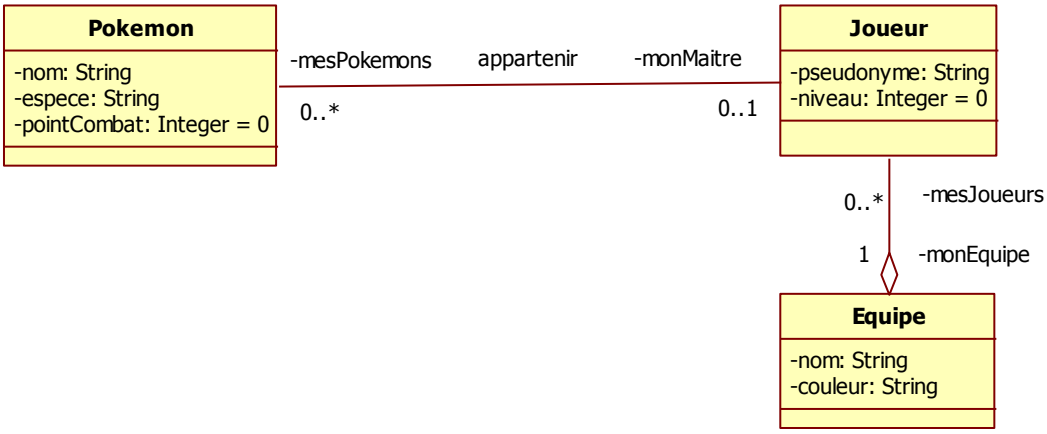
Programmation C++

Parcours GI – ING2

TP2 : Constructeur, destructeur et allocation dynamique

Jeu de Pokémon

Voici un diagramme de classes partiel en UML qui décrit les Pokémons, les joueurs et les équipes.



Étape 1. Implémentez ces classes en C++, en respectant les contraintes suivantes :

1. Un fichier.h et un fichier.cpp pour chaque classe.
2. L'encapsulation des données.
3. Des différents constructeurs avec les attributs obligatoires (not null) et les attributs optionnels (qui peuvent être nuls ou avoir une valeur par défaut au moment de création).
4. Un destructeur pour chaque classe.
5. L'implémentation d'une fonction amie d'affichage pour chaque classe, de type :

```
friend ostream& operator<<(ostream& out, const Pokemon & p);
```

N'oubliez pas d'écrire en parallèle un programme de test qui permet de créer d'abord dans la pile quelques objets de type Pokémon, joueur et équipe; et puis les afficher. Voici quelques données :

Équipe Intuition, couleur Jaune
Équipe Sagesse, couleur Bleu
Équipe Bravoure, couleur Rouge
Joueur Moustache, niveau 5
Pokémon Pikachu, espèce Souris, point de combat 887
Pokémon Rondoudou, espèce Bouboule

Étape 2. Complétez ces classes en ajoutant des contraintes suivantes:

6. Le mapping entre des associations en UML et en C++ (navigabilité, multiplicité, rôle, visibilité, différence entre l'agrégation et la composition):

    a. Un Pokémon peut être sauvage ou capturé par un maître.
    b. Un joueur peut avoir zéro ou plusieurs Pokémons.
    c. Une équipe est composée de plusieurs joueurs.
    d. Un joueur appartient à une et une seule équipe.
7. L'implémentation de toutes les méthodes getter et setter que vous jugez utiles, ainsi que les méthodes qui permettent de :
    a. associer un Pokémon à un joueur.
    b. retirer un Pokémon d'un joueur.
    c. ajouter un joueur dans une équipe.
8. La déclaration des méthodes constantes si elles ne modifient pas les attributs des objets.

Il faut faire attention à l'inclusion multiple des fichiers headers (utiliser des directives de préprocesseur comme #pragma once, #ifndef, …) et l'inclusion croisée (utiliser la déclaration en avance des classes).

Pour le programme de test, associer les différents objets de manière cohérente et utiliser l'allocation dynamique pour créer des objets.