TD3 - Abstraction & Gestion des exceptions

Imene Kerboua

2022/2023

1 Rappels

1.1 Abstraction

```python
from abc import ABC

class Personne(ABC):
    def marcher():
        pass

    @abstractmethod # oblige l'implementation de manger() dans la classe fille
    def manger():
        pass

class Etudiant(Personne):
    def manger(aliment: str):
        print(f"Je mange {aliment}")
```

1.2 Gestion des exceptions

Déclencher une exception:

```python
... # rest of code, class Circle declaration
def set_radius(self, value: float):
    if value <= 0:
        raise Exception("Please enter a positive value")
    else self.radius = value
```

Gérer une exception avec les clauses `try`, `except`, `finally` :

```python
...# rest of code, Circle object instanciation
try:
    circle.set_radius(value=-1)
except Exception as e:
    print(e)
finally:
    print("End of script")

# out => Please enter a positive value
# out => End of script
```

Créer sa propre exception en héritant de la classe `Exception` :

```python
class ValueError(Exception):
    def __init__(self, message):
```

```
        self.message = message
        # or super().__init__(message)

    def __str__(self):
         return f"{self.message}"

def set_radius(self, value: int):
    if value <= 0:
        raise ValueError("Please enter a positive value")
    else self.radius = value
```

2 Application

Exercice 1: Classe abstraite

- Créer une classe abstraite `Shape`. Ajouter une méthode `get_area()`, faire en sorte de forcer l'implémentation de cette méthode à l'implémentation de cette classe abstraite. La classe `Shape` est-elle instanciable ?
- Créer une classe `Circle` qui implémente la classe `Shape`. De même pour la classe `Square`.
- Donnez des exemples d'exécution dans `"__main__"`.

Exercice 2 : Gestion des exceptions

Reprenez l'exercice sur les `Pokemon` (Exercice 4 TD2) pour y ajouter une exception lorsque l'attaque est menée sur un `Pokemon` avec `un hp = 0`. L'exception devra retourner le message suivant : "Impossible d'attaquer un Pokémon déjà dead x_x !". pensez à gérer l'exception à l'appel de la méthode `attaquer()` dans votre code.

Exercice 3:

L'objectif est de définir une classe abstraite destinée à gérer un tableau trié d'éléments et comportant une méthode abstraite `plus_grand(self, a, b)`. Cette méthode devra comparer deux éléments. Pour gérer un tableau trié d'objets d'un certain type, il faudra étendre la classe abstraite en une classe définissant la méthode `plus_grand(self, a, b)` pour le type d'objets en question. On construira :

1. Une classe abstraite `TableauTrieAbstrait` gérant un tableau d'éléments qui reste toujours trié par ordre croissant par rapport à la relation définie par une méthode abstraite `plus_grand()`. Cette classe devra contenir au moins :
   - Une méthode abstraite `plus_grand(self, a, b)`.
   - Une méthode `inserer(self, element)` pour insérer une instance d'élément dans le tableau (il faut faire attention à l'ordre pour ne pas l'insérer n'importe où).
2. Une classe `TableauTrieEntiers` qui étend la classe `TableauTrieAbstrait` ; cette classe est destinée à gérer un tableau trié d'entier. Il faut essentiellement y définir la méthode `plus_grand(self, a, b)` pour des entiers (**PS:** Ne pas utiliser `append()` et `sort()` pour insérer l'entier, mais plutôt faire en sorte de l'insérer avec la méthode `insert(position, element)` à la bonne position. Il faut aussi accepter les doublons et les insérer dans le tableau).
3. Une classe, `TableauTrieChaines` qui étend la classe `TableauTrieAbstrait` ; cette classe est destinée à gérer un tableau trié de chaîne de caractères. Il faut essentiellement y définir la méthode `plus_grand(self, a, b)` en se basant sur le nombre de caractères (**PS:** Idem que pour la question précédente).
4. Ajouter une variable de classe `TAILLE_MAX` aux deux classes précédentes, qui permet de définir la taille maximale du tableau. Initialiser la valeur de `TAILLE_MAX` à `5`.

5. Ajouter aux deux classes précédentes une méthode de classe `modifier_taille_max(nouvelle_taille_max)` qui permet de modifier la taille maximale que peut avoir un tableau. Faire en sorte que cette méthode soit obligatoire à implémenter par les classes qui héritent de `TableauTrieAbstrait`.

6. Déclencher une exception à l'insertion d'un élément qui existe déjà dans le tableau et une autre lorsque la taille maximale du tableau est atteinte.

7. Proposez une amélioration du code précédemment créé.