Groupe "Python 2" - Activité 03

Mentor : MILEGNE Dieu donné K. Jules

Comment rendre les divers projets

Contenu du rendu (dans un fichier "**README.md**") :

- Difficultés rencontrées ;
- Comment vous les avez résolues (bref votre approches de solutions).
- Tout envoyé dans un repo distant sur votre compte GitHub

Exercice 1

Un nombre complexe z se présente généralement en coordonnées cartésiennes $z = a + bi$ où a est la partie réelle de z et b sa partie imaginaire (a et b sont deux réels et $i^2 = -1$). Créer une classe Complexe qui permet de décrire des nombres complexes définis sous cette forme.

La classe doit fournir un constructeur convenable pour instancier ces nombres correctement. Elle doit également fournir des méthodes permettant de retourner pour un nombre complexe :

- Sa somme avec un deuxième nombre complexe passé en paramètre.
- Son produit avec un deuxième nombre complexe passé en paramètre.
- Son conjugué.
- Son module.
- Son carré.

Proposer ensuite une méthode permettant de comparer deux nombres complexes

Exercice 2

Une entreprise dispose d'un certain nombre d'employés. Un employé est connu par son nom, son matricule (qui l'identifie de façon unique) et son indice salarial. Le salaire est calculé en multipliant cet indice par une certaine valeur qui peut changer en cas d'augmentation générale des salaires, mais qui est la même pour tous les employés.

Question 1 :

Écrivez la classe des employés avec les informations utiles et des méthodes pour afficher les caractéristiques d'un employé et calculer son salaire.

Question 2 :

Certains employés ont des responsabilités hiérarchiques. Ils ont sous leurs ordres d'autres employés.

- Écrivez une sous-classe des employés qui représente ces responsables en enregistrant leurs inférieurs hiérarchiques directs dans un tableau.
- Écrivez une méthode qui affiche les inférieurs directs (placés directement sous leurs ordres).

Question 3 :

Les commerciaux ont un salaire composé d'un fixe et d'un intéressement proportionnel à leurs ventes.

- Écrivez une sous-classe des commerciaux qui contient l'information sur leurs ventes du dernier mois, une méthode pour mettre à jour cette information et redéfinissez la méthode de calcul de leurs salaires.

Question 4 :

Écrivez une classe représentant tout le personnel de l'entreprise, avec une méthode calculant la somme des salaires à verser.

Exercice 3 : Mini gestion d'une bibliothèque

Vous allez créer un système simple de gestion de bibliothèque. Le système doit permettre de gérer les livres, les membres, et les emprunts de livres.

- **Classe `Livre` :**
  - Attributs :

    `titre : string` (Le titre du livre)

    `auteur : string` (L'auteur du livre)

    `isbn : integer` (Le numéro ISBN du livre)

    `disponible : boolean` (Un booléen indiquant si le livre est disponible à l'emprunt)

  - Méthodes :

    `emprunter() : boolean` (Modifie l'état de disponibilité du livre à False si le livre est disponible)

    `retourner() : boolean` (Modifie l'état de disponibilité du livre à True)

- **Classe `Membre` :**
  - Attributs :

    `nom : string` (Le nom du membre)

    `id_membre : integer` (Un identifiant unique pour le membre)

    `livres_empruntes : list<Livre>` (Une liste des livres empruntés par le membre)

  - Méthodes :

    `emprunter_livre(livre : Livre) : boolean` (Permet au membre d'emprunter un livre s'il est disponible)

    `retourner_livre(livre : Livre) : boolean` (Permet au membre de retourner un livre emprunté)

- **Classe `Bibliotheque` :**
  - Attributs :

    `nom : string` (Le nom de la bibliothèque)

    `livres : list` (Une liste des livres disponibles dans la bibliothèque)

    `membres : list` (Une liste des membres inscrits à la bibliothèque)

  - Méthodes :

    `ajouter_livre(livre : Livre) : void` (Ajoute un nouveau livre à la bibliothèque)

    `inscrire_membre(membre : Membre) : void` (Inscrit un nouveau membre à la bibliothèque)

    `lister_livres_disponibles() : void` (Affiche tous les livres disponibles à l'emprunt)

    `lister_livres_empruntes() : void` (Affiche tous les livres actuellement empruntés)

Proposez une implémentation simple pour ce modèle de gestion de bibliothèque. Créez au moins une bibliothèque, deux (02) membres et trois (03) livres. Puis essayez de manipuler ces objets créés avec les méthodes prédéfinies par leur classe.

NB : - Tous les attributs sont privés ;

- Toutes les méthodes sont publiques ;

- On suppose qu'il n'existe qu'un seul exemplaire pour une chaque livre créé.

Exercice 4

Consignes : Le code doit être lisible, bien indenté et écrit en ***Python***. Les Pokémons sont certes de très mignonnes créatures, mais ils sont également un bon exemple pour illustrer l'héritage.

On se propose de décrire un pokémon par la classe ***Pokemon*** qui contient :

➔ un attribut ***nom*** qui représente le nom du Pokémon.

➔ un attribut ***hp*** (pour Health Points) qui représente les points de vie du Pokémon.

➔ un attribut qui s'appelle ***atk*** qui représente la force de base de l'attaque du Pokémon.

➔ un ***constructeur*** pour instancier des Pokémon adéquatement,

➔ des ***getters*** (accesseurs) qui permettent de consulter les attributs du Pokémon,

➔ une méthode ***isDead()*** qui retourne un booléen pour indiquer si un Pokémon est mort ***(hp == 0)*** ou non.

➔ une méthode ***attaquer(Pokemon p)*** qui permet au Pokémon appelant d'attaquer le Pokémon passé en paramètre. L'attaque éduit ***atk*** points de la vie ***hp*** du Pokémon attaqué ***p***.

➔ une méthode ***afficher()*** qui affiche les informations du Pokémon.

En plus des Pokémon normaux décrits à travers la classe Pokémon, on recense trois types de Pokémon. Les Pokémon de type **Feu**, les Pokémon de type **Eau** et les Pokémon de type **Plante** (en réalité il existe **17 types** en tout mais on ne va pas s'amuser à tous les coder) :

— les **Pokémon de type Feu** sont super efficaces contre les Pokémon de type Plante et leur infligent deux fois plus de dégâts (***2*atk***). Par contre, ils sont très peu efficaces contre les Pokémon de type Eau ou de type Feu et ne leur infligent que la moitié des dégâts (***0.5*atk***). Ils infligent des dégâts normaux aux Pokémon de type Normal. Ils sont décrits par la classe ***PokemonFeu***.

— les **Pokémon de type Eau** sont super efficaces contre les Pokémon de type Feu et leur infligent deux fois plus de dégâts (***2*atk***). Par contre, ils sont très peu efficaces contre les Pokémon de type Eau ou de type Plante et ne leur infligent que la moitié des dégâts (***0.5*atk***). Ils infligent des dégâts normaux aux Pokémon de type Normal. Ils sont décrits par la classe **PokemonEau**.

— enfin, les **Pokémon de type Plante** sont super efficaces contre les Pokémon de type Eau et leur infligent deux fois plus de dégâts (***2*atk***). Par contre, ils sont très peu efficaces contre les Pokémon de type Plante ou de type Feu et ne leur infligent que la moitié des dégâts (***0.5*atk***). Ils infligent des dégâts normaux aux Pokémon de type Normal. Ils sont décrits par la classe **PokemonPlante**.

Travail à faire

1. Ecrire le code de la classe **Pokemon**
2. Écrire les classes **PokemonFeu**, **PokemonEau** et **PokemonPlante** en se servant des méthodes suivantes :
   - **getClasse()** : retourne la classe d'un objet,
   - **getName()** : retourne le nom complet (y compris le nom du package) d'une classe,
   - **getSimpleName()** : retourne le nom simple (sans le package) d'une classe.
3. En se servant du polymorphisme et de la surcharge, proposer une version améliorée du code de chacune des classes **PokemonFeu**, **PokemonEau** et **PokemonPlante**.

Thomas Stearns Eliot, une fois a dit : *"Seulement ceux qui prendront le risque d'aller loin découvriront jusqu'où on peut aller."*