Bases de la conception orientée objet
Classes, objets, messages
TP 1

Steven Costiou
Stéphane Ducasse
Inria

May 20, 2021

1 Exercice 1 : points et rectangles

Dans cet exercice, nous allons créer et afficher des rectangles sous forme d'objets. Pour cela, nous allons modéliser le concept de rectangle sous forme de classe : les variables d'instance de la classe représenteront la structure d'un rectangle et les méthodes les opérations pour calculer les propriétés du rectangle (aire, périmètre).

Un rectangle est représenté par un couple de points origin (point supérieur gauche) et corner (point inférieur droit). Les propriétés du rectangle sont ensuite calculées à partir de ces deux points.

1.1 Mise en bouche : une petite classe Point

Avant de pouvoir modéliser un rectangle, nous devons modéliser le concept de point. Programmer une classe Point dans un fichier Point.java :

1. Un point possède une variable d'instance x qui est un entier représentant sa position sur l'axe x d'un repère orthonormé,
2. une variable d'instance y qui est un entier représentant sa position sur l'axe y d'un repère orthonormé,
3. des méthodes d'accès en lecture (*getters*) et en écriture (*setters*) sur ces deux variables, appelées *accesseurs*.

Pour instancier des points, vous aurez besoin d'un constructeur :

4. Écrire un constructeur pour notre classe Point qui prend en paramètre deux entiers i et j puis les affecte aux variables d'instance x et x du point en cours d'instantiation via ses accesseurs,

5. écrire une méthode de test qui vérifie qu'un nouveau point possède bien les coordonnées passées à son constructeur.

1.2 La classe Rectangle

Nous pouvons maintenant programmer la classe Rectangle, dans un fichier Rectangle.java :

1. Un rectangle est représenté par un couple d'objets Point : origin (point supérieur gauche) et corner (point inférieur droit).

2. Programmer un constructeur paramétré par les coordonnées des 2 points origin et corner, on veut pouvoir instancier des rectangles comme suit: Rectangle r = new Rectangle(4, 2, 3, 1).

Nous allons programmer en TDD les opérations du rectangle. Cela signifie que nous allons écrire d'abord des méthodes de test qui décrivent et testent le comportement attendu. Ces tests seront en échec car le comportement décrit n'existe pas encore. Dans un second temps, nous mettrons alors en oeuvre ce comportement dans la classe Rectangle afin de se conformer aux tests.

1. Créer une classe de test, et programmer les tests suivants :

   (a) Tester que l'instanciation d'un rectangle via le constructeur de Rectangle produit bien un objet rectangle avec deux points origin et corner possédant des coordonnées correctes.

   (b) Tester les opérations du rectangle *largeur*, *longueur*, *aire*, et *périmètre*, mises en oeuvre respectivement par les méthodes width(), height(), area(), perimeter(). À chaque test, implémenter le code adéquet pour que le test passe.

2. Écrire un test pour une opération *impression*, mise en oeuvre par la méthode print(), qui imprime les propriétés du rectangle dans la console. Par exemple, pour un rectangle instancié comme Rectangle r = new Rectangle(4, 2, 3, 5), envoyer le message r.print() affiche la chaine de caractères suivantes sur la console : *Rectangle: (<4@2> , <3@5>)*. Vous utiliserez les méthodes getX() et getY() des points du rectangle pour obtenir leurs valeurs.

3. Modifier le code de print() sans toucher au test unitaire : au lieu d'utiliser getX() et getY() des points du rectangle, vous demanderez directement aux points de s'imprimer sur la console en leur envoyant le message print(). Votre modification est complète une fois que le test unitaire s'exécute avec succès.