TP 4 - AOOP

Olivier Goudet

March 6, 2019

Exercice 1

On souhaite créer deux types d'animaux, des poissons et des oiseaux. Les poissons et les oiseaux ont en commun deux méthodes *manger()* et *dormir()*, ainsi que deux attributs : leur poids et leur âge. Les poissons seulement ont une méthode *nager()*. Les oiseaux seulement ont une méthode *voler()*. Chacune de ces méthodes renvoie une sortie console du type "Je nage" pour la méthode *nager()*. De même pour les méthodes *manger()*, *dormir()* et *voler()*.

Questions

1. Proposez une conception du programme et implémentez la. Quel type de classe peut être utile dans pour factoriser les méthodes et les attributs communs des oiseaux et des poissons ?

2. Créez une classe de test qui instancie deux oiseaux et un poisson. Appelez les fonctions *manger()*, *dormir()*, *nager()* et *voler()* de chacun de ces objets et vérifiez les sorties du programme.

3. Regroupez ces trois animaux dans un tableau. Faites appelle aux méthodes *manger()* de tous ces animaux en parcourant le tableau.

On souhaite maintenant créer deux types d'oiseaux, **Poulet** et **Moineau**, qui héritent des méthodes et attributs de la classe **Oiseau**. Il y a cependant une particularité : les poulets ne volent pas (ou pas bien).

1. Proposez une première implémentation qui redéfinit la méthode *voler()* pour le poulet de façon à afficher "Je ne suis pas capable de voler" quand la méthode *voler()* est appelée.

2. En quoi cette solution n'est pas très "propre" du point de vue de la conception du programme ?

3. Proposer une autre solution avec une interface **Volant** qui précise le comportement des objets qui volent.

4. Créez un nouveau type d'objet volant qui n'est pas un animal par exemple un **Avion**.

5. Créez une classe de test qui instancie une ArrayList qui contient des avions et des moineaux et fait appel à leurs méthodes *voler()* (en parcourant la liste).

6. Peut-on instancier une classe abstraite ? une interface ?

7. Quel est l'intérêt d'utiliser une classe abstraite ?

8. Quel est l'intérêt d'utiliser une interface ?

9. Quels sont les points communs et les différences des classes abstraites et des interfaces ?

10. Pourquoi ne pas utiliser des classes abstraites (dans lesquelles aucunes méthodes ne seraient implémentées) à la place des interfaces ?

Exercice 2

On souhaite créer des formes géométriques pour un éditeur de dessin. Cet éditeur doit avoir une fonction qui permet d'afficher toutes les positions des formes à l'écran. Il doit aussi permettre d'effectuer une translation de toutes ces formes géométriques.

Questions

1. Créez une classe **Point** avec deux attributs x et y qui représentent ses coordonnées dans le plan. Ajoutez un constructeur qui initialise un point.

2. Créez une classe abstraite **FormeGeometrique** qui a un attribut *nom* et une méthode abstraite *perimetre()* qui retourne le périmètre de cette forme géométrique.

3. Créer une classe concrète de forme géométrique de type **Cercle** avec comme attribut son centre qui est de type Point et son rayon. Un cercle doit disposer d'une méthode *toString()* qui permet d'afficher son nom, la position de son centre, son rayon et son périmètre.

On souhaite maintenant créer des formes géométriques de type **Polygone**. Chaque polygone est constitué d'une liste de points. Un polygone doit posséder une méthode *perimetre()* qui permet de calculer son périmètre. Pour calculer ce périmètre on va avoir besoin d'une fonction qui calcule la distance entre deux points. On définit ensuite deux types de polygone : des rectangles (définis par une liste de quatre points) et des triangles (définis par une liste de trois points). Les rectangles et les triangles doivent disposer d'une méthode *toString()* qui permet d'afficher leur type, les positions de l'ensemble de leurs points (contenus dans la liste) et leur périmètre.

1. Proposez une conception du programme qui implémente les classes **Polygone**, **Rectangle** et **Triangle**. Quel type de classe peut être utile dans pour factoriser les méthodes et les attributs communs des rectangles et des triangles ?

On souhaite maintenant ajouter un comportement de translation aux classes **Point**, **Cercle**, **Rectangle** et **Triangle**.

1. Comment définir ce comportement commun aux quatre classes ?

2. Implémentez tout d'abord la méthode *translation()* pour la classe **Point**. Cette méthode doit permettre d'effectuer une translation du point suivant un vecteur de composante vx et vy.

3. Implémentez cette méthode de translation pour le cercle.

4. Implémentez cette méthode de translation pour les classes **Rectangle** et **Triangle**. A quel niveau faut-il mieux définir cette méthode pour factoriser du code.

5. Créez une classe de test avec un tableau de différentes formes géométriques. Affichez l'ensemble de ces formes géométriques, puis effectuez une même translation de ces formes géométriques et affichez à nouveau ces formes.