Programmation orientée objet (UAA14)

Exercices : série 2

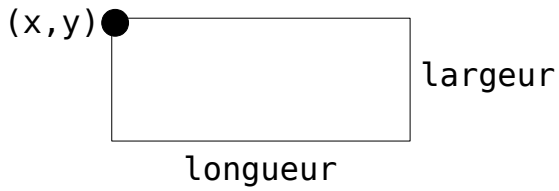
Objets d'apprentissage :

- ✔ Modéliser une logique de programmation orientée objet.
- ✔ Déclarer une classe.
- ✔ Instancier une classe.
- ✔ Utiliser les méthodes de l'objet instancié.
- ✔ Traduire un algorithme dans un langage de programmation.
- ✔ Commenter des lignes de code.
- ✔ Tester le programme conçu.
- ✔ Caractériser les attributs dans une classe (encapsulation).
- ✔ Caractériser les méthodes dans une classe (encapsulation).
- ✔ Décrire la création d'un constructeur.
- ✔ Extraire d'un cahier des charges les informations nécessaires à la programmation.
- ✔ Programmer en recourant aux instructions et types de données nécessaires au développement d'une application.
- ✔ Corriger un programme défaillant.
- ✔ Développer une classe sur la base d'un cahier des charges en respectant le paradigme de la programmation orientée objet.
- ✔ Programmer en recourant aux classes nécessaires au développement d'une application orientée objet.
- ✔ Améliorer un programme pour répondre à un besoin défini.

Exercice 1 : rectangle

Etape 1

Crée une classe intitulée `Rectangle` qui doit permettre de créer des rectangles. Un rectangle est caractérisé par des coordonnées x et y (la position du coin supérieur gauche), une largeur et une longueur.



La classe `Rectangle` contient les variables d'instances suivantes (toute de type `integer`) :

- `x` : coordonnée X du point supérieur gauche
- `y` : coordonnée Y du point supérieur gauche
- `largeur` : la largeur du rectangle
- `longueur` : la longueur du rectangle

Etape 2

Crée un constructeur pour la classe `Rectangle` (constructeur à 4 arguments).

Etape 3

Fais en sorte que les v.i `largeur` et `longueur` puissent être accédées en lecture et en écriture depuis l'extérieur de la classe.

Ajoute également les méthodes suivantes à la classe `Rectangle` :

- `périmètre` : retourne le périmètre (calculé) d'un rectangle
- `surface` : retourne la surface (calculée) du rectangle
- `déplacer` : modifie l'emplacement du coin supérieur gauche en une nouvelle position `(x,y)` sur base de deux entiers reçus en argument

Etape 4

Dans la sandbox, effectue les opérations suivantes (dans l'ordre) :

1. crée un objet de type `Rectangle` avec les propriétés de ton choix
2. affiche à l'écran les coordonnées X et Y du coin supérieur gauche du rectangle créé ainsi que la largeur et la longueur de ce rectangle.
3. modifie la longueur et la largeur de ce rectangle
4. affiche à nouveau à l'écran la largeur et la longueur du rectangle
5. crée d'autres rectangles avec d'autres valeurs et fais appel aux différentes méthodes de la classe

Exercice 2 : entrainement

Etape 1

Crée une classe nommée `Entrainement` permettant de créer des entrainements de course à pied.

Un entrainement est caractérisé par :

- le nom du coureur ;
- la longueur (en mètres) du parcours course à pied ;
- le temps (en secondes) mis par le coureur pour le parcourir.

Crée les variables d'instance nécessaires avec les types adéquats.

Etape 2

Crée un constructeur pour la classe `Entrainement`.

Etape 3

Ajoute les méthodes suivantes :

- `vitesseMoyenne` : retourne la vitesse moyenne (en km/h) du coureur (attention, une valeur réelle et pas entière !)
- `comparerTemps` : reçoit un entier en argument représentant le temps en secondes mis lors d'un autre entrainement et retourne la différence entre ce temps et le temps mis par le coureur lors de l'entrainement courant

Etape 4

Dans la sandbox, effectue les opérations suivantes :

1. crée un objet de type `Entrainement`
2. affiche à l'écran le nom du coureur réalisant cet entrainement
3. affiche à l'écran (en faisant appel aux méthodes adéquates) la vitesse moyenne lors de l'entrainement et la comparaison entre le temps de cet entrainement et le temps d'un autre (de ton choix)

Exercice 3 : placement

Etape 1

Crée la classe nommée `Placement` permettant de gérer des capitaux placés pendant un certains temps dans une banque.

Un placement est caractérisé par :

- un nombre total d'années durant lequel le capital est placé (ex. 10 ans)
- un taux d'intérêts (ex. 2,5)
- un montant placé

Etape 2

Prévois un constructeur permettant de créer un placement.

Etape 3

Ajoute les méthodes suivantes :

- `déterminerCapitalPartiel` : reçoit en argument un nombre d'années et calcule la valeur du capital après ce nombre d'années.
  La formule à employer est : $montant * (1 + taux/100)^{durée}$
- `déterminerCapitalFinal` : calcule la valeur finale du montant placé (utilise la méthode précédente !)
- `déterminerCapitalAcquis` : calcule le montant que le client va percevoir sachant que l'état perçoit un certain pourcentage (reçu en argument)

Etape 4

Dans la sandbox PHP :

- construire un objet de type `Placement` : le montant de départ est de 1000€ avec une durée de 10 ans et un taux de 2,5%
- afficher à l'écran le capital partiel après 3 ans, le capital final et le montant perçu de ce placement (l'état perçoit 10% de la somme)

Exercice 4 : véhicules

Etape 1

Crée une classe nommée `Véhicule` permettant de créer des véhicules. Un véhicule est caractérisé par :

- une plaque d'immatriculation (peut contenir des chiffres, des lettres et des tirets) ;
- une marque ;
- un modèle ;
- une capacité maximale du réservoir ;
- un niveau de jauge de carburant (nombre de litres de carburant réellement dans le réservoir).

Etape 2

Crée un constructeur pour la classe `Véhicule`.

Etape 3

Adapte le constructeur afin de faciliter la création de véhicules avec un réservoir plein. Autrement dis, il n'est pas nécessaire de préciser le niveau de la jauge de carburant lors de la création d'un véhicule : ce niveau est automatiquement égal à la capacité maximale du réservoir.

Etape 4

Ajoute les méthodes suivantes :

- `typeVéhicule` : retourne une chaine de caractère contenant la marque et le modèle du véhicule (séparés par un espace)
- `ajouterCarburant` : reçoit en argument une quantité (en litres) de carburant et rajoute cette quantité dans le réservoir (attention à ne pas déborder !)
- `état` : retourne la chaine de caractère suivante :
  « `<type>` (`<jauge>` / `<capacité max>` litres) » (ex. « Opel Adam (20/30 litres) »)

Etape 5

Adapte la méthode `ajouterCarburant` afin que la quantité soit facultative : lorsqu'aucune quantité n'est précisée, le réservoir est rempli au maximum.

Etape 6

Dans la sandbox, réalise les opérations suivantes :

1. crée plusieurs véhicules de ton choix (en utilisant également les valeurs par défaut)
2. affiche l'état de tous les véhicules créés (en utilisant la méthode `état`)
3. teste la méthode `ajouterCarburant` en l'appelant de différente manière (avec / sans la valeur par défaut)
4. ré-affiche l'état de tous les véhicules et vérifie le bon fonctionnement de la méthodes `ajouterCarburant`

Exercice 5 : commande

Etape 1

Crée une classe `Commande` permettant de gérer des commandes d'articles effectués par des clients.

Par facilité, une commande ne peut contenir qu'un seul article mais ce dernier peut être commandé en plusieurs exemplaires (ex. Une commande comporte 3 exemplaires du livre « Le PHP pour les nuls »).

Une commande est caractérisée par :

- le nom du client qui passe la commande ;
- le libellé de l'article commandé ;
- le nombre d'exemplaires de l'article commandé ;
- le prix unitaire (en €) de l'article commandé.

Etape 2

Crée un constructeur permettant de créer une commande. Fais en sorte que le constructeur permette de créer facilement des commandes avec un seul exemplaire.

Etape 3

Ajoute les méthodes suivantes :

- `modifierQuantité` : permet de modifier la quantité commandée
- `coûtTotal` : calcule le coût total de la commande
- `imprimerBonCommande` : affiche la commande sous la forme suivante

```
BON DE COMMANDE DU CLIENT <nom>

Article : <libellé>
Quantité commandée : <quantité>
Prix unitaire : <prix> euros

COUT TOTAL : <total> EUROS
```

Etape 4

Crée des commandes et affiche le bon de commande pour chacune d'elles.

Références

Les présents exercices ont été élaborés à l'aide des ressources suivantes :

- https://www.pierre-giraud.com/php-mysql-apprendre-coder-cours/introduction-programmation-orientee-objet/
- Cours de « Développement d'applications WEB », Hénallux (2019)