Spécification et implantation des points

Les exercices suivants montrent comment spécifier et implanter une classe en technologie objet. Même s'il est recommandé de lire tout l'énoncé avant de répondre aux questions, nous vous demandons de ne lire la question suivante qu'après avoir répondu à la question précédente.

Exercice 1 : Spécifier en programmation objet : exemple du point géométrique

1.1 Recenser tout ce qui est défini sur le point géométrique.

1.2 Classer tous les éléments recensés en 1.1 en deux catégories :
- les *requêtes* : informations qui peuvent être demandées à un objet instance de la classe ;
- les *commandes* : services (méthodes) qui peuvent être réalisés par un objet

conformément à la méthode préconisée par Walden et Nerson (méthode BON).

1.3 Nous décrivons avec plus de précision le point géométrique que nous souhaitons spécifier. On représentera les points du plan en s'intéressant aussi bien aux coordonnées cartésiennes (abscisse et ordonnée) qu'aux coordonnées polaires (module et argument). Nous souhaitons pouvoir obtenir la distance qui sépare deux points. Tout point peut être translaté en précisant un déplacement suivant l'axe des X (abscisses) et un déplacement suivant l'axe des Y (ordonnées). Nous souhaitons pouvoir afficher les caractéristiques du point qui se limitent à l'abscisse et l'ordonnée sous la forme du couple (abscisse,ordonnée). Enfin, on veut pouvoir modifier l'abscisse, l'ordonnée, le module ou l'argument d'un point.

Spécifier le point géométrique décrit ci-dessus, c'est-à-dire identifier les requêtes et commandes, dessiner le diagramme d'analyse UML correspondant, ajouter les informations de type, préciser la sémantique (objectif, conditions d'utilisation et effets) des requêtes et commandes.
Rappel : On appelle diagramme d'analyse UML le diagramme de classes faisant apparaître les rubriques requêtes/commandes et non attributs/méthodes.

Exercice 2 : Définir les constructeurs

Intéressons nous aux constructeurs.

2.1 Qu'est-ce qu'un constructeur ? Une classe peut-elle avoir plusieurs constructeurs ? Quel est l'intérêt d'un constructeur ?

2.2 Spécifier un constructeur pour la classe Point qui permet d'initialiser un point à partir de ses coordonnées cartésiennes.

2.3 Peut-on spécifier un second constructeur pour initialiser un point à partir de ses coordonnées polaires ? Pourquoi ?

2.4 Après avoir défini les constructeurs précédents, a-t-on accès au constructeur par défaut (celui qui ne prend pas de paramètres) ? A-t-on intérêt à le définir ? Pourquoi ?

Exercice 3 : Implanter en programmation objet

Maintenant que la spécification des points est terminée, on peut s'intéresser aux implantations possibles. Il s'agit de définir pour une classe :

— son état représenté par les *attributs* ;
— son comportement décrit par ses *méthodes*.

Les commandes et les requêtes deviennent des méthodes. Une requête peut correspondre à une information stockée (si un attribut lui correspond) ou calculée à partir de l'état de l'objet.

3.1 *Choix d'implantation.*

3.1.1 Proposer plusieurs implantations de la spécification d'un point géométrique (en utilisant un diagramme de classe attributs/méthodes) et indiquer dans quelles conditions les utiliser.

3.1.2 Écrire la méthode qui change l'abscisse du point pour chaque implantation envisagée.

3.2 On se propose de coder en Java une implantation particulière des points qui consiste à choisir comme attributs l'abscisse et l'ordonnée du point. Écrire la classe correspondante (version cartésienne). On se limitera à coder les attributs, le constructeur et les méthodes qui permettent de translater un point, l'afficher, changer son abscisse, obtenir son module.
Remarque : Lors de l'implantation de la classe, il est possible d'introduire de nouveaux attributs et/ou méthodes. Cependant ces nouveaux éléments ne seront pas publics mais privés.

Exercice 4 : Utiliser la classe Point

Dans la suite nous considérerons deux programmes qui utilisent la classe Point. Ce pourrait être des applications volumineuses utilisant intensivement les points.

4.1 *ExemplePoint1.* Écrire un programme qui crée un point de coordonnées cartésiennes (1,0), puis affiche son module et son abscisse, le translate de (-1, 1) et, enfin, l'affiche. Indiquer ce que doit afficher son exécution et dessiner l'évolution de la mémoire.

4.2 *ExemplePoint2.* Écrire un programme qui crée un point de coordonnées cartésiennes (1,0), met son abscisse à 10 et affiche son module. Indiquer ce que doit afficher son exécution.

Exercice 5 : Comprendre pourquoi les attributs doivent être privés

Regardons les conséquences de mettre les attributs en public dans la classe Point.

5.1 *Utiliser les attributs.* On suppose que les attributs de la classe Point ont été déclarés en public. L'utilisateur de classe Point peut donc les utiliser. Indiquer les modifications qu'il pourrait faire aux classes ExemplePoint1 et ExemplePoint2 en utilisant les attributs partout où c'est possible. On appellera ExemplePoint1V2 et ExemplePoint2V2 ces classes modifiées.

5.2 *Comprendre le principe de l'accès uniforme.* Le programmeur de la classe Point décide de changer son implantation des points. Il choisit comme attributs les coordonnées polaires, c'est-à-dire le module et l'argument. On dispose donc d'une deuxième version de la classe Point.

Est-ce que la classe ExemplePoint1V2 est correcte ?

5.3 *Comprendre le principe de protection en écriture des attributs.* Le programmeur décide de changer l'implantation de sa classe. Cette fois, il choisit pour attributs les quatre coordonnées. On dispose donc d'une troisième version de la classe Point.

Est-ce que la classe ExemplePoint2V2 est correcte ?

Objectifs :

- Faire la différence entre la spécification et l'implantation en technologie objet
- Savoir comment spécifier et implanter une classe
- Comprendre les notions de classe, attribut, méthode, constructeur, objet/instance, poignée, droit d'accès/visibilité
- Savoir écrire une classe en Java
- Comprendre le rôle des constructeurs
- Comprendre pourquoi les attributs doivent être privés