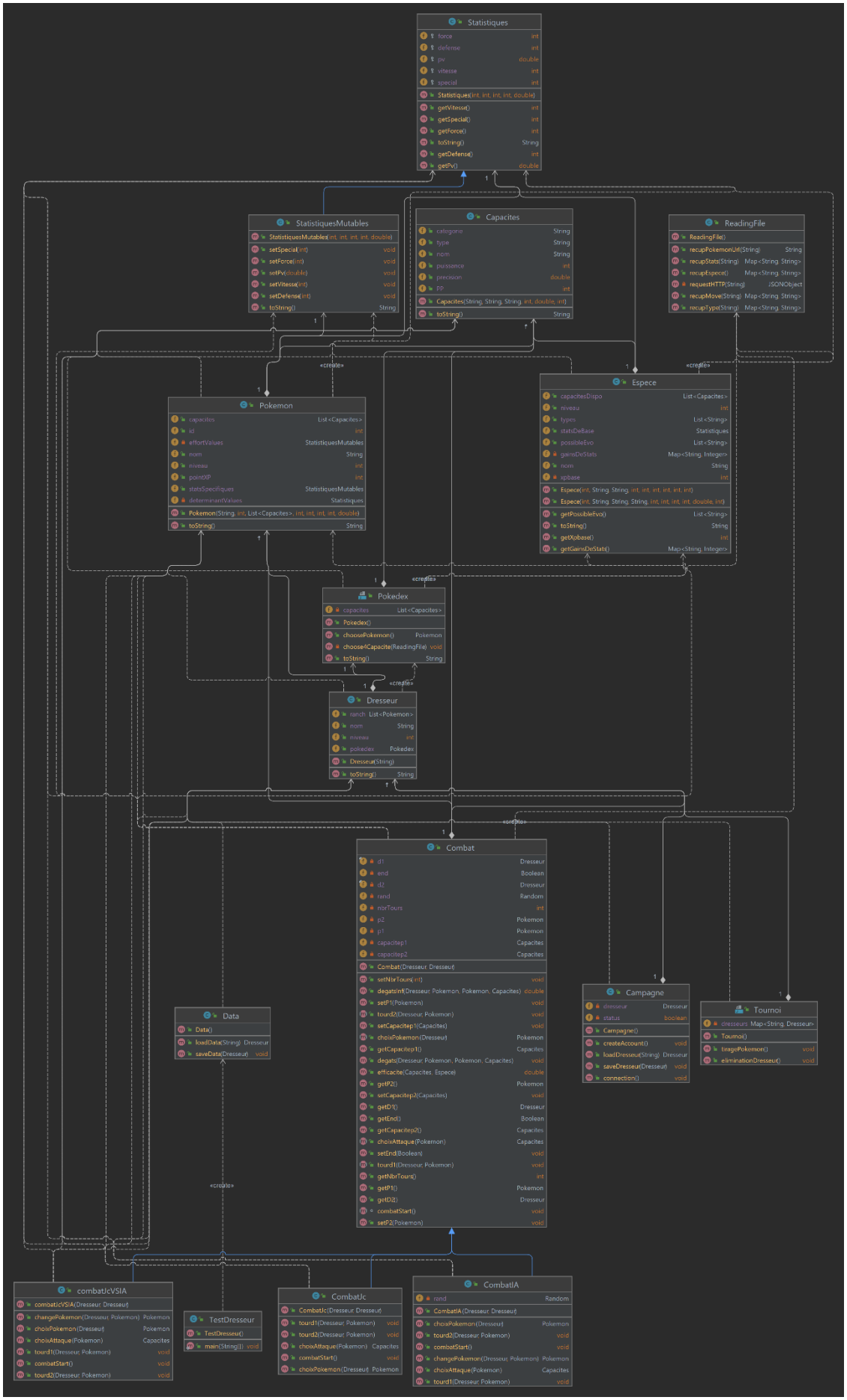
SAE 201 : Développement d'une application

Responsable projet : Donati Leo

Département Informatique
Lacroix Baptiste – Vidal Théo
Semestre 2
Groupe 2
Equipe G2 BL TV

Table des matières


Statistiques
force int
defense int
pv double
vitesse int
special int
Statistiques(int, int, int, int, double)
getVitesse() int
getSpecial() int
getForce() int
toString() String
getDefense() int
getPv() double
StatistiquesMutables
StatistiquesMutables(int, int, int, int, double)
setSpecial(int) void
setForce(int) void
setPv(double) void
setVitesse(int) void
setDefense(int) void
toString() String
Capacites
categorie String
type String
nom String
puissance int
precision double
PP int
Capacites(String, String, String, int, double, int)
toString() String
ReadingFile
ReadingFile()
recupPokemonUrl(String) String
recupStats(String) Map<String, String>
recupEspece() Map<String, String>
requestHTTP(String) JSONObject
recupMove(String) Map<String, String>
recupType(String) Map<String, String>
«create»
Pokemon
capacites List<Capacites>
id int
effortValues StatistiquesMutables
nom String
niveau int
pointXP int
statsSpecifiques StatistiquesMutables
determinantValues Statistiques
Pokemon(String, int, List<Capacites>, int, int, int, int, double)
toString() String
Espece
capacitesDispo List<Capacites>
niveau int
types List<String>
statsDeBase Statistiques
possibleEvo List<String>
gainsDeStats Map<String, Integer>
nom String
xpbase int
Espece(int, String, String, int, int, int, int, int, int)
Espece(int, String, String, String, int, int, int, int, double, int)
getPossibleEvo() List<String>
toString() String
getXpbase() int
getGainsDeStat() Map<String, Integer>
Pokedex
capacites List<Capacites>
Pokedex()
chooisePokemon() Pokemon
choose4Capacite(ReadingFile) void
toString() String
Dresseur
ranch List<Pokemon>
nom String
niveau int
pokedex Pokedex
Dresseur(String)
toString() String
Combat
d1 Dresseur
end Boolean
d2 Dresseur
rand Random
nbrTours int
p2 Pokemon
p1 Pokemon
capacitep1 Capacites
capacitep2 Capacites
Combat(Dresseur, Dresseur)
setNbrTours(int) void
degatsInfl(Dresseur, Pokemon, Pokemon, Capacites) double
setP1(Pokemon) void
tourd2(Dresseur, Pokemon) void
setCapacitep1(Capacites) void
choisPokemon(Dresseur) Pokemon
getCapacitep1() Capacites
degats(Dresseur, Pokemon, Pokemon, Capacites) void
efficacite(Capacites, Espece) double
getP2() Pokemon
setCapacitep2(Capacites) void
getD1() Dresseur
getEnd() Boolean
getCapacitep2() Capacites
choisAttaque(Pokemon) Capacites
setEnd(Boolean) void
tourd1(Dresseur, Pokemon) void
getNbrTours() int
getP1() Pokemon
getD2() Dresseur
combatStart() void
setP2(Pokemon) void
Data
Data()
loadData(String) Dresseur
saveData(Dresseur) void
Campagne
dresseur Dresseur
status boolean
Campagne()
createAccount() void
loadDresseur(String) Dresseur
saveDresseur(Dresseur) void
connection() void
Tournoi
dresseurs Map<String, Dresseur>
Tournoi()
triagePokemon() void
eliminationDresseur() void
combatJcVSIA
combatJcVSIA(Dresseur, Dresseur)
changePokemon(Dresseur, Pokemon) Pokemon
choisPokemon(Dresseur) Pokemon
choisAttaque(Pokemon) Capacites
tourd1(Dresseur, Pokemon) void
combatStart() void
tourd2(Dresseur, Pokemon) void
TestDresseur
TestDresseur()
main(String[]) void
CombatJc
CombatJc(Dresseur, Dresseur)
tourd1(Dresseur, Pokemon) void
tourd2(Dresseur, Pokemon) void
choisAttaque(Pokemon) Capacites
combatStart() void
choisPokemon(Dresseur) Pokemon
CombatIA
rand Random
CombatIA(Dresseur, Dresseur)
choisPokemon(Dresseur) Pokemon
tourd2(Dresseur, Pokemon) void
combatStart() void
changePokemon(Dresseur, Pokemon) Pokemon
choisAttaque(Pokemon) Capacites
tourd1(Dresseur, Pokemon) void

1. Package readingFile

1.1 Classe ReadingFile()

Cette classe va permettre de lire les fichier json en ligne et de renvoyer les données recherchées en fonction de ce que l'on recherche

1.1.1 Méthodes :

```
public Map<String, String> recupMove(String url)
```

Prend en paramètre une chaine de caractère qui correspond à l'url du Pokémon dont on recherche les moves et renvoie un dictionnaire contenant en clef le nom de l'attaque et en valeur une url allant vers ses spécificités

```
public Map<String, String> recupEspece()
```

Récupère toutes les espèces de Pokémons et les renvois dans un dictionnaire ayant comme clef le nom de l'espèce et comme valeur une url allant vers ses spécificités.

```
public Map<String, String> recupStats(String url)
```

Prend en paramètre une chaine de caractère qui correspond à l'url du Pokémon dont on recherche les stats et renvoie un dictionnaire contenant en clef le nom du type de la statistique et en valeur, sa valeur.

```
public Map<String, String> recupType(String url)
```

Prend en paramètre une chaine de caractère qui correspond à l'url du Pokémon dont on recherche les types et renvoie un dictionnaire contenant en clef le type et en valeur une url allant vers ses spécificités

```
public String recupPokemonUrl(String url)
```

Prend en paramètre une chaine de caractère qui correspond à l'url de l'espèce dont on recherche les spécificités du Pokémon et renvoie un dictionnaire contenant en clef le nom du Pokémon et en valeur une url allant vers ses spécificités

```
private JSONObject requestHTTP(String url)
```

Prend en paramètre une chaine de caractère qui correspond à l'url et permet de faire les requêtes nécessaires pour accéder au json. Et renvoie un JsonObject. Cette méthode est utilisée pour éviter la répétition dans les autres méthodes.

1.2 Classe Data()

Cette classe permet de sauvegarder un ou plusieurs dresseurs dans un fichier

1.2.1 Méthodes :

```
public void saveData(Dresseur d)
```

Prend en paramètre l'objet dresseur pour l'enregistrer dans un fichier json

```
public ArrayList<Dresseur> loadData(String nomDresseur)
```

Prend en paramètre le nom du dresseur recherché et renvoie ce dresseur s'il est existant

2. Package pokémon

2.1 Classe Pokemon()

Cette classe permet de créer un Pokemon

2.1.1 Attributs :

```
// Attributs visibles
public int id = 0;
public String nom;
public int niveau;
public int pointXP = 0;
public List<Capacites> capacites;
public StatistiquesMutables statsSpecifiques;

// Attributs non visibles
private StatistiquesMutables effortValues;
private Statistiques determinantValues;
```

Id unique pour chaque Pokémon,
Nom du Pokémon,
Niveau du Pokémon,
Points d'exp du Pokémon,
Liste contenant toutes les capacités du Pokémon
Attribut de type StatistiqueMutables (classes) statsSpecifiques se seront les stats qui pourront être modifié après un combat
Attribut de type StatistiqueMutables (classes) effortValues se seront les stats qui pourront être modifié après un combat
Attribut de type Statistiques (classes) determinantValues se seront les stats qui ne pourront pas être modifié

2.1.2 Constructeur :

```
public Pokemon(String nom, int niveau, List<Capacites> capacites, int
force, int defense, int vitesse, int special, double pv)
```

Constructeur du Pokémon qui se chargera d'initialiser toutes les variables

```
public String toString()
```

Permet l'affichage du Pokémon

2.2 Classe Espece()

Cette classe permet de créer une espèce

2.1.1 Attributs :

```
// Attributs visibles
public String nom;
public List<String> types = new ArrayList<>(2);
public Statistiques statsDeBase;
public int niveau;
public List<Capacites> capacitesDispo;
public List<String> possibleEvo;

// Attributs non visibles
private int xpbase;
private Map<String, Integer> gainsDeStats = new HashMap<>();
```

Nom correspond au nom de l'espèce
types Liste de 2 String (type1, type2) ou (type1, null)
Attribut de type Statistiques (classes) statsDeBase se seront les stats qui ne pourront pas être modifié
Niveau de l'espèce non changeable une fois initialisé
capacitesDispo List contenant toutes les capacités disponibles pour cette espèce
possibleEvo liste contenant le nom de la prochaine espèce disponible
xpbase non changeable une fois initialisé
Dictionnaire contenant en clef la statistique correspondante et en valeur sa valeur non changeable une fois initialisé

2.1.2 Constructeur :

```
public Espece(int idPokemon, String nom, String type1, String type2, int
force, int defense, int vitesse, int special, double pv, int niveau)
```

Sert a initialisé une espèce avec tous ces attributs ce constructeur permet d'initialiser une espèce avec 2 types

```
public Espece(int idPokemon, String nom, String type1, int force, int
defense, int vitesse, int special, int pv, int niveau)
```

Sert a initialisé une espèce avec tous ces attributs ce constructeur permet d'initialiser une espèce avec 1 type

2.1.3 Méthodes :

Getters
Pour récupérer les informations utiles

```
public String toString()
```

Permet l'affichage de l'espèce

2.3 Classe Capacites()

Cette classe permet de créer une capacitée

2.3.1 Attributs :

```
public String nom;
public String type;
public String categorie;
public int puissance;
public double precision;
public int PP;
```

Nom de la capacité,
Nom du type de la capacité
Nom de la catégorie
Valeur de la puissance de la capacité
Valeur de la précision de la capacité
Valeur du PP de la capacité

2.3.1 Constructeur :

```
public Capacites(String nom, String type, String categorie, int puissance,
double precision, int PP)
```

Sert a initialisé une capacite

2.3.2 Méthodes :

```
public String toString()
```

Affiche la capacite

2.4 Classe Statistiques()

2.4.1 Attributs :

```
protected int force;
protected int defense;
protected int vitesse;
protected int special;
protected double pv;
```

Valeur de la force, non changeable
Valeur de la defense, non changeable
Valeur de la vitesse, non changeable
Valeur du special, non changeable
Valeur des pv, non changeable

2.4.2 Constructeur :

```
public Statistiques(int force, int defense, int vitesse, int special,
double pv)
```

Sert à initialiser les stats

2.4.3 Méthodes :

```
public String toString()
```

Permet l'affichage des statistiques

2.5 Classe StatistiquesMutable()

```
public class StatistiquesMutables extends Statistiques
```

Contient les mêmes attributs que Statistiques sauf que cette classe permet de changer les valeurs des attributs

2.5.1 Méthodes :

Grâce aux Setters

```
public String toString()
```

Permet l'affichage des stats

3. Package pokedex

3.1 Classe Pokedex()

Classe permettant d'accéder aux Pokémons et permettant de choisir le Pokémon et les capacités de celui-ci pour le Dresseur

3.1.1 Attributs :

```
private List<Capacites> capacites = new ArrayList<>();
```

Capacites contient la liste des capacités que le dresseur a choisis de mettre à son Pokémon

3.1.2 Méthodes :

```
public Pokemon choosePokemon()
```

Permet de renvoyer un Pokémon choisi aléatoirement parmi toutes les espèces de niveau 1

```
private void choose4Capacite(ReadingFile p)
```

Permet de choisir 4 capacités choisies par l'utilisateur parmi celles disponibles pour l'espèce de Pokémon

```
public String toString()
```

Permet d'afficher le Pokedex

4. Package dresseur

4.1 Classe Dresseur()

Classe permettant de créer un dresseur

4.1.1 Attributs:

```
public String nom;
public int niveau;
public List<Pokemon> ranch = new ArrayList<>();
public Pokedex pokedex = new Pokedex();
```

nom du dresseur
niveau du dresseur
Ranch List contenant tous les Pokémons possédés
Pokedex du Dresseur

4.1.2 Constructeur :

```
public Dresseur(String nom)
```

Permet d'intialiser les attributs + choisis 4 pokemons en appelant pokedex

```
public String toString()
```

Permet d'afficher le Dresseur

5. Package combat

5.1 Classe Combat()

Classe permettant de gérer les combats

5.1.1 Attributs :

```
public abstract class Combat
private Random rand = new Random();
private int nbrTours = 1;
private Boolean end = true;
private final Dresseur d1;
private final Dresseur d2;
private Pokemon p1;
private Pokemon p2;
private Capacites capacitep1;
private Capacites capacitep2;
```

5.1.2 Constructeur :

```
public Combat(Dresseur d1, Dresseur d2)
```

Permet l'initialisation des attributs

5.1.3 Méthode :

```
abstract void combatStart() throws BadAttributeValueExpException;
```

Méthode qui gérera le déroulement des combats

```
public abstract Pokemon choixPokemon(Dresseur d) throws
BadAttributeValueExpException;
```

Méthode qui se chargera du choix du pokémon qui attaquera en premier

```
public abstract void tourd1(Dresseur d, Pokemon p) throws
BadAttributeValueExpException;
```

Méthode qui se chargera de donner les options au dresseur 1 (combat, changement etc…)

```
public abstract void tourd2(Dresseur d, Pokemon p) throws
BadAttributeValueExpException;
```

Méthode qui se chargera de donner les options au dresseur 2 (combat, changement etc…)

```
public abstract Capacites choixAttaque(Pokemon p) throws
BadAttributeValueExpException;
```

Méthode qui se chargera de demander qu'elle attaque le dresseur veut effectuer

```
public void degats(Dresseur d2, Pokemon p1, Pokemon p2, Capacites c1)
```

Méthode qui se charge de savoir si le Pokémon rate son attaque sinon inflige des dégats au Pokémon adverse et vérifie si celui-ci n'est pas ko

```
public double degatsInf(Dresseur d, Pokemon p, Pokemon p2, Capacites c)
```

Méthode qui se charge de calculer les dégâts à infliger au Pokemon.

```
public double efficacite(Capacites c, Espece e)
```

Méthode qui se charge de regarder l'efficacité de l'attaque effectuée

Getters & Setters :

```
public int getNbrTours()
public void setNbrTours(int nbrTours)
public Boolean getEnd()
public void setEnd(Boolean end)
public Dresseur getD1()
public Dresseur getD2()
public Pokemon getP1()
public void setP1(Pokemon p1)
public Pokemon getP2()
public void setP2(Pokemon p2)
public Capacites getCapacitep1()
public void setCapacitep1(Capacites capacitep1)
public Capacites getCapacitep2()
public void setCapacitep2(Capacites capacitep2)
```

5.2 Classe CombatJc()

Classe permettant de gérer un combat de joueurs

```
public class CombatJc extends Combat
```

5.2.1 Constructeur :

```
public CombatJc(Dresseur d1, Dresseur d2) throws
BadAttributeValueExpException
```

5.2.2 Méthodes :

```
@Override
public void combatStart() throws BadAttributeValueExpException

@Override
public Pokemon choixPokemon(Dresseur d) throws
BadAttributeValueExpException

@Override
public void tourd1(Dresseur d, Pokemon p) throws
BadAttributeValueExpException

@Override
public void tourd2(Dresseur d, Pokemon p) throws
BadAttributeValueExpException

@Override
public Capacites choixAttaque(Pokemon p) throws
BadAttributeValueExpException
```

5.3 Classe CombatIA()

Classe permettant de gérer un combat d'IA

```
public class CombatIA extends Combat
```

5.3.1 Attributs :

```
private Random rand = new Random();
```

5.3.2 Constructeur :

```
public CombatIA(Dresseur d1, Dresseur d2) throws
BadAttributeValueExpException
```

5.3.3 Méthodes :

```
@Override
public Pokemon choixPokemon(Dresseur d)

public Pokemon changePokemon(Dresseur d, Pokemon pA)

@Override
public void tourd1(Dresseur d, Pokemon p)

@Override
public void tourd2(Dresseur d, Pokemon p)
@Override
public Capacites choixAttaque(Pokemon p)
```

5.3 Classe CombatJcVSIA()

Classe permettant de gérer un combat entre une IA et un Joueur

```
public class combatJcVSIA extends Combat
```

5.3.1 Attributs :

```
private Random rand = new Random();
```

5.3.2 Constructeur :

```
public CombatIA(Dresseur d1, Dresseur d2) throws
BadAttributeValueExpException
```

5.3.3 Méthodes :

```
@Override
public void combatStart() throws BadAttributeValueExpException
```

Permet de lancer le combat entre les 2 Dresseurs

```
@Override
public Pokemon choixPokemon(Dresseur d)
```

Permet au dresseur de choisir le pokemon qui se battra

```
public Pokemon changePokemon(Dresseur d, Pokemon pA)
```

Permet de changer de Pokemon au cours du combat

```
@Override
public void tourd1(Dresseur d, Pokemon p)
```

Options disponibles pour le dresseur 1 (changement pokemon, attaque etc..)

```
@Override
public void tourd2(Dresseur d, Pokemon p)
```

Options disponibles pour le dresseur 2 (changement pokemon, attaque etc..)

```
@Override
public Capacites choixAttaque(Pokemon p)
```

Permet au dresseur de choisir l'attaque à infliger au pokémon adverse

6. Package modeDeJeu

6.1 Classe Campagne()

```
public class Campagne
```

Classe qui se charge de gérer le mode campagne

6.1.1 Attributs :

```
private boolean status = true;
private Dresseur dresseur;
```

status permet de savoir si le mode campagne est toujours en cours
dresseur correspond au dresseur connecté

6.1.2 Constructeur

```
public Campagne()
```

Permet l'initialisation des valeurs et la gestion de la boucle pour le mode campagne

6.1.3 Méthodes

```
public void connection()
```

La méthode connection se charge de regarder si l'utilisateur existe (s'occupe du mdp et de l'identifiant)

```
public void createAccount()
```

Permet de créer un compte si l'utilisateur n'en a pas

```
public Dresseur loadDresseur(String nom)
```

Permet de charger le Dresseur connecté

```
public void saveDresseur(Dresseur d)
```

Permet de sauvegarder le dresseur avant de quitter le mode campagne

7. Package tournoi

7.1 Classe Tournoi()

```
public class Tournoi
```

Classe qui se charge de s'occuper du déroulement du Tournoi

7.1.1 Attributs

```
private Map<String, Dresseur> dresseurs = new HashMap<>();
```

Dictionnaire ayant en clef le nom du Dresseur et en valeur le Dresseur

7.1.2 Constructeur

```
public Tournoi()
```

Permet d'initialiser les attributs, et se charge de la boucle pour le tournoi

7.1.2 Méthodes

```
public void tiragePokemon()
```

Méthode tirant au hasard pour chaque dresseur des Pokémons de niveau 1

```
public void eliminationDresseur()
```

Méthode se chargeant d'éliminer du tournoi les dresseurs perdants