# Découvrir la programmation orientée objet avec Java

**Travaux pratiques**

frederic.rallo@univ-cotedazur.fr

# TP2 : Types en java

**L'objectif de ce TD est de prendre en main l'IDE Eclipse ou intellij et de manipuler les types usuels de données et les tableaux en Java.**

## Exercice 1

Préparez l'environnement de travail pour les exercices suivants. Dans Eclipse, copiez le projet de la semaine précédente de la façon suivante :
# nom du nouveau projet ➔ TP2
+ créez un package ex2 et placez-y vos classes Point et Vecteur et placez-y la classe qui teste Point (TestPoint.java) et la classe qui teste les Vecteur (TestVecteur.java)
# mettez à jour les infos de version dans la javadoc (ceci devra être automatique pour les prochains exos)

## Exercice 2

Dans la classe Point, définir la méthode public boolean equals(Point p) qui retourne true si le point courant et celui passé en paramètre sont confondus, false sinon.

**Rappel** : *On dit que 2 points A (xA, yA) et B (xB, yB) sont confondus (égaux) si leurs coordonnées sont « suffisamment proches » :* $xA - xB \leq \varepsilon$ *et* $yA - yB \leq \varepsilon$.

```
* ------------------------------------------------------------ *
* TP java                                                      *
*                                                              *
* @author frédéric rallo - frederic.rallo@univ-cotedazur.fr *
* @version TD2   - ex2                                         *
* ------------------------------------------------------------ *


point A (1.0 ; 1.0)
point B (5.0 ; 4.0)
point C (0.3333333333333333 ; 3.141592)
point D (0.3333333 ; 3.141592653589793)

projection du point A sur abscisses X = (1.0 ; 0.0)
projection du point A sur ordonnées Y = (0.0 ; 1.0)
projection du point B sur abscisses X = (5.0 ; 0.0)
projection du point B sur ordonnées Y = (0.0 ; 4.0)

point A =?= A true
point A =?= C false
point C =?= D true
```

# Découvrir la programmation orientée objet avec Java

**Travaux pratiques**

frederic.rallo@univ-cotedazur.fr

## Exercice 3

1. Ajoutez une classe Segment dans laquelle un segment est **composé** de 2 points **distincts**, que l'on nommera *origine* et *extrémite.*

2. Dans un premier temps, on se limitera à la déclaration des attributs, la définition des constructeurs, des accesseurs en lecture, et de la méthode toString.

3. Ecrivez un programme de test de la classe Segment (une classe TestSegment). Compiler et exécuter avec succès.

4. Ajouter dans la classe Segment la méthode longueur qui retournera la longueur de l'objet courant. (Modifiez TestSegment pour tester l'appel à la nouvelle méthode)

5. Introduire dans la classe Segment deux nouvelles méthodes projX et projY permettant respectivement de calculer le projeté d'un segment support sur l'axe des abscisses et sur l'axe des ordonnées. (Mettre à jour TestSegment).

```
* ------------------------------------------------------------------------- *
* TP java                                                                   *
*                                                                           *
* @author frédéric rallo - frederic.rallo@univ-cotedazur.fr  *
* @version TD2   - ex3                                                      *
* ------------------------------------------------------------------------- *

point A (1.0 ; 1.0)
point B (2.0 ; 5.0)
segment AB=[ (1.0 ; 1.0) ; (2.0 ; 5.0) ]
segment S=[ (0.0 ; 0.0) ; (0.0 ; 1.0) ]

---------- longueur
longueur du segment AB=4.123105625617661

---------- projetés
projX du segment AB=[ (1.0 ; 0.0) ; (2.0 ; 0.0) ]
projY du segment AB=[ (0.0 ; 1.0) ; (0.0 ; 5.0) ]
```

# Découvrir la programmation orientée objet avec Java

**Travaux pratiques**

frederic.rallo@univ-cotedazur.fr

## Exercice 4

Le but de cet exercice est de réaliser une classe EnsembleEntierBorne. Ajoutez une classe dans votre projet, package ex4. Cette classe devra permettre de manipuler des ensembles de taille bornée de nombres entiers .

1. **Attributs :** Chaque instance de EnsembleEntierBorne dispose au moins des attributs :
   a. une constante entière MAXIMUM qui contient le nombre maximum d'éléments de l'ensemble,
   b. un tableau de valeurs.

2. **Constructeurs :** l'unique constructeur prend en paramètre la taille de l'ensemble ainsi que la plus grande valeur que peut contenir l'ensemble . En effet, l'ensemble que vous construisez ne peut contenir que des entiers compris entre 0 (inclus) et cet entier (inclus).

3. **Méthodes :**
   - toString() Affiche l'ensemble sous forme {2,4,6,7,99}. Ce sera l'occasion de regarder la javadoc de la classe String et notamment la méthode substring de la classe.
   - add(…) Cette procédure prend un entier en paramètre et le rajoute à l'ensemble (rappel : un ensemble ne peut pas contenir de doublon).
   - remove(…) qui permet de supprimer un élément.
   - findOccurence(…) Cette fonction vérifie si une valeur est dans l'ensemble. Elle retourne la valeur -2 si l'ensemble est plein val n'est pas dans l'ensemble ; -1 si l'ensemble n'est pas plein mais que val n'est pas dans l'ensemble ; ou l'indice si la valeur cherchée est dans l'ensemble (indice est la première occurrence)
   - union(…) Cette fonction renvoie un nouvel ensemble correspondant à l'union entre l'ensemb le courant et celui entré en paramètre.
   - intersect(…) Cette fonction renvoie un nouvel ensemble correspondant à l'intersection entre l'ensemble courant et celui entré en paramètre.

4. **Main :** Créez la classe `TestEntiersBornes` dans le package miseEnOeuvre, la méthode main qui pourra instancier des objets EnsembleEntierBorne et tester les méthodes développées.

# Découvrir la programmation orientée objet avec Java

## Travaux pratiques

frederic.rallo@univ-cotedazur.fr

```
* ------------------------------------------------------------ *
* TP java                                                      *
*                                                              *
* @author frédéric rallo - frederic.rallo@univ-cotedazur.fr   *
* @version TD2    - ex4                                        *
* ------------------------------------------------------------ *

ensemble e1 = { }
ensemble e2 = { }

TEST DE LA METHODE add()
on ajoute --> 5 à e1
 -->  ensemble  e1 = { 5 }
 -->  ensemble  e2 = { }

on ajoute --> 5 (encore) à e1
 -->  ensemble  e1 = { 5 }
 -->  ensemble  e2 = { }

on ajoute --> 1 à e1
 -->  ensemble  e1 = { 5,1 }
 -->  ensemble  e2 = { }

on ajoute --> 2 à e1
 -->  ensemble  e1 = { 5,1,2 }
 -->  ensemble  e2 = { }

on ajoute --> 1 à e2
 -->  ensemble  e1 = { 5,1,2 }
 -->  ensemble  e2 = { 1 }

on ajoute --> 2 à e2
 -->  ensemble  e1 = { 5,1,2 }
 -->  ensemble  e2 = { 1,2 }

on ajoute --> 6 à e2
 -->  ensemble  e1 = { 5,1,2 }
 -->  ensemble  e2 = { 1,2,6 }

on ajoute --> 3 à e2
 -->  ensemble  e1 = { 5,1,2 }
 -->  ensemble  e2 = { 1,2,6 }

on ajoute --> 4 à e2
 -->  ensemble  e1 = { 5,1,2 }
 -->  ensemble  e2 = { 1,2,6 }

on ajoute --> 5 à e2
 -->  ensemble  e1 = { 5,1,2 }
 -->  ensemble  e2 = { 1,2,6 }

TEST DE LA METHODE remove()
```

# Découvrir la programmation orientée objet avec Java

## Travaux pratiques

frederic.rallo@univ-cotedazur.fr

```
on supprime --> 4 à e2
 -->  ensemble  e1 = { 5,1,2 }
 -->  ensemble  e2 = { 1,2,6 }
on supprime --> 3 à e2
 -->  ensemble  e1 = { 5,1,2 }
 -->  ensemble  e2 = { 1,2,6 }

TEST DE LA METHODE union()
ensemble e1 = { 5,1,2 }
ensemble e2 = { 1,2,6 }
 --> ensemble e1 UNION e2 = { 5,1,2,6 }

TEST DE LA METHODE intersect()
ensemble e1 = { 5,1,2 }
ensemble e2 = { 1,2,6 }
 --> ensemble e1 INTER e2 = { 1,2 }
```