Module : POO Python (M103)
TP N° 3

Année de Formation 2023/2024

Filière : Développement digital Groupe : DEV 101 - DEV 102 Niveau : 1ère année

Exercice 1 :

1. Créez une classe de base `**Vehicule**` avec les attributs suivants :

- `**immatriculation**` (l'immatriculation du véhicule)
- `**capacite**` (la capacité du véhicule en termes de poids ou de volume)
- `**vitesse_max**` (la vitesse maximale du véhicule en km/h)

2. Créez des classes dérivées pour différents types de véhicules, par exemple, `**Camion**`, `**Fourgon**`, `**Moto**`, etc. Chaque classe dérivée devrait hériter de la classe `**Vehicule**` et peut avoir des attributs spécifiques, tels que `**charge_max**` pour les camions, `**volume_max**` pour les fourgons, et `**cylindree**` pour les motos.

3. Ajoutez des méthodes pour chaque classe dérivée pour représenter des comportements spécifiques. Par exemple, un `Camion` pourrait avoir une méthode `**charger_marchandises()**`, un `**Fourgon**` pourrait avoir une méthode `**charger_colis()**`, et une `**Moto**` pourrait avoir une méthode `**livrer_rapidement()**`.

4. Créez une classe `**MissionTransport**` avec les attributs suivants :

- `**depart**` (le lieu de départ de la mission)
- `**arrivee**` (le lieu d'arrivée de la mission)
- `**date**` (la date de la mission)
- `**vehicule**` (le véhicule assigné à la mission)

5. Implémentez une méthode `**assigner_vehicule()**` dans la classe `**MissionTransport**` pour permettre d'assigner un véhicule à la mission.

6. Créez des instances de différents types de véhicules et de missions de transport, puis assignez un véhicule à chaque mission.

7. Appelez les méthodes spécifiques à chaque type de véhicule pour simuler les opérations de chargement, de transport et de livraison dans le cadre des missions de transport.

Exercice 2 :

1. Créez une classe de base `**Produit**` avec les attributs suivants :

- `**nom**` (le nom du produit)

- `**prix**` (le prix du produit)

2. Créez des classes dérivées pour différents types de produits, par exemple, `**Livre**`, `**Vêtement**`, `**Électronique**`, etc. Chaque classe dérivée devrait hériter de la classe `**Produit**` et peut avoir des attributs spécifiques, tels que `**auteur**` pour les livres, `taille` pour les vêtements, et `marque` pour les produits électroniques.

3. Ajoutez des méthodes pour chaque classe dérivée pour représenter des fonctionnalités spécifiques. Par exemple, un `Livre` pourrait avoir une méthode `**afficher_auteur()**`, un `Vêtement` pourrait avoir une méthode `**afficher_taille()**`, et un `Électronique` pourrait avoir une méthode `**afficher_marque()**`.

4. Créez une classe `**Client**` avec les attributs suivants :

- `**nom**` (le nom du client)

- `**adresse**` (l'adresse du client)

- `**panier**` (une liste des produits que le client a ajoutés à son panier)

5. Implémentez une méthode `**ajouter_au_panier()**` dans la classe `Client` pour permettre au client d'ajouter des produits à son panier.

6. Implémentez une méthode `**afficher_panier()**` dans la classe `**Client**` pour afficher les détails de tous les produits dans le panier du client, en appelant les méthodes spécifiques à chaque type de produit.

7. Créez des instances de différents produits et clients, puis ajoutez les produits au panier des clients.

8. Appelez la méthode `**afficher_panier()**` pour afficher les détails de tous les produits dans le panier d'un client.

Exercice 3 :

1. Créez une classe de base `**Personnage**` avec les attributs suivants :

- `**nom**` (le nom du personnage)
- `**niveau**` (le niveau du personnage)
- `**points_de_vie**` (les points de vie du personnage)
- `**points_d'attaque**` (les points d'attaque du personnage)

2. Créez des classes dérivées pour différents types de personnages, par exemple, `**Guerrier**`, `**Mage**`, `**Archer**`, etc. Chaque classe dérivée devrait hériter de la classe `**Personnage**` et peut avoir des attributs spécifiques, tels que `magie` pour les mages, `**force**` pour les guerriers, et `**agilite**` pour les archers.

3. Ajoutez des méthodes pour chaque classe dérivée pour représenter des compétences spécifiques. Par exemple, un `Guerrier` pourrait avoir une méthode `**frappe_lourde()**`, un `**Mage**` pourrait avoir une méthode `**lancer_sort()**`, et un `**Archer**` pourrait avoir une méthode `**tir_precision()**`.

4. Créez une classe `Arme` avec les attributs suivants :

- `**nom**` (le nom de l'arme)
- `**degats**` (les dégâts que l'arme inflige)

5. Créez une classe `**Inventaire**` qui contient une liste d'armes.

6. Implémentez une méthode `**ajouter_arme()**` dans la classe `Inventaire` pour ajouter des armes à l'inventaire.

7. Implémentez une méthode `**afficher_inventaire()**` dans la classe `Inventaire` qui affiche les détails de toutes les armes dans l'inventaire.

8. Créez des instances de différents personnages, d'armes et d'inventaires, puis ajoutez des armes à l'inventaire des personnages.

9. Appelez la méthode `**afficher_inventaire()**` pour afficher les détails de toutes les armes dans l'inventaire d'un personnage.

Exercice 4 :

1. Créez une classe de base `**Paiement**` avec les attributs suivants :

- `**montant**` (le montant du paiement)

- `**date**` (la date du paiement)

2. Créez des classes dérivées pour différents types de paiements, par exemple, `**CarteDeCredit**`, `**PayPal**`, `**ViremmentBancaire**`, etc. Chaque classe dérivée devrait hériter de la classe `Paiement` et peut avoir des attributs spécifiques, tels que `**numero_de_carte**` pour les cartes de crédit, `**adresse_email**` pour PayPal, et `**compte_bancaire**` pour les virements bancaires.

3. Ajoutez des méthodes pour chaque classe dérivée pour représenter des comportements spécifiques. Par exemple, une `**CarteDeCredit**` pourrait avoir une méthode `**effectuer_paiement()**`, un `**PayPal**` pourrait avoir une méthode `**connexion_paypal()**`, et un `**ViremmentBancaire**` pourrait avoir une méthode `**initier_virement()**`.

4. Créez une classe `**Client**` avec les attributs suivants :

- `**nom**` (le nom du client)

- `**adresse**` (l'adresse du client)

- `**moyen_de_paiement**` (le moyen de paiement utilisé par le client)

5. Implémentez une méthode `**effectuer_paiement()**` dans la classe `**Client**` pour permettre au client d'effectuer un paiement en utilisant son moyen de paiement.

6. Créez des instances de différents types de paiements (carte de crédit, PayPal, virement bancaire) et de clients, puis effectuez des paiements en utilisant les moyens de paiement associés.

7. Appelez les méthodes spécifiques à chaque type de paiement pour simuler les transactions.