TP4
« Introduction aux tests automatisés avec Junit : le Cercle »

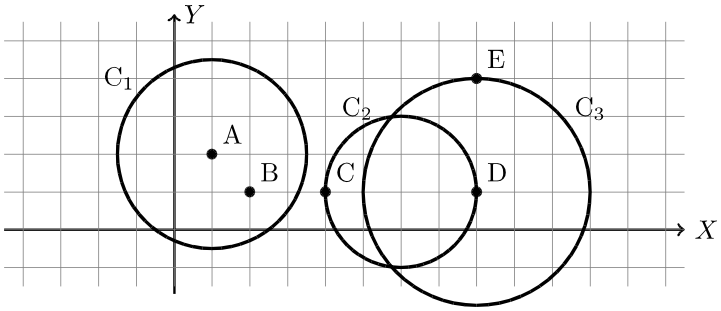
JUnit 4 est un environnement de test intégré dans la plupart des IDE, dont *NetBeans*. Il permet de mettre en œuvre des stratégies de tests, principalement du test unitaire, au moyen d'annotations dédiées : `@Test`, `@Before`, `@After`, `@BeforeClass` et `@AfterClass`.

Dans ce TP, vous devrez d'une part écrire la classe Cercle respectant les exigences et contraintes listées ci-après et d'autre part compléter les tests pour ajouter une vérification de certaines exigences. Les classes fournies vous donnent des exemples de ce qu'il est possible de vérifier (éléments fonctionnels ou structurels).

1ère étape : Spécification du Cercle

Un cercle est une courbe plane fermée constituée des points situés à égale distance d'un point nommé centre. La valeur de cette distance est appelée rayon du cercle. On appelle diamètre la distance entre deux points diamétralement opposés. La valeur du diamètre est donc le double de la valeur du rayon

Exemple : La figure suivante présente trois cercles, nommés C1, C2 et C3.



Exigences

E1. On peut translater un cercle en précisant un déplacement suivant l'axe des X et un déplacement suivant l'axe des Y.

E2. On peut obtenir le centre d'un cercle.

E3. On peut obtenir le rayon d'un cercle.

E4. On peut obtenir le diamètre d'un cercle.

E5. On peut savoir si un point est à l'intérieur (au sens large) d'un cercle. Par exemple, les points A et B sont à l'intérieur du cercle C1 et le point C à l'extérieur. Le point E est à l'intérieur du cercle C3.

E6. Un cercle est un Mesurable2D (il implémente l'interface `Mesurable2D`). À ce titre, on peut obtenir son périmètre et son aire (en fait, il s'agit de l'aire de la surface délimitée par le cercle). Le périmètre d'un cercle est donnée par la formule $2\pi R$ où R représente le rayon du cercle. L'aire est πR^2 .

E7. Non applicable

E8. Le cercle possède une couleur qui est utilisée pour dessiner sa circonférence.

E9. On peut obtenir la couleur d'un cercle.

E10. On peut changer la couleur d'un cercle.

E11. On peut construire un cercle à partir d'un point qui désigne son centre et d'un réel correspondant à la valeur de son rayon. Sa couleur est considérée comme étant le bleu. Par exemple, le cercle C1 est construit à partir du point A de coordonnées (1, 2) et du rayon 2,5.

E12. On peut construire un cercle à partir de deux points diamétralement opposés. Sa couleur est considérée comme étant le bleu. Par exemple, le cercle C 2 est construit à partir des deux points C et D.

E13. On peut construire un cercle à partir de deux points diamétralement opposés et de sa couleur.

E14. Une méthode de classe `creerCercle(Point, Point)` permet de créer un cercle à partir de deux points, le premier correspondant au centre du cercle et le deuxième étant un point du cercle (de la circonférence). Ces deux points forment donc un rayon du cercle. Par exemple, le cercle C3 est construit à partir des points D (centre) et E (circonférence). Le cercle est bleu.

E15. Lorsqu'un cercle est affiché sur le terminal, il est affiché sous la forme suivante Cr@(a, b) où r est la valeur du rayon et (a, b) le centre du cercle, par exemple C2.5@(1.0, 2.0) .

E16. On peut changer le rayon du cercle.

E17. On peut changer le diamètre du cercle.

E18. On ne doit pas pouvoir changer les caractéristiques d'un cercle sans passer par les opérations de modification que la classe propose (`translater()`, `setRayon()`, `setDiametre()`, `setCouleur()`...).

Contraintes

Les contraintes de réalisation à respecter impérativement sont les suivantes :

C1. Il est nécessaire de partir des classes fournies.

C2. Il est interdit de modifier les classes fournies, qu'elles soient de test ou non.

C3. Les lettres accentuées ne doivent pas être utilisées dans les identifiants.

C4. On ne stockera pas d'informations redondantes.

C5. On définira dans la classe `Cercle` une constante appelée `PI` qui sera initialisée à la valeur de π donnée dans la classe `Math` .

2ème étape : Mettre en place JUnit

- Créez un nouveau projet Java « TP3 » sans classe principale
- Téléchargez l'archive `Cercle.zip` et collez son contenu dans le dossier *« src »* de votre projet (identifié comme «Source Packages» dans la vue *Projects* de *NetBeans*).
- Pour pouvoir lancer les tests il faut référencer les librairies « JUnit 4.12 » et « Hamcrest 1.3 » d'une façon similaire à ce que vous avez déjà fait avec les librairies des TP précédents : clic droit sur le projet puis *Properties* > *Libraries* > *Compile Tests* > *Classpath* > + > *Add Library* > ...
- Pour pouvoir créer les tests il faut ensuite faire un clic droit sur le projet puis *New* > *Other...* > *Unit Tests* > *JUnit Test* et valider les choix par défaut. Cela crée un package de tests (un répertoire « test » à la racine de votre projet, identifié comme «Test Packages» dans la vue *Projects* de *NetBeans*) et une classe de tests vide nommée `NewEmptyJUnitTest`.
- Supprimez cette dernière et collez à la place le contenu de l'archive `TestCercle.zip`. Veillez bien à ce que la structure des packages corresponde toujours à celle du dossier *« src »* sinon l'exécution des tests ne fonctionnera pas.
- Les erreurs de compilations qui apparaissent sont liées à des appels de méthodes de la classe `Cercle` que vous n'avez pas encore écrites. Vous ne pouvez donc pas en l'état lancer vos tests.

3ème étape : Implémenter la classe Cercle

- Pour vous aider, vous pouvez ouvrir le fichier `FonctionsCercleTest.java` pour examiner ces erreurs et utiliser l'assistance de *NetBeans* afin de générer les méthodes qui manquent dans la classe `Cercle`. Faites le méthode par méthode, c'est moins risqué, et vérifiez bien les types de renvoi des fonctions générées car *NetBeans* ne peut pas toujours l'inférer à la simple lecture des tests. Modifiez si besoin ces types pour qu'ils correspondent précisément à l'objet renvoyé.
- Dès lors qu'il n'y a plus d'erreur de compilation dans les classes de test vous pouvez lancer ces derniers depuis le menu *Run > Test Project*. Il se peut que vous ayez à afficher la palette des résultats de test pour pouvoir les visualiser (Menu *Window > IDE Tools > Test Results*).
- Les résultats des tests consistent en une liste de réussites et d'échecs accompagnés d'indications. Un double clic sur ces résultats vous mène à la méthode de test dans l'une des deux classes de test fournies. Suivez maintenant les exigences et contraintes de ce document pour implémenter/modifier toutes les méthodes et attributs de la classe `Cercle` en n'oubliant pas de relancer régulièrement les tests pour voir l'évolution de votre travail et vous aider des indications.

Cette méthodologie de développement est appelée *Test Driven Developpement*, ou développement piloté par les tests en français.

4ème étape : Ajouter des tests

- Les tests de la classe `FonctionsCercleTest` ne sont pas exhaustifs. En particulier, il n'y a pas de test pour les exigences E12, E13 et E14. En conséquence, vous devez écrire une classe `SupplementsCercleTest` (nom à respecter impérativement) qui complète les tests et s'appuie sur JUnit 4.
- Il n'y a pas ici de tests de robustesse, par exemple il est possible de créer un cercle avec un rayon nul ou négatif, de créer un cercle à partir de deux points superposés, etc. Comment pourriez-vous améliorer votre code (et vos tests) pour prendre en compte cette préoccupation ? Appliquez votre idée sur un exemple au choix.