## Dresseur

+ nom : String
+ deck : Pokemon [3]
- pokemonListe [*]

---

+ <<create>> Dresseur(nom, pokemonDictionary, attaqueDictionary, defenseDictionary)
+ equals() : Boolean
+ afficherDeck()
+ afficherPokemonListe()
+ afficherTousLesPokemons()
+ changerDeck()
+ combatreDresseur(game : Game)
+ combatrePokemonLibre(game : Game)
+ combatreDresseurExistant(game : Game)
+ combatreNouveauDresseur(game: Game)
+ isDeckKO() : Boolean
+ choisirUnPokemon()
+ choisirUneAction( pokemonAdversaire : Pokemon, combat : Combat)
+ capturerPokemon(pokemonLibre : Pokemon)
+ jouerUneCompetence(competenceChoisie : Competence, combat : Combat)

## IA_dresseur

+ pokemonDictionary : Dictionary
+ attaqueDictionary : Dictionary
+ defenseDictionary : Dictionary

---

+ <<create>> Dresseur(nom, pokemonDictionary, attaqueDictionary, defenseDictionary)
+ choisirUnPokemon(pokemon_adversaire : Pokemon)
+ choisirUneAction(pokemonAdversaire : Pokemon, combat : Combat)
+ calculer_CoefMultiplicateur(pokemon : Pokemon,adversaire : Pokemon)
+ jouerUneCompetence( competenceChoisi : Competence , combat : Combat)
+ ai_Ultron_decision( pokemondversaire) : Integer
+ changerPokemon(self, pokemon_adversaire) : Integer

## Pokemon

+ nom : String
+ element : String
+ niveau : Integer
+ vie : Integer
+ vieMax : Integer
+ energie : Integer
+ energieMax : Integer
+ regeneration : Integer
+ resistance : Integer
+ experience : Integer
+ competences : Competence [3..*]

---

+ <<create>> Pokemon( nom : String , pokemonDictionary : Dictionary)

+ toString() : String
+ equals()
+ ShowAllcompetences()
+ isKO() : Boolean
+ reset_VieEnergie()
+ regenererEnergie()
+ retrancherCout(competence : Competence)
+ jouerUneCompetence(dresseurAdversaire: Dresseur)
+ calculer_CoefMultiplicateur(adversaire : Pokemon)
+ retrancherVie(degatsInfilges : Integer)
+ regenererVieEnergie(competenceChoisi : Competence)

## AI

+ nom : String
+ pokemonDictionary : Dictionary
+ attaqueDictionary : Dictionary
+ defenseDictionary : Dictionary

---

+ <<create>> AI( nom : String , pokemonDictionary : Dictionary)

+ jouerUneCompetence(dresseurAdversaire : Dresseur)
+ aiDecision(dresseurAdversaire : Dresseur)

listDresseurs : Dresseur [1..*]

## Game

+ pokemonDictionary : Dictionary
+ attaqueDictionary : Dictionary
+ defenseDictionary : Dictionary
+ utilisateur : Dresseur

---

# <<create>> Game()

+ afficherBienvenue()
+ getNomUtilisateur()
+ afficherMenuPrincipal()
+ initialiterGame()
+ creeNouveauDresseur()
+ main()

## Combat  {abstract}

+ gameOver : Bolean

---

+ <<create>> Competence( nom : string, competencesDictionary: Dictionary)

---

+ reset_VieEnergie_forAll() {abstract}
+ gagnerExperience(dresseurVainqueur : Dresseur) {abstract}
+ swapTour()  {abstract}

## Attaque

+ nom
+ description
+ element
+ puissance : integer
+ precision : integer
+ cout

---

# <<create>> Attaque( nom : string, competencesDictionary: Dictionary)

+ toString() : String
+ equals() : Boolean
+ afficherCompetence() : String
+ isSuccess() : Boolean
+ calculer_degatsInfilges( pokemon: Pokemon, adversaire : Pokemon )

## Defense

+ nom
+ description
+ element
+ soin : integer [2]
+ energie : integer [2]
+ cout : integer

---

# <<create>> Defense( nom : string, competencesDictionary: Dictionary)

+ toString() : String
+ equals()
+ afficherCompetence() : string

## JCJ

+ dresseur1 : Dresseur
+ dresseur2 : Dresseur
+ isDresseur1Turn : Boolean
+ isDresseur2Trun : Boolean

---

# <<create>> JCJ( dresseur1 : Dresseur , dresseur2 : Dresseur)

+ reset_VieEnergie_forAll()
+ gagnerExperience(dresseurVainqueur : Dresseur)
+ toString() : String
+ regenererEnergie_forALL()
+ combatJCJ()

## JCE

+ dresseur : Dresseur
+ pokemonLibre : Pokemon
+ isDresseurTurn : Boolean
+ isPokemonTrun : Boolean

---

+ <<create>> JCE( dresseur : Dresseur , pokemonLibre : Pokemon)

+ reset_VieEnergie_forAll()
+ gagnerExperience(dresseurVainqueur : Dresseur)
+ swapTour()
+ capturerPokemon()
+ reset_VieEnergie_forAll()
+ regenererEnergie_forALL()
+ combatJCE()