POLYTECH NICE-SOPHIA — Université Nice Sophia Antipolis — Membre de UNIVERSITÉ CÔTE D'AZUR

Découvrir la programmation orientée objet avec Java

Travaux pratiques

frederic.rallo@univ-cotedazur.fr

TP5 : types énumérés et algo

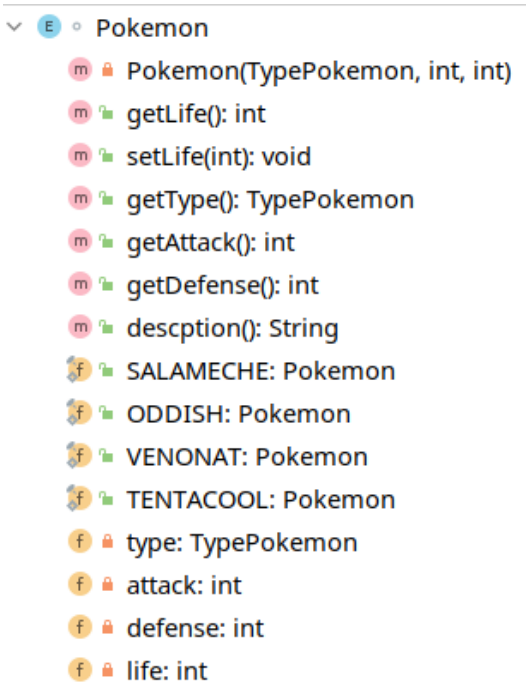
L'objectif des exercices ci-dessous est de mettre en œuvre les types énumérés et de s'exercer au travers d'algorithmes non fournis et de vous initier à l'architecture logiciel.

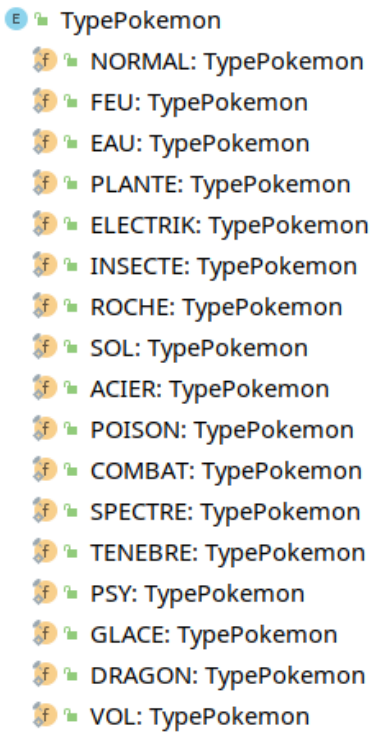
On considère le TP 4 terminé !

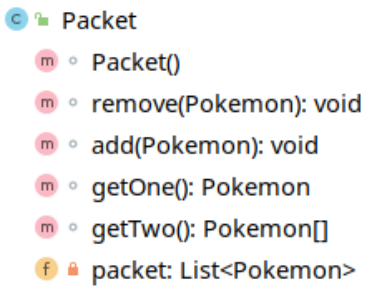
Exercice 1

On désire de créer un jeu de cartes de Pokémon. Chaque personnage ayant
- un nom
- un type (eau, feu...)
- des points de vie
- des points d'attaque
- des points de défense

- Piochez 2 cartes dans le paquet
- Afficher la description des combattants (pt de vie, pt attaque et pt défense)
- Simulez une succession de 10 combats :
  → Aux points de vie de chaque combattant on soustrait les point d'attaque de l'adversaire et on ajoute ses propres points de défense







Découvrir la programmation orientée objet avec Java

Travaux pratiques

frederic.rallo@univ-cotedazur.fr

voici un exemple de trace :

```
* ------------------------------------------------------- *
* TP java                                                 *
*                                                         *
* @author frédéric rallo - frederic.rallo@univ-cotedazur.fr *
* @version TD5 - ex1                                      *
* ------------------------------------------------------- *

This is packet contains:
SALAMECHE
ODDISH
VENONAT
TENTACOOL

ODDISH (life=10, attack=6, defense=2) vs SALAMECHE (life=10, attack=4, defense=5)
   ===> ODDISH (life=8, attack=6, defense=2) and SALAMECHE (life=9, attack=4, defense=5)

VENONAT (life=10, attack=9, defense=1) vs SALAMECHE (life=9, attack=4, defense=5)
   ===> VENONAT (life=7, attack=9, defense=1) and SALAMECHE (life=5, attack=4, defense=5)

SALAMECHE (life=5, attack=4, defense=5) vs VENONAT (life=7, attack=9, defense=1)
   ===> SALAMECHE (life=1, attack=4, defense=5) and VENONAT (life=4, attack=9, defense=1)

ODDISH (life=8, attack=6, defense=2) vs SALAMECHE (life=1, attack=4, defense=5)
SALAMECHE is dead!
   ===> ODDISH (life=6, attack=6, defense=2)

ODDISH (life=6, attack=6, defense=2) vs VENONAT (life=4, attack=9, defense=1)
ODDISH is dead!
VENONAT is dead!
   ===> ODDISH (life=6, attack=6, defense=2)
```

Découvrir la programmation orientée objet avec Java

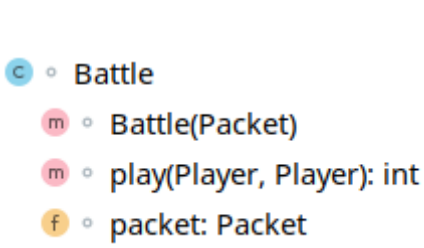
Travaux pratiques

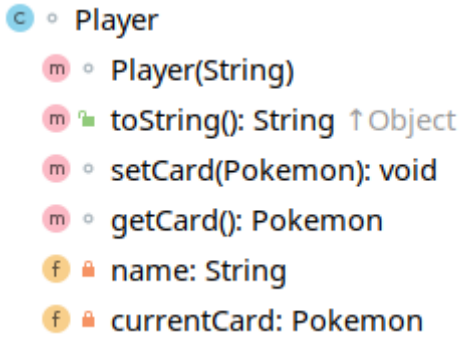
frederic.rallo@univ-cotedazur.fr

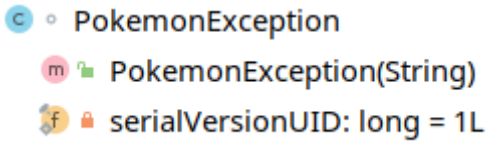
Exercice 2

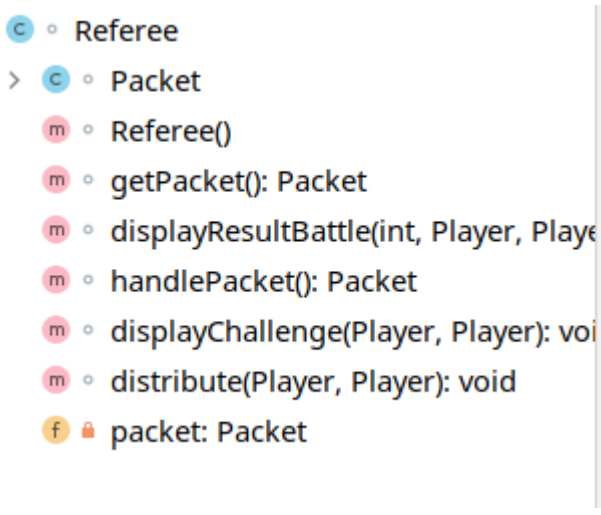
Vous devez améliorer avec les éléments suivants. Trouver une architecture qui convient et modifier le code en respectant les conditions suivantes :
- aucune classe n'est publique
- vous devez disposer d'un classe Player
- vous devez disposer d'un classe Packet
- vous devez disposer d'un classe Referee
- vous devez disposer d'un classe Battle
- vous devez disposer d'une Exception PokemonException
- vous devez disposer d'un enum Pokemon
- La classe Packet est une classe imbriquée dans Referee

Vous attribuerez judicieusement les méthodes suivantes aux classes qui leur correspondent
- play(player1, player2) : les joueurs jouent chacun leur carte et les valeurs de point de vie sont modifiées
- distribute() : distribuer une carte à chaque joueur
- getCard() et setCard() obtenir/modifier la carte d'un joueur
-`displayResultBattle`() : afficher à l'écran le résultat d'une bataille de 2 cartes
- getLife(), setLife btenir/modifier les points de vie d'une carte
- description() : obtenir une String contenant la description d'une carte
-`displayChallenge`() : affiche à l'écran l'affiche des joueurs en vue d'en faire la promotion à des spectacteurs









Découvrir la programmation orientée objet avec Java

Travaux pratiques

frederic.rallo@univ-cotedazur.fr

```
* ------------------------------------------------------- *
* TP java                                                 *
*                                                         *
* @author frédéric rallo - frederic.rallo@univ-cotedazur.fr *
* @version TD5 - ex2                                      *
* ------------------------------------------------------- *

This is packet contains:
SALAMECHE
ODDISH
VENONAT
TENTACOOL

SALAMECHE (life=10, attack=4, defense=5) vs VENONAT (life=10, attack=9, defense=1)
   ===> nobody loose: SALAMECHE (life=6, attack=4, defense=5) and VENONAT (life=7,
attack=9, defense=1)

ODDISH (life=10, attack=6, defense=2) vs SALAMECHE (life=6, attack=4, defense=5)
   ===> nobody loose: ODDISH (life=8, attack=6, defense=2) and SALAMECHE (life=5,
attack=4, defense=5)

ODDISH (life=8, attack=6, defense=2) vs SALAMECHE (life=5, attack=4, defense=5)
   ===> nobody loose: ODDISH (life=6, attack=6, defense=2) and SALAMECHE (life=4,
attack=4, defense=5)

SALAMECHE (life=4, attack=4, defense=5) vs VENONAT (life=7, attack=9, defense=1)
Joueur [fred] lose: -->SALAMECHE is dead!
   ===> winner is: VENONAT (life=4, attack=9, defense=1)

ODDISH (life=6, attack=6, defense=2) vs TENTACOOL (life=10, attack=2, defense=8)
   ===> nobody loose: ODDISH (life=6, attack=6, defense=2) and TENTACOOL (life=12,
attack=2, defense=8)

ODDISH (life=6, attack=6, defense=2) vs VENONAT (life=4, attack=9, defense=1)
Joueur [fred] lose: -->ODDISH is dead!
Joueur [sandrine] lose: -->VENONAT is dead!
   ===> two loosers!
```

Découvrir la programmation orientée objet avec Java

Travaux pratiques

frederic.rallo@univ-cotedazur.fr

Exercice 3

Jouons un peu : ça ne peut pas faire de mal !
On désire modéliser la structure de base d'un jeu de Mastermind qui se joue à 2 joueurs à tour de rôle. Notre objectif à terme consiste à mettre en œuvre l'algorithme de résolution de Knuth.
Le jeu comporte des trous (**peg** en anglais). Le joueur « codeur » choisit une combinaison à faire deviner au joueur « chercheur » et répond aux propositions du joueur « chercheur » chaque tour.

Le joureur « chercheur » doit deviner le code en un nombre de tour limité. A chaque nouveau tour, le joueur « chercheur » fait une nouvelle proposition (**guess**) qui consiste à remplir les trous (pegs) avec une couleur dans chacun. Le joueur « codeur » marque alors le **score** de la proposition avec une paire de valeur (« nb de couleurs bien placées » , « nb de couleurs mal placées »).

Votre mission se limite à créer un objet Mastermind. Mastermind dispose d'un constructeur normal qui prend en paramètre :
- le nombre de trous
- le nombre de couleurs
Le constructeur initialisent les collections guesses et scores

Attention : les couleurs seront symbolisées par des entiers de **1 à n** (zéro étant réservé).

a) Codez le constructeur du Mastermind qui devra créer et remplir une collection de l'ensemble des propositions possibles. Le nombre de trous et de couleurs est passé en paramètre
b) Créez les méthode displayUnusedGuesses() et displayPossibleScore() qui affiche les collections
c) Créez la méthode displayUnusedGuesses()
d) créez la méthode getPossibleScore() qui retournera une collection de toutes les scores (combinaisons de réponses) possibles

REMARQUE :
- lorsqu'il y a plus de couleurs que de trous, le score (0,0) est possible
- le score (x,y) est toujours tel que x+y <= nb de trous

Découvrir la programmation orientée objet avec Java

Travaux pratiques

frederic.rallo@univ-cotedazur.fr

```
* ------------------------------------------------------------ *
* TP java                                                      *
*                                                              *
* @author frédéric rallo - frederic.rallo@univ-cotedazur.fr *
* @version TD5 - ex3                                          *
* ------------------------------------------------------------ *

111
112
113
121
132
113
121
132
113
221
332
113
221
332
113
221
332
113
221
332
113
221
332
113
221
332
113
total: 27 values

(0, 1)
(0, 2)
(0, 3)
(1, 1)
(1, 2)
(2, 1)
total: 6 values
```