# Travaux Dirigés n°1

**Java Avancé**

*—M1 Apprentissage—*

---

## Révisions

Objets, références, égalité, encapsulation, mutabilité, redéfinition etc.

---

### ▸ Exercice 1. Eclipse

1. Que se passe lorsque vous tapez `main` puis ctrl et barre d'espace dans une classe vide ?
2. Même question en tapant toStr puis ctrl et espace dans une classe ?
3. Que se passe t-il si l'on tape sysout et que l'on appuie sur ctrl et espace dans une méthode main ?
4. Créer un champ toto de type int dans une classe. Que se passe t-il si l'on tape get puis ctrl et espace dans la classe ? Et set puis ctrl et espace ?
5. Sélectionner le nom de la classe. Que se passe t-il si l'on tape ⌥ (alt)+⇧+R ? Meme question avec le champ toto.
6. Il peut être utile de voir le code source utilisé par la JDK. Pour cela, télécharger le fichier src.zip sur le site d'oracle (sur les machines du CRIO c'est déjà dans /usr/local/jdk***/src.zip) et attacher-le dans Window › Preferences › Installed JREs › Edit › rt.jar › Source Attachment. Déclarer une variable de type String et cliquer sur String en maintenant la touche ctrl. Que se passe-t-il ?
7. Si la javadoc n'est pas proposée dans Eclipse, mettre `https://docs.oracle.com/javase/8/docs/api/` dans Javadoc Attachment.

### ▸ Exercice 2. Recherche de nombres premiers

Cet exercice constitue un prérequis pour le cours de Java Avancé, si vous ne parvenez pas à le faire, manifestez-vous rapidement.

Rappel : pour tester si un nombre $p$ est premier, une méthode simple consiste à tester tous ses diviseurs potentiels entre 2 et $\sqrt{p}$. Le nombre $p$ est premier si et seulement s'il n'existe aucun $x \in \{2 \dots \sqrt{p}\}$ tel que le *reste* de la division $\frac{p}{x}$ est nul.

1. Créer une classe qui s'appelle `PrimeCollection` et a en champ privé une instance de `java.util.ArrayList<Integer>`.

2. Créer une nouvelle méthode `initRandom(n, m)` pour initialiser une `PrimeCollection` avec `n` entiers tirés aléatoirement entre 1 et `m`. (Pour générer des nombres aléatoires, utiliser `java.util.Random`).
3. Ajouter une méthode privée `isPrime(p)` qui retourne `true` si l'entier `p` passé en paramètre est premier ou `false` sinon.
4. Écrire une méthode `printPrimes` qui affiche uniquement les nombres premiers de la collection.
5. Écrire une méthode `main` pour qui génère une collection de 100 entiers aléatoires tirés entre 1 et 100, et affiche ceux qui sont premiers grâce aux méthodes précédentes.

### ▸ Exercice 3. Parser

1. Créer une méthode qui prend une un tableau de String, un char et renvoie une String correspondant à la concaténation de chaque élément du tableau séparer par le char. Par exemple pour les arguments {"Afghanistan","AF","AFG","004"} et ';' la fonction doit renvoyer "Afghanistan ;AF ;AFG ;004"
2. Créer la méthode inverse à celle d'au dessus : elle prend un String, un char et renvoie un tableau de String.
3. Créer des test unitaires associés à ces deux fonctions.

### ▸ Exercice 4. Point

Le but est d'écrire une classe représentant des coordonnées cartésiennes.

1. Créer une classe `Point` avec deux champs **privés** $x$ et $y$, puis écrire une méthode main avec le code suivant :

```
1 Point p=new Point();
2 System.out.println(p.x+" "+p.y)
```

   Pourquoi cela fonctionne t-il ?
2. Créer une classe `TestPoint` avec un main ayant le même code que précédemment. Que se passe-t-il ? Comment peut-on y remédier ?
3. Pourquoi il faut toujours que les champs d'une classe soient privés ?
4. Qu'est-ce qu'un accesseur ? Doit-on le faire ici ?
5. Créer un constructeur prenant les coordonnées du point en paramètre (appelés `px` et `py`). Quel est le problème au niveau du main ?
6. Modifier les paramètres du constructeur pour les appeler `x` et `y`. Que se passe-t-il ?
7. On veut pouvoir connaitre à tout moment le nombre de points qui ont été créés. Comment faire ?
8. Écrire un autre constructeur prenant un point en argument et utilisant les coordonnées de ce dernier pour la création. Comment le compilateur sait quel constructeur appeler ?

9. Faire en sorte que l'appel à `System.out.println(point);` affiche les coordonnées du point comme ceci : $(x, y)$.

### ▸ Exercice 5. Egalité

On utilise la classe `Point` de l'exercice précédent.

1.
```
Point p1=new Point(1,2);
Point p2=p1;
Point p3=new Point(1,2);

System.out.println(p1==p2);
System.out.println(p1==p3);
```
Qu'affiche ce code ? Pourquoi ?

2. Écrire une méthode `isSameAs(Point)` renvoyant true si deux points ont les mêmes coordonnées.

3.
```
Point p1=new Point(1,2);
Point p2=p1;
Point p3=new Point(1,2);

ArrayList<Point> list = new ArrayList<>();
list.add(p1);
System.out.println(list.indexOf(p2));
System.out.println(list.indexOf(p3));
```
Quel est le problème ? Lire la doc d'`indexOf` (ou faire control+clic gauche sur indexOf) et indiquer quelle méthode est appelée. Modifier la classe `Point` pour résoudre le problème.

4. Que doit renvoyer `p1.equals(new String());` ?

### ▸ Exercice 6. Ligne brisée

On utilise toujours la classe `Point` de l'exercice précédent. On veut maintenant écrire une classe représentant une ligne brisée, c'est-à-dire une suite de points. La ligne brisée aura un nombre maximum de points défini à la création, mais pouvant varier d'une instance à une autre.

1. On utilisera un tableau pour stocker les points d'une ligne brisée. Écrire le constructeur d'une ligne brisée.
2. Écrire une méthode `add` ajoutant un point à la ligne brisée. Si on écrit pas de code supplémentaire, que se passe t il si on dépasse la capacité fixée ? Que faire ?
3. Écrire une méthode `pointCapacity()` et `nbPoints()` indiquant la capacité de la ligne brisée et le nombre de points actuellement sur la ligne.
4. Écrire une méthode `contains` indiquant si un point passé en argument est contenu dans la ligne brisée. Vous utiliserez pour cela une boucle **for each** et non une boucle classique.
5. Que se passe t-il si `null` est passé en argument à la méthode contains ? Et si on a fait un `add(null)` avant ? Regarder la documentation de `Objects.requireNonNull(o)`.

6. Soyez plus moderne et modifier la classe afin qu'elle utilise une `LinkedList` plutôt qu'un tableau (et ainsi ne plus avoir de limite sur sa taille). Que deviennent `pointCapacity, nbPoints` et `contains` ?

### ▸ Exercice 7. Mutabilité et cercle

1. Ajouter une méthode `translate(dx, dy)` à `Point`. Quelles sont les différentes signatures et possibilités pour cette méthode ?
2. Écrire une classe `Circle`, défini comme étant un point (centre) et un rayon, ainsi que son constructeur.
3. Écrire le `toString`.
4. Écrire une méthode `translate(dx, dy)` qui translate un cercle.
5. 
```
Point p=new Point(1,2);
Circle c=new Circle(p,1);

Circle c2=new Circle(p,2);
c2.translate(1,1);

System.out.println(c+" "+c2);
```
   Quel est le problème ? Que faire pour l'éviter ?
6. Quel est le problème si on écrit une méthode `getCenter()` renvoyant le centre ? Pour y réfléchir, que fait le code suivant ?
```
Circle c=new Circle(new Point(1,2), 1);
c.getCenter().translate(1,1);
System.out.println(c);
```
   Modifier pour que cela soit correct.
7. Ajouter une méthode `surface()` et l'ajouter dans l'affichage du cercle.
8. Ecrire la méthode `equals` d'un Circle.
9. Créer une méthode `contains(Point p)` indiquant si le point `p` est contenu dans le cercle.
10. Créer la méthode `contains(Point p, Circle...circles)` qui renvoit vrai si le point est dans un des cercles. Doit-on en faire une méthode statique ?

### ▸ Exercice 8. Anneaux

On veut définir maintenant un anneau : un cercle dont un cercle interne a été supprimé.

1. Est-ce intéressant de faire de l'héritage ici ?
2. Écrire une classe `Ring`, prenant en argument un centre, un rayon et un rayon interne (qui doit être inférieur au rayon).

3. Écrire la méthode `equals`.
4. On veut afficher un anneau avec son centre, son rayon et son rayon interne. Quel est le problème si on fait `System.out.println(ring);` sans code supplémentaire ? Le corriger.
5. Écrire une méthode `contains(Point)` en évitant d'allouer des objets ou de dupliquer du code.
6. Écrire la méthode `contains(Point p, Ring...rings)`.