OPENCLASSROOMS

Corrigé Activité P1C3

Le corrigé ci-dessous illustre l'utilisation du constructeur (__init__) pour l'initialisation et l'application de la convention d'encapsulation (via _id_interne).

Python

```
// 1. Définition de la Classe Player

class Player:

// 2. Implémentation du Constructeur (init)

// S'assure que les données de base sont valides dès la création

        def __init__(self, nom_joueur: str, identifiant_unique: int):

        self.nom = nom_joueur

// Utilisation de la convention _ pour marquer l'attribut comme privé/interne

        self._id_interne = identifiant_unique

            print(f"Nouveau joueur créé : {self.nom} (ID interne:
        {self._id_interne})")

// 3. Implémentation du Getter pour l'Encapsulation (lecture contrôlée)

// On expose l'identifiant uniquement via cette méthode publique

        def get_id_interne(self):

// Permet de consulter la valeur de l'attribut privé

        return self._id_interne

// 4. Implémentation optionnelle d'un Setter pour le Nom (changement
contrôlé)
```

```
        def set_nom(self, nouveau_nom: str):
// Ici, vous pourriez ajouter une vérification, par exemple :
        if len(nouveau_nom) > 2:
                self.nom = nouveau_nom
                print(f"Nom mis à jour pour {nouveau_nom}.")
                else:
                print("Erreur: Le nom du joueur est trop court.")
// 5. Implémentation de la méthode __repr__ pour le débogage
        def __repr__(self):
// Retourne une chaîne de caractères claire représentant l'objet
        return f"Player(nom='{self.nom}', id_interne={self.get_id_interne()})"
// Démonstration et Instanciation (Création d'objets)
// L'appel du constructeur est la première étape du cycle de vie de l'objet.
        joueur_alice = Player("Alice", 101)
        joueur_bob = Player("Bob", 102)
// Lecture des données via la méthode publique (Getter)
        print(f"ID d'Alice: {joueur_alice.get_id_interne()}")
// Tentative d'accès et de modification directe (déconseillée par la convention d'encapsulation)
// joueur_alice._id_interne = 999  // Conventionnellement, on évite cela
// Modification contrôlée via le Setter
        joueur_bob.set_nom("Robert")
        joueur_bob.set_nom("Bo") // Échoue à cause de la validation dans le setter
// Utilisation de __repr__ pour la visualisation (utile pour le débogage)
```

```
print(repr(joueur_alice))
```

Ce corrigé montre comment l'utilisation du constructeur (__init__) garantit la validité de l'état initial et comment l'application de la convention d'encapsulation (_id_interne et get_id_interne) **protège les données internes** de modifications externes inattendues. L'implémentation de __repr__ permet, quant à elle, de rendre l'objet facilement lisible dans l'environnement de développement.