Programmation orientée objet (UAA14)

Exercices : série 5

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 : points et rectangles

On souhaite mettre en place un programme permettant de créer des rectangles à partir de points placés sur un plan cartésien.

Etape 1

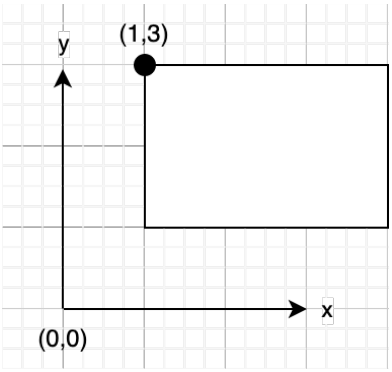
Crée la classe `Point` permettant de créer des objets représentant des points contenant les variables d'instance suivantes :

- `coordX` : la position en x du point
- `coordY` : la position en y du point

Crée un constructeur pour cette classe permettant de garnir toutes les variables d'instance.

Réécris la méthode `__toString` de la classe pour qu'un point soit représenté par ses coordonnées `(x,y)`.

Etape 2

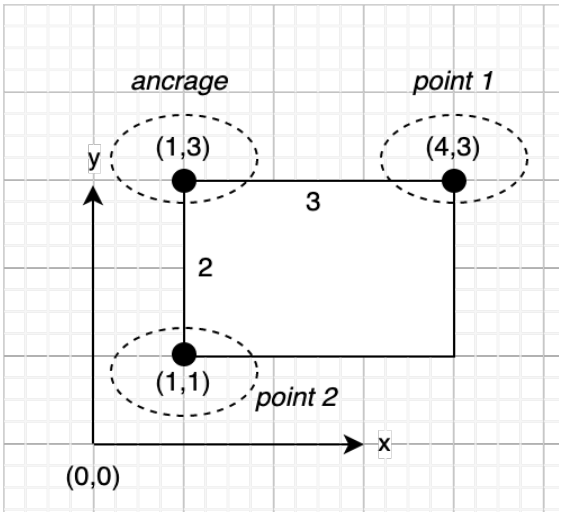


Crée la classe `Rectangle` permettant de créer des rectangles.
Un rectangle est défini par :

- `ancrage` : un `Point` représentant la position du coin supérieur gauche du `Rectangle`
- `largeur` : la largeur du rectangle
- `hauteur` : la hauteur du rectangle

Crée un constructeur pour cette classe permettant de garnir toutes les variables d'instance.

Etape 3



On souhaite maintenant pouvoir unir deux points afin de créer un rectangle.

L'union de deux points consiste à calculer l'ancrage, la largeur et la hauteur du rectangle sur base des coordonnées `(x,y)` de ces deux points. Ces derniers représentent les coins du rectangle à créer.

Dans la classe `Point`, crée la méthode `union` qui permet de créer (retourne) un `Rectangle` à partir de deux points donnés. Le premier point est le point courant (celui sur lequel on appelle la méthode) et le second est reçu en argument.

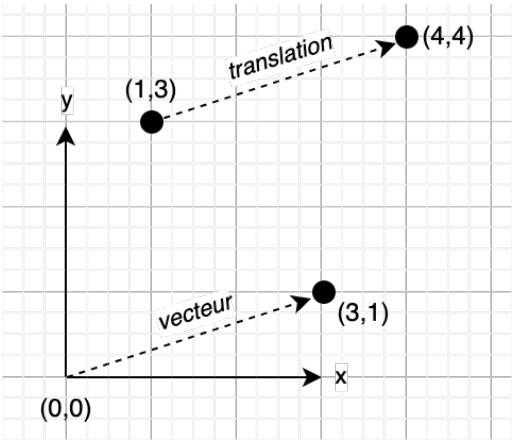
Indice : la méthode `union` doit utiliser l'instruction `new Rectangle(…)`.

Etape 4

On souhaite maintenant effectuer la translation d'un point à l'aide d'un vecteur donné. Un vecteur est représenté par un point.

Crée la méthode `translation` permettant d'effectuer la translation du point courant sur base d'un point reçu en argument.

Attention : la méthode ne retourne rien ! Elle se contente de modifier les variables d'instance du point courant.



Exercice 2 : gestion des personnages dans un RPG

Dans ce programme, chaque personnage peut interagir avec d'autres personnages, ramasser des objets, et obtenir un équipement spécial.

Étape 1

Crée une classe `Personnage` représentant un personnage du jeu avec les attributs suivants :

- `$nom` : le nom du personnage,
- `$niveau` : le niveau du personnage,
- `$pointsDeVie` : les points de vie du personnage,
- `$arme` : l'équipement actuellement porté par le personnage pour combattre (peut être `null`)

Définis un constructeur pour initialiser ces attributs.

Ajoute une méthode `__toString` qui affiche une description du personnage : "Nom : `<nom>`, Niveau : `<niveau>`, Points de Vie : `<pointsDeVie>`".

Étape 2

Crée une classe `Equipement` qui représente un objet d'équipement avec les attributs suivants :

- `$type` : le type de l'équipement (par exemple, "épée", "bouclier"),
- `$puissance` : un nombre entier représentant la puissance de l'équipement.

Définis un constructeur pour initialiser ces attributs.

Étape 3

Dans la classe `Personnage`, crée une méthode `soigner` qui prend en entrée un autre objet `Personnage`. Cette méthode ajoute 10 points de vie au personnage passé en paramètre. Si les points de vie de ce personnage dépassent 100, ils doivent être limités à 100.

Étape 4

Dans la classe `Personnage`, crée une méthode `fabriquerEquipement` qui ne prend aucun paramètre et retourne un nouvel objet `Equipement`. Le type d'équipement et sa puissance peuvent être générés de manière aléatoire ou prédéfinie. Par exemple, cette méthode peut créer une épée avec une puissance de 10.

Étape 5

Dans la classe `Personnage`, crée une méthode `affronter` qui prend un autre objet `Personnage` en entrée et retourne le personnage gagnant du duel entre le personnage courant et celui passé en paramètre. Le gagnant est déterminé par le niveau le plus élevé. La méthode retourne l'objet `Personnage` qui a le niveau le plus élevé, ou le personnage courant en cas d'égalité.

Étape 6

Enfin, toujours dans la classe `Personnage`, crée une méthode `porter` qui fait en sorte que le personne porte actuellement l'arme reçue en entrée (s'il en porte déjà une, cette dernière est remplacée).

Étape 7

Réalise les opérations suivantes :

1. Crée deux personnage : `$perso1` (niveau 2 et 30 de PV)et `$perso2` (niveau 3 et 60 de PV)
2. Fais en sorte que `$perso1` affronte `$perso2`
3. Fais en sorte que `$perso1` soigne `$perso2`
4. Fais en sorte que `$perso1` fabrique une arme
5. Fais en sorte que `$perso2` porte l'arme fabriquée par `$perso1`

Références

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

- Granet, V. (2018), Algorithmique et programmation en Java
- Lonchamp, J. (2019), Conception d'applications en Java/JEE
- Dubisy, F. (2014), Langage Java (Syllabus), Hénallux