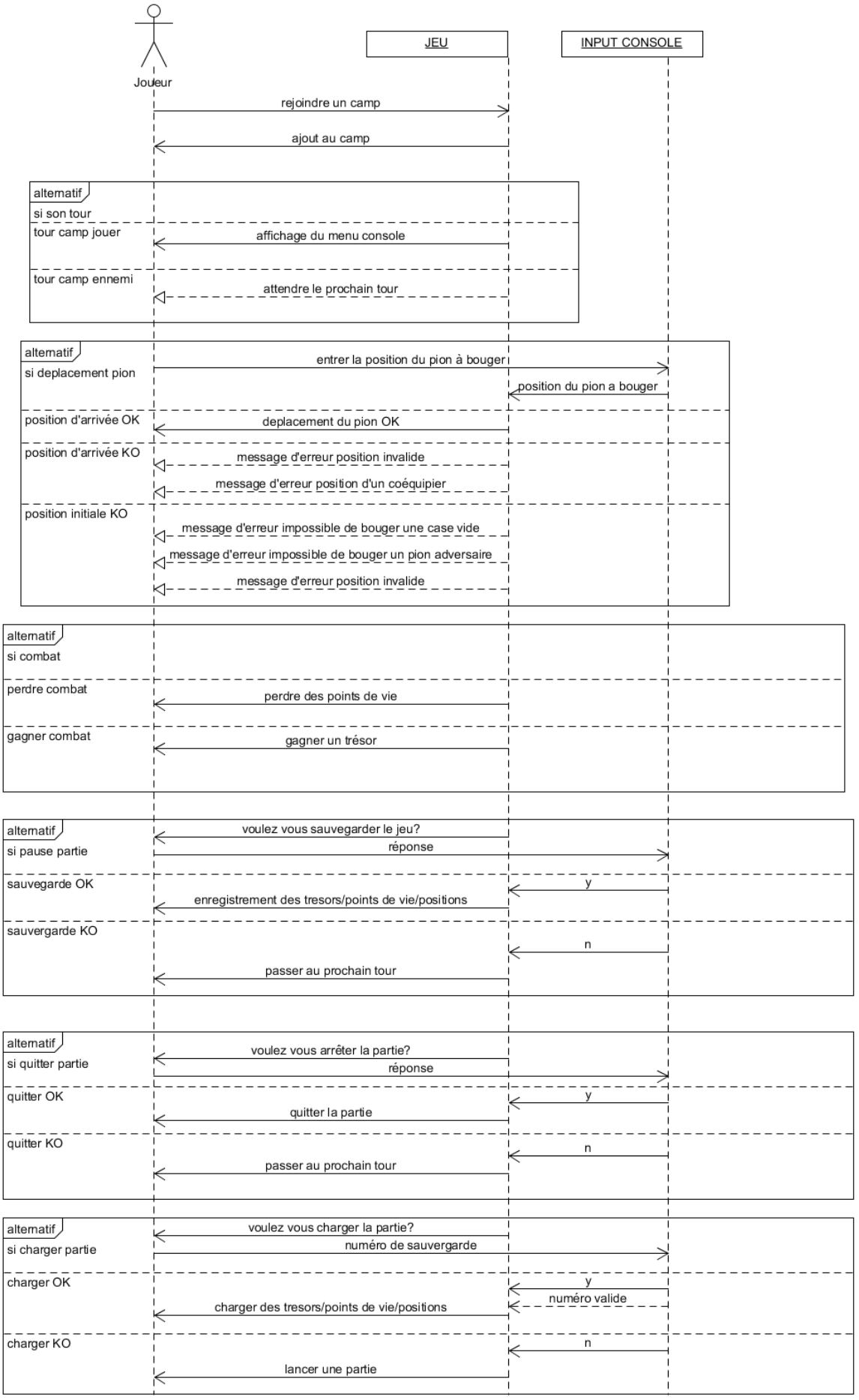


Diagramme de cas d'utilisation:

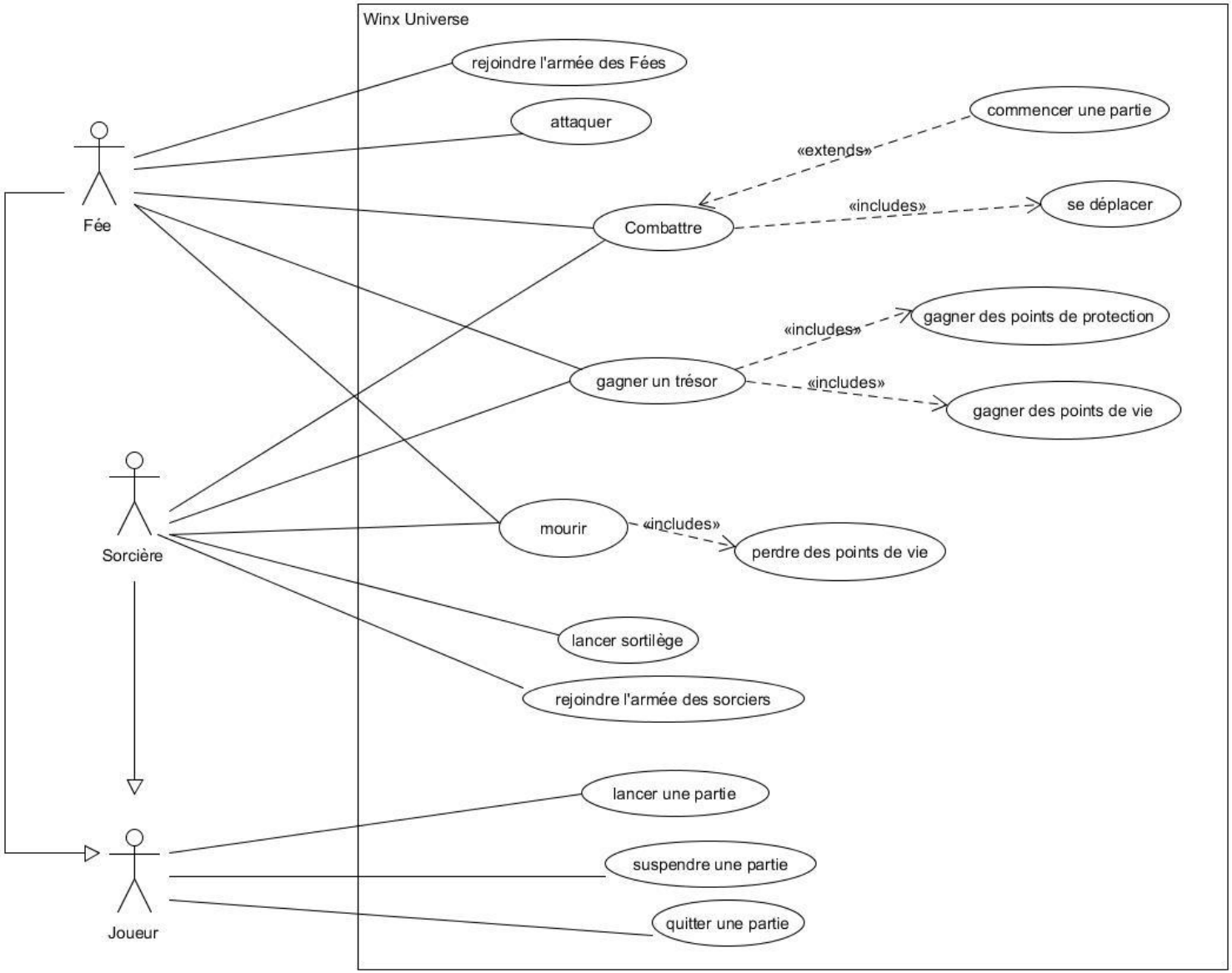
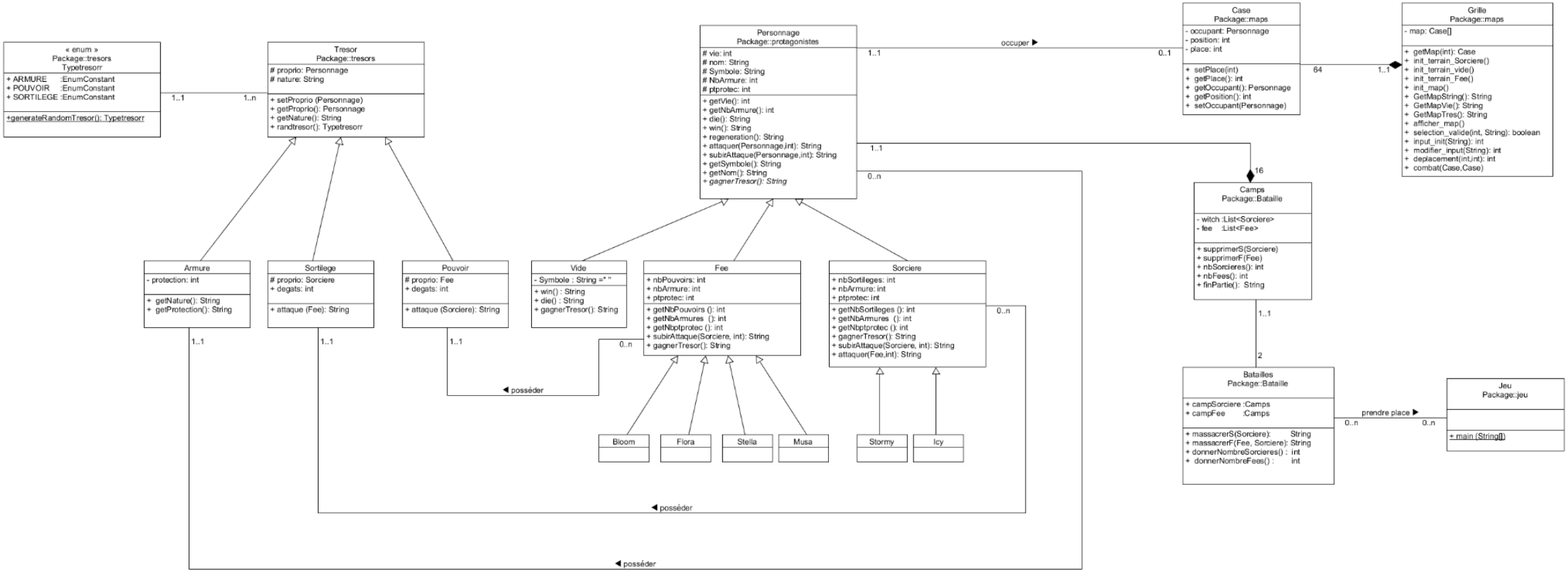


Diagramme de classe:



5 Description du jeu

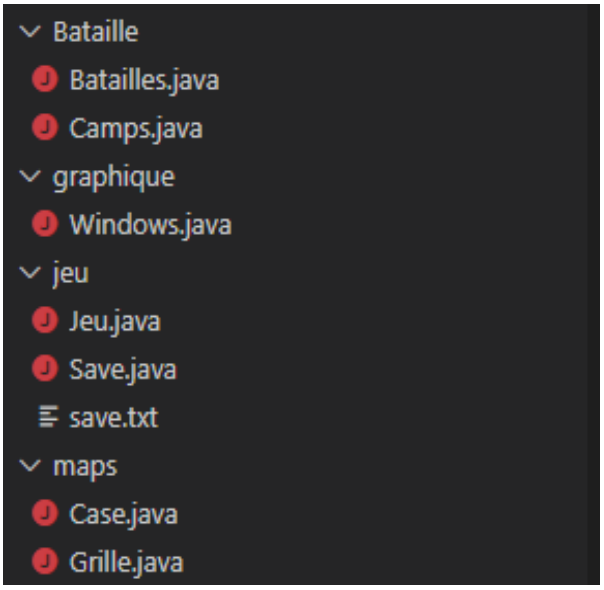
- Les personnages qui s'affrontent sont séparés en deux armées, les fées et les sorcières
- Les fées possèdent des pouvoirs magiques et les sorcières peuvent lancer des sortilèges.
- A la fin d'un combat les personnages peuvent gagner un trésor.
- Lorsqu'un trésor est obtenu, le protagoniste peut aléatoirement gagner une armure, un pouvoir ou un sortilège.
- les points de protections de l'armure permettent au protagoniste de protéger leurs points de vie
- Nous utiliserons le thème des WINX dans notre travail avec leurs personnages et leurs icônes
- Tous les personnages ont en début de partie un certain nombre de points de vie
- Les attaques peuvent faire perdre à l'ennemi des points de vie.
- Le jeu peut être mis sur pause et repris plus tard

6 Architecture du projet:

Le projet a été décomposé en différents packages. Chaque package regroupe les classes de même thème. Les classes inter-packages importent les méthodes des autres packages grâce aux " import nompackage.nomclasse".

On a 6 packages:

1. graphique: pour la partie graphique
2. jeu: pour lancer le jeu et le sauvegarder/charger
3. maps: pour avoir la grille (plateau de jeu)
4. protagonistes: pour pouvoir développer les personnages
5. Bataille: pour créer les camps et combats
6. trésor: pour les trésors que gagnent les personnages



7 Etape 1: Les protagonistes entrent en scène

7.1 Les protagonistes:

Personnages:

Il y a deux classes de protagonistes qui s'affrontent, les fées et les sorcières. Ces deux classes héritent (**extends**) d'une seule et même classe Personnage qui présente dans son constructeur les attributs communs aux fées et sorcières.

ex classe Fee:

```java
public class Fee extends Personnage{
    public String Symbole;
    public int nbPouvoirs;
    public int nbArmure=0;
    public int ptprotec;

    public Fee(String nom) {
        super(nom, vie: 30);
        this.nbPouvoirs=1;
        this.ptprotec=0;
        this.nbArmure=0;
    }
```

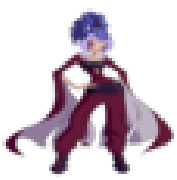
Fées: les fées sont les êtres ayant des pouvoirs magiques propres à chacunes, elles appartiennent à la classe Fée mais présentent chacune des spécificités (nombre d'armures, nombre de pouvoirs).

Une fée est spécifiée par son nom, son pouvoir (arme), son nombre de points de vie et son nombre de points de protection dépendant de son nombre d'armures.

1. Musa 
2. Bloom 
3. Stella 
4. Flora 

Sorcières

Les fées ont des combats contre les sorcières de la classe Sorcière. Chacun.e.s a ses propres sortilèges. Une sorcière ou un sorcier est spécifié par son nom, son nombre de sortilèges , son nombre de points de vie et son nombre de points de protection dépendant de son nombre d'armures.

1. **Icy** 
2. **Stormy** 

Les Fées et Sorcieres sont déclarées sur un plateau grâce à un **symbole**: 'S' pour Sorciere et 'F' pour Fée suivi de la première lettre de leur nom, et ça dans chacune de leurs classe.

ex classe Icy de la sorciere Icy:

```java
public class Icy extends Sorciere{
    public Icy() {
        super(nom: "Icy");
        this.Symbole = "SI";
    }
```

7.2 Le plateau:

Le plateau sur la console:

Le plateau est un échiquier de 64 cases s'affichant sur l'écran. La fonction afficher_map dans la classe Grille permet cela :

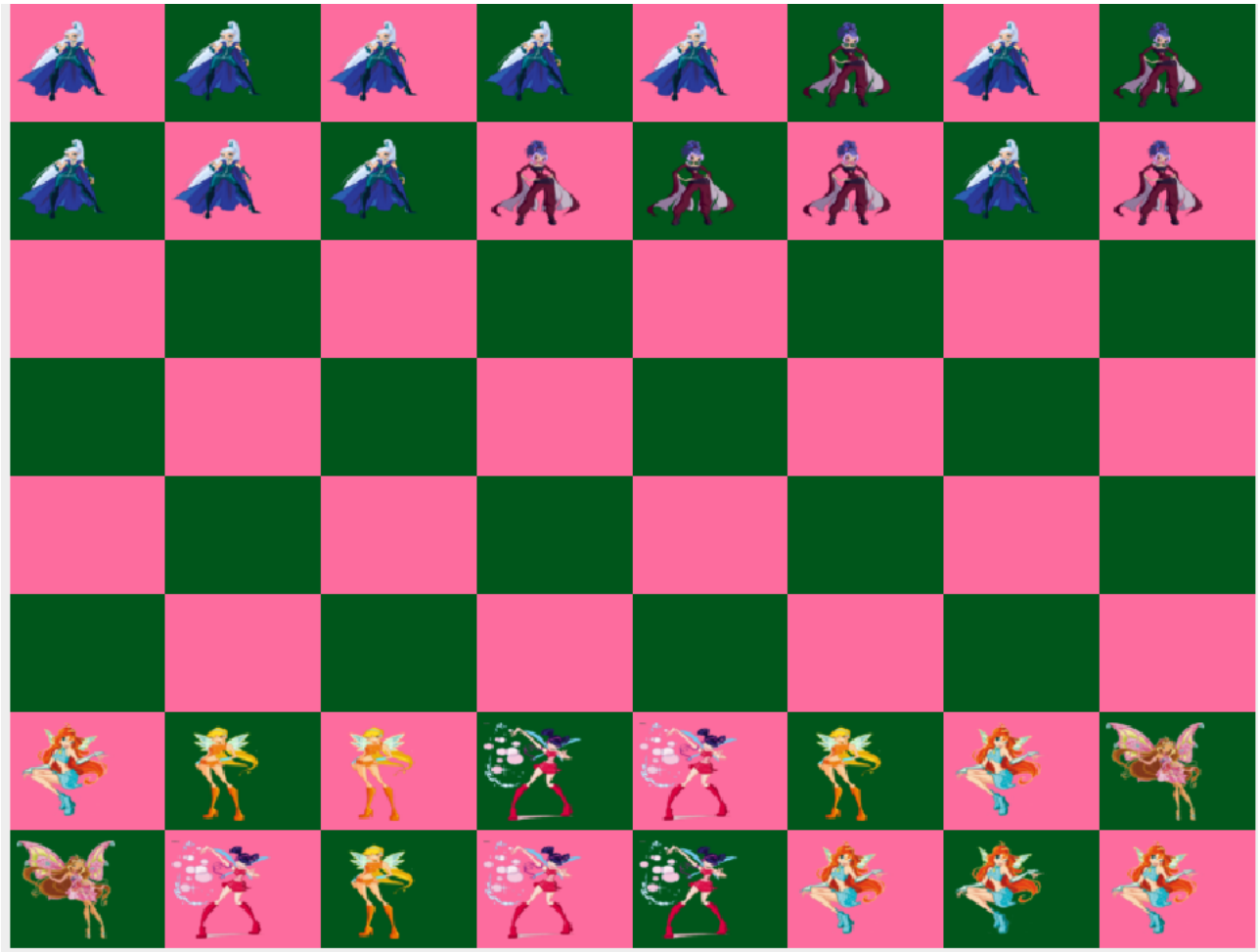
```java
public void afficher_map() {
    String maps = "";
    int i = 0;
    while(i<64) {
        maps += "|";
        maps += map[i].getOccupant().getSymbole();
        i++;
        if (i%8 == 0) {maps += "|\n";}
    }
    System.out.println(maps);
}
```

On place sur cette grille les personnages du jeu avec leurs symboles précisés plus haut grâce au getOccupant().getSymbole(). On obtient au lancement:

```
|SI|SS|SI|SI|SI|SI|SS|SI|
|SI|SS|SI|SI|SS|SS|SS|SS|
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|FM|FF|FM|FM|FM|FS|FF|FS|
|FF|FF|FS|FS|FB|FM|FS|FF|
```

Les deux camps de personnages se font face sur l'échiquier.

La partie graphique du code quant à elle affiche un échiquier rose et vert sur lequel les personnages sont placés de la même façon que sur le plateau de la console. On utilise des boutons pour sélectionner les cases et pour pouvoir y placer des personnages.



8/ Étape 2: Et voilà les combats...

Dans cette partie on crée les combats. Dès qu'un personnage d'une des équipes tente de se positionner sur une case de la grille déjà occupée par un personnage de l'équipe adverse, c'est le début d'un combat.

8.1 Les déplacements:

Les personnages peuvent se déplacer sur tout l'échiquier à condition que la case ne soit pas occupée. Les personnages d'un même camp ne peuvent pas tenter de prendre la position d'un de leur allié. Cela est réalisable grâce à la méthode deplacement dans la classe Grille.

```java
public int deplacement(int i1,int i2) {
    if (map[i1].getOccupant().getSymbole() == "  ") {
        System.out.println("Séléction non correcte !");
        return 2;
    }
    else if (map[i2].getOccupant().getSymbole() == "  ") {
        Case tmp = map[i1];
        map[i1] = map[i2];
        map[i2] = tmp;
        return 0;
    }
    else if (map[i2].getOccupant().getClass().getSuperclass() == map[i1].getOccupant().getClass().getSuperclass()) {
        System.out.println("Déplacement impossible, case occup?e par un coh?quipier !");
        return 3;
    }
    else {
        combat(map[i1],map[i2]);
        return 1;
    }
```

On demande sur la console au joueur ou souhaite t-il se positionner et depuis ou?

Cela grâce à la méthode main de la classe Jeu

```
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|FM|FM|FB|FF|FF|FM|FM|FS|
|FM|FM|FB|FM|FF|FS|FS|FS|

Quel pion voulez-vous bouger ?
60
Ou voulez-vous aller ?
2
```

8.2 Les combats

Les combats se produisent entre les personnages de camps adverse lorsque l'un des personnages tente de prendre la place de son ennemi.

Les méthodes attaquer() et subirAttaque() de la classe Personnage sont lancées.

```
public String attaquer(Personnage p,int degat) {
    String texte=" ";
    texte+="Le Personnage "+getNom()+" attaque "+p.getNom()+" presente sur la case";
    texte+="\n"+p.subirAttaque( this, nb_degats: 10  );
    return texte;
}
```

```
public String subirAttaque(Personnage p,int nb_degats) {
    String texte=" ";
    texte+="La sorciere "+this.getNom()+" est attaquee\n";
    if(NbArmure!=0){
        if(ptprotec>nb_degats){
            ptprotec-=nb_degats;
            texte+=", Heureusement que son armure la protege \n";

        }

        else{
            vie-=(nb_degats-ptprotec);
        }
    }
    else{
        if(vie> nb_degats){
            texte+="Aie \n";
            vie-=nb_degats;
        }
        else{
            this.vie = 0;
            texte += this.die();
            texte += p.win();
        }
    }
        return texte;
    }
```

On affiche à l'écran le nom du personnage qui est attaqué et son agresseur:

```
Le Personnage Flora attaque Stormy presente sur la case
La sorciere Stormy est attaquee
```

9/ Etape 3: Où l'on découvre les trésors...

Les personnages en gagnant un combat face à leur adversaires gagnent un trésor. Ce trésor peut être une armure qui leur octroie des points de protection, un sortilège pour les sorcières ou un nouveau pouvoir pour les fées.

Une énumération générant un trésor aléatoire a été créée pour l'occasion. De cette façon les personnages peuvent gagner un trésor indéfini grâce à la bibliothèque Random de java que l'on importe.

```java
import java.util.Random;

public enum Typetresorr {

    ARMURE, POUVOIR, SORTILEGE;

    public static Typetresorr generateRandomTresor() {
        Typetresorr[] values = Typetresorr.values();
        int length = values.length;
        int randIndex = new Random().nextInt(length);
        return values[randIndex];
    }
```

De là on voit que dans la méthode gagner : win() dans la classe Personnage, le personnage vainqueur de la bataille remporte un trésor grâce à l'appel de la méthode aléatoire de l'énumération

```java
public String win() {
    String msg=" ";
    msg+=this.getNom()+ " remporte le duel \n";
    msg+=this.regeneration();
    msg += this.gagnerTresor();
    return msg;
}
```

10/ Etape 4: Où l'on se range dans les armées...

10.1 Les camps:

Comme détaillé précédemment il y a deux équipes qui s'affrontent celle des Sorcieres et celle des Fées. De ce fait, nous avons créé deux listes avec 16 fées et sorcieres, qui diminue selon le nombre de personnages vaincus.

Une classe camps a été créée pour l'occasion et les méthodes nécessaires y sont définies.

```
public class Camps {
    private List<Sorciere> witch= new ArrayList<>(16);
    private List<Fee> fee= new ArrayList<>(16);

    public void supprimerS(Sorciere w) {
        witch.remove(w);
    }
    public void supprimerF(Fee f) {
        fee.remove(f);
    }

    public int nbSorcieres() {
        return witch.size();
    }

    public int nbFees() {
        return fee.size();
    }
    public String finPartie(){
        String txt=" ";
        if (nbSorcieres()==0 || nbFees()==0){
            txt+="C'est la fin de la partie";
        }
        else{
            txt+="Passez au tour suivant";
        }
        return txt;
    }
```

Les méthodes supprimer() permettent de retirer par 1 le nombre de personnage dans chacune des Arraylist déclarées plus haut, les méthodes nbSorcieres et nbFees servent de compteur à la méthode finPartie() permettant d'arrêter la partie si le nombre de fées ou de sorcières est nul dans l'une des listes.

10.2 Tour:

En début de partie on donne aléatoirement l'équipe qui commencera grâce à une méthode de la classe Jeu utilisant le Random

```
Random tour_debut = new Random();
int choix = tour_debut.nextInt(2);
```

Après ça, l'ordre de jeu offre à chaque camps l'une après l'autre l'opportunité de jouer et on indique au haut de la grille dans quel tour nous sommes.

```
Tour numero 1 c'est au tour de l'equipe des Fees
|SS|SS|SS|SS|SI|SI|SI|SI|
|SS|SI|SI|SI|SS|SI|SI|SS|
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|FM|FM|FB|FF|FF|FM|FM|FS|
|FM|FM|FB|FM|FF|FS|FS|FS|
```

11/ Sauvegarde et Chargement

11.1 Sauvegarde:

Pour gérer le système de sauvegarde on va demander à la fin d'une partie si l'on souhaite la sauvegarder:

```
Voulez-vous stoper le jeu ? (y/n)
y
Voulez-vous sauvegarder le jeu ? (y/n)
y
partie sauvegardée !
```

On appelle alors une fonction dans le package Jeu et la classe Save qui va permettre de rajouter la sauvegarde comprenant une ligne avec les placements des personnages, une ligne avec les points de vie de ceux-ci et une ligne avec leur point d'armure. Ces lignes sont créés via des fonctions de grille pour les traduire en String :

```
/* Dossier de sauvegarde des parties
/* Premiere ligne = map, seconde ligne = point de vie, troisieme ligne = armure
/*
SSSISSSISSSISISISSSSSSSSSISI  SI                          SI                                  FSFMFFFBFMFMFMFBFMFFFMFMFMFMFMFM
20202020202020202020202020200020000000000000000000000000000020000000000000000000000000000030303030303030303030303030303030
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
SSSSSISSSSSISISI  SISSSSSISSSISSFF                          SI                                    FMFMFBFSFBFMFSFSFMFMFMFMFMFMFM
20202020202020200020202020202030000000000000000000000000000020000000000000000000000000000030303030303030303030303030303030
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
SSSISSSISSSISISISSSSSSSSSISI  SI                SI                                          FSFMFFFBFMFMFMFBFMFFFMFMFMFMFMFM
20202020202020202020202020200020000000000000000000020000000000000000000000000000000000030303030303030303030303030303030
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
SSSISSSISISSSSSSSISSSISISSSSSSSI                                                            FFFFFSFBFSFFFMFBFSFMFMFFFSFSFMFB
10202020202020202020202020202020000000000000000000000000000000000000000000000000000000000030303030303030303030303030303030
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
```

```java
public String GetMapString(){
    String sortie = "";
    for (int i =0;i<64;i++){
        sortie += this.map[i].getOccupant().getSymbole();
    }
    return sortie;
}
```

fonction permettant de traduire le placement des protagonistes en chaîne de caractère. on aura les identiques pour la vie et l'armure.

11.2 Chargement:

Le chargement est demandé au début du jeu:

```
Voulez vous charger une partie ? (y/n)
y
Entrer le numéro de sauvegarde
1
Chargement de la partie...
Tour numero 1: c'est au tour de l'equipe des Fees
   a  b  c  d  e  f  g  h
1 |SS|SI|SS|SI|SS|SI|SI|SI|
2 |SS|SS|SS|SS|SI|SI|  |SI|
3 |  |  |  |  |  |  |  |  |
4 |  |  |  |  |  |  |SI|  |
5 |  |  |  |  |  |  |  |  |
6 |  |  |  |  |  |  |  |  |
7 |FS|FM|FF|FB|FM|FM|FM|FB|
8 |FM|FF|FM|FM|FM|FM|FM|FM|
Quel pion voulez-vous bouger ?
```

Cette fonction va lire le fichier save.txt et ressortir les trois lignes de sauvegarde qui vont ensuite être interprétées par une fonction de grille a savoir `public void init_map_saved(String[] map_str)` qui prend en paramètre map_str[0]: le placement, map_str[1]: les points de vie et map_str[2]: les points d'armure et qui crée la grille correspondante.

12/ Gestion de la persistance:

Bilan:

Le projet Java nous aura permis d'apprendre différentes choses:

- L'organisation: Le respect des dates limites, et des contraintes de rendus
- Mieux connaître les IDE Java: nous avons utilisé Visual Studio Code, Eclipse et Netbeans
- Utiliser GitHub: Nous travaillions en groupe sur un projet gitHub commun sur lequel on pouvait pull, push et commit
- Développer ses capacités en programmation orientée objet
- La recherche: en cherchant des bibliothèques et ressources Java en ligne