

## FICHE 2 – SUITE : ASSOCIATIONS, ENCAPSULATION

### Objectifs

- Comprendre et assimiler la notion d'**encapsulation**.
- Être capable d'écrire les "**getter**" et les "**setter**".
- Être capable d'écrire une méthode `toString()`

### Vocabulaire

encapsulation	getter	setter	private	public	toString
---------------	--------	--------	---------	--------	----------

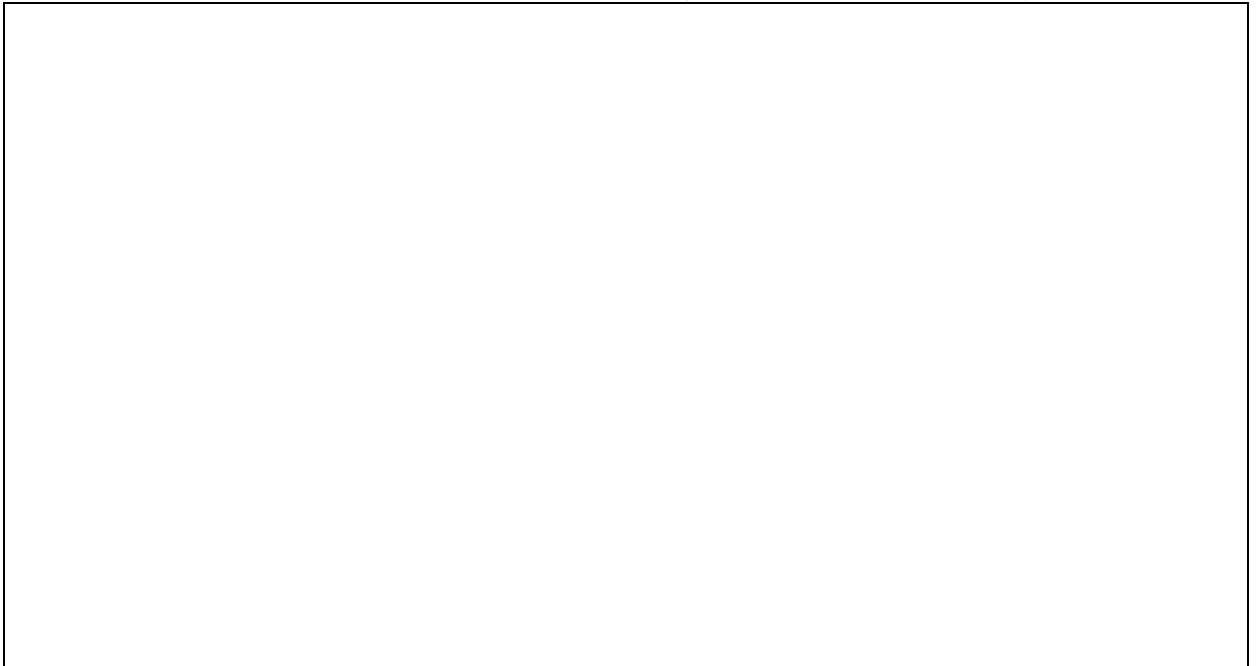
### Exercices

#### 6. Des cercles dans le plan

Un cercle dans le plan est caractérisé par son rayon et par le point où se situe son centre. La classe `CercleDansLePlan` fournit également :

- un constructeur ayant pour paramètres le rayon et le centre dans cet ordre ;
- un getter et un setter pour le rayon du cercle ;
- un getter et un setter pour le centre du cercle ;
- une méthode `toString` qui renvoie le rayon et les coordonnées du centre sous forme de `String`.

a) Définissez la classe `CercleDansLePlan` en UML (réutilisez la classe `Point`) :



- b) Implémentez la classe `CercleDansLePlan` en java !
- c) Écrivez la classe `TestCercleDansLePlan` en respectant les étapes précisées ci-dessous :
1. Créez, dans une variable `point`, un point dont la coordonnée x est à 4 et la coordonnée y à 2.5.

2. Créez, dans une variable `cercle1`, un cercle dans le plan ayant comme rayon 4 et comme centre le point créé ci-dessus.
3. Créez, dans une variable `cercle2`, un cercle dans le plan ayant comme rayon 12.5 et comme centre le point créé ci-dessus.
4. Affichez les informations des deux cercles.
5. Modifiez la coordonnée x du centre de « `cercle2` » pour lui mettre la valeur 8.
6. Affichez les informations des deux cercles.

## 7. Compte en banque

Chaque compte en banque possède comme attributs un titulaire (de type `Personne`), la date de validité de la carte d'identité du titulaire (de type `Date`), un numéro, une date d'ouverture (de type `Date`), un solde. Cette classe possède un constructeur où les différents champs sont fournis en paramètre. Elle possède également un getter pour chaque attribut et une méthode `toString()` qui renvoie une chaîne de caractères reprenant le numéro du compte, le prénom et le nom du titulaire (ajoutez les getters nécessaires dans la classe `Personne`), la date d'ouverture et le solde.

- a) Dessinez le diagramme UML de la classe `CompteEnBanque` sachant que vous devez y intégrer les opérations suivantes :
  - la modification de la date de validité de la carte d'identité du titulaire ;
  - le retrait d'un certain montant du compte ;
  - le dépôt d'un certain montant sur le compte ;
  - le virement d'un certain montant sur un autre compte en banque. Un virement permet de faire un transfert d'un certain montant du compte courant vers un autre compte. Le montant d'un dépôt, d'un retrait ou d'un virement doit toujours être strictement positif. De plus, un retrait ou un virement ne peut être effectué si le solde est insuffisant. Il faut pouvoir déterminer si l'opération (dépôt, retrait ou virement) a pu être effectuée ou non.
- b) Implémentez la classe `CompteEnBanque` en Java.
- c) Décrivez ensuite les tests que vous allez effectuer pour vérifier votre implémentation. Par exemple :
  - Création d'un compte (titulaire de type `Personne` : Leconte, numéro : 000-000000089-89 et solde : 1200)
  - Affichage du compte (appel `toString`)
  - Dépôt sur ce compte de 100
  - Affichage du compte (appel `toString`) -> le solde devrait être de 1300 !
  - ...
- d) Implémentez vos tests en Java dans une classe `TestCompteEnBanque`.

## 8. Académie de musique

Une académie de musique désire informatiser les cours qu'elle organise. On a déjà répertorié 3 classes : `Cours`, `Professeur` et `Eleve`.

Un élève est caractérisé par son nom, son prénom, le cours principal auquel il est inscrit et le cours complémentaire auquel il est inscrit. La classe `Eleve` disposera aussi d'un constructeur permettant d'initialiser tous les attributs, de getters pour tous les attributs et d'une méthode `toString()`. Un élève pourra changer de cours complémentaire.

Un cours est caractérisé par un intitulé, un niveau, le nombre d'inscrits et le professeur qui le dispense. Le nombre d'inscrits n'est pas connu au moment de la création du cours. Pour un cours, il faut pouvoir modifier le professeur. De plus, afin de tenir à jour le nombre d'inscrits, il faut pouvoir signaler l'inscription et la désinscription d'un élève. La classe `Cours` disposera aussi de getters pour tous les attributs et d'une méthode `toString()`.

Un professeur est caractérisé par son matricule, son nom, son prénom et sa spécialisation. La classe `Professeur` disposera aussi d'un constructeur initialisant tous les attributs. Elle aura aussi un getter pour chaque attribut et une méthode `toString()`.

- a) Réalisez le **diagramme de classes** (observez le programme test `TestAcademie` avant de faire le diagramme) :
- b) Implémentez les classes `Eleve`, `Cours` et `Professeur` (pour connaître le format attendu pour les `toString`, observez le contenu du fichier *sortieTestAcademie.txt* qui est la sortie attendue du programme `TestAcademie`). Exécutez le programme de test et vérifiez que la sortie corresponde à ce qui se trouve dans le fichier *sortieTestAcademie.txt*.
- c) Donnez la représentation en mémoire des objets à la fin de l'exécution du programme `TestAcademie`.