



# LU2IN002 JAVADOC, DÉBUGGAGE... VERS PLUS D'AUTONOMIE

Vincent Guigue & Christophe Marsala



- 1 Lire les messages d'erreur dans la console
- 2 Savoir corriger les erreurs les plus courantes
- 3 Savoir chercher dans la documentation officielle JAVA...
- 4 ... Et éventuellement documenter votre propre code

## Exemple d'instructions de compilation/exécution:

```
1 javac Point.java
2 javac MainPoint.java
3 java MainPoint
```


- Les deux premières instructions concernent la compilation...  
⇒ Même s'il n'y a pas de message d'erreur, il faut encore vérifier le bon fonctionnement du programme !
- La troisième ligne exécute le code compilé

## ■ **Compilateur**

- **syntaxe** (;, parenthèses, ...)
- vérifie le **type des variables**,
- l'existence des méthodes/attributs et les niveaux d'accès:
  - les méthodes/attributs existent-elles dans l'objet,
  - les accès sont-ils permis (**public/private**)

## ■ **JVM**

- gestion dynamique des liens (cf redéfinition avec l'héritage)
- gestion des erreurs d'utilisation des objets
  - problème d'instanciation (NullPointerException),
  - dépassement dans les tableaux,
  - gestion des fichiers...
- garbage collector (cf cycle de vie des objets)

 <b>CPoint</b> dessin
<ul style="list-style-type: none"><li>■ x: double</li><li>■ y: double</li></ul>
<ul style="list-style-type: none"><li>● Point(double,double)</li><li>● getX():double</li><li>● getY():double</li><li>● toString():String</li><li>● move(double,double):void</li></ul>

## Compilation (les plus faciles !):


Toujours bien regarder la ligne de l'erreur (elle est donnée).  
Trouver le raccourci de votre éditeur permettant d'aller à la ligne fautive

## Syntaxe & fermeture de blocs

```
1 Point p = new Point(1,2)
2 p.move(1, 0);
3 // Syntax error , insert ";" to complete BlockStatements
```

Toutes les erreurs de fermetures de {}... ⇒ importance **d'indenter** votre code !!!

Il y a souvent **plusieurs erreurs**: toujours regarder la **première**, les autres sont peut-être simplement des conséquences de la première

 <b>CPoint</b> dessin
<ul style="list-style-type: none"><li>■ x: double</li><li>■ y: double</li></ul>
<ul style="list-style-type: none"><li>● CPoint(double,double)</li><li>● getX():double</li><li>● getY():double</li><li>● toString():String</li><li>● move(double,double):void</li></ul>


## Compilation (les plus faciles !):

Toujours bien regarder la ligne de l'erreur (elle est donnée).  
Trouver le raccourci de votre éditeur permettant d'aller à la ligne fautive

## Niveau d'accès

```
1 Point p = new Point(1,2);  
2 p.x = 3;  
3 // The field Point.x is not visible
```

Il y a souvent **plusieurs erreurs**: toujours regarder la **première**, les autres sont peut-être simplement des conséquences de la première

 <b>CPoint</b> dessin
<ul style="list-style-type: none"><li>■ x: double</li><li>■ y: double</li></ul>
<ul style="list-style-type: none"><li>● Point(double,double)</li><li>● getX():double</li><li>● getY():double</li><li>● toString():String</li><li>● move(double,double):void</li></ul>



## Compilation (les plus faciles !):

Toujours bien regarder la ligne de l'erreur (elle est donnée).  
Trouver le raccourci de votre éditeur permettant d'aller à la ligne fautive

## Existence des méthodes

```
1 Point p = new Point(1,2);
2 p.mover(1,3);
3 // The method mover(int, int) is undefined for the type Point
4 p.move(1, 2, 3);
5 // The method move(double, double) in the type Point is
6 // not applicable for the arguments (int, int, int)
```

Il y a souvent **plusieurs erreurs**: toujours regarder la **première**, les autres sont peut-être simplement des conséquences de la première

 <b>Point</b> dessin
<ul style="list-style-type: none"><li>■ x: double</li><li>■ y: double</li></ul>
<ul style="list-style-type: none"><li>●  Point(double,double)</li><li>● getX():double</li><li>● getY():double</li><li>● toString():String</li><li>● move(double,double):void</li></ul>

## Compilation (les plus faciles !):


Toujours bien regarder la ligne de l'erreur (elle est donnée).  
Trouver le raccourci de votre éditeur permettant d'aller à la ligne fautive

## Anticipation des erreurs de la JVM

```
1 Point p;  
2 p.move(1, 0);  
3 // The local variable p may not have been initialized
```

Il y a souvent **plusieurs erreurs**: toujours regarder la **première**, les autres sont peut-être simplement des conséquences de la première



 <b>CPoint</b> dessin
<ul style="list-style-type: none"><li>■ x: double</li><li>■ y: double</li></ul>
<ul style="list-style-type: none"><li>● Point(double,double)</li><li>● getX():double</li><li>● getY():double</li><li>● toString():String</li><li>● move(double,double):void</li></ul>


## Compilation (les plus faciles !):

Toujours bien regarder la ligne de l'erreur (elle est donnée).  
Trouver le raccourci de votre éditeur permettant d'aller à la ligne fautive

## Exploration des chemins de retour

```
1 public double valabs(double d){  
2     if(d>=0) return d;  
3     if(d<0) return -d;  
4 }  
5 // This method must return a result of type double
```

Il y a souvent **plusieurs erreurs**: toujours regarder la **première**, les autres sont peut-être simplement des conséquences de la première

 <b>CPoint</b> dessin
<ul style="list-style-type: none"><li>■ x: double</li><li>■ y: double</li></ul>
<ul style="list-style-type: none"><li>● Point(double,double)</li><li>● getX():double</li><li>● getY():double</li><li>● toString():String</li><li>● move(double,double):void</li></ul>

## Compilation (les plus faciles !):

Toujours bien regarder la ligne de l'erreur (elle est donnée).  
Trouver le raccourci de votre éditeur permettant d'aller à la ligne fautive

## Exploration des chemins impossibles

```
1 public double test(double d){  
2     return d-1;  
3     System.out.println("je ne passe pas ici");  
4 }  
5 // Unreachable code
```

Il y a souvent **plusieurs erreurs**: toujours regarder la **première**, les autres sont peut-être simplement des conséquences de la première

## Execution (JVM) : Toujours vérifier la ligne également

### ■ NullPointerException

```
1 Point p = null;  
2 p.move(1, 0);  
3 // Exception in thread "main" java.lang.NullPointerException  
4 //       at cours1.TestPoint.main(TestPoint.java:11)
```

- Cette erreur arrive souvent dans des cas plus complexe de composition d'objet

### ■ IndexOutOfBoundsException

```
1 int[] tab = new int[3];  
2 tab[3] = 2;  
3 // Exception in thread "main"  
4 //       java.lang.ArrayIndexOutOfBoundsException: 3  
5 //       at cours1.TestPoint.main(TestPoint.java:11)
```

- Vérifier la ligne + l'index !
- Souvent dans les boucle for

```
1 public class Segment{
2     private Point p1, p2;
3
4     public Segment(){
5     }
6
7     public void decaler(){
8         p1.move(1, 1);
9         p2.move(1, 1);
10    }
11
12 }
```

```
1 public class Test{
2     public static void main(String [] args){
3         Segment s1 = new Segment();
4         s1.decaler();
5     }
6 }
```

⇒ Ca compile... Mais **NullPointerException**

```
1 public class Point{
2     private double x,y;
3
4     public Point(double x2,
5                 double y2){
6         double x = x2;
7         double y = y2;
8     }
9
10    public void move(double d1,
11                    double d2){
12        x+=d1; y+=d2;
13    }
14 }
```

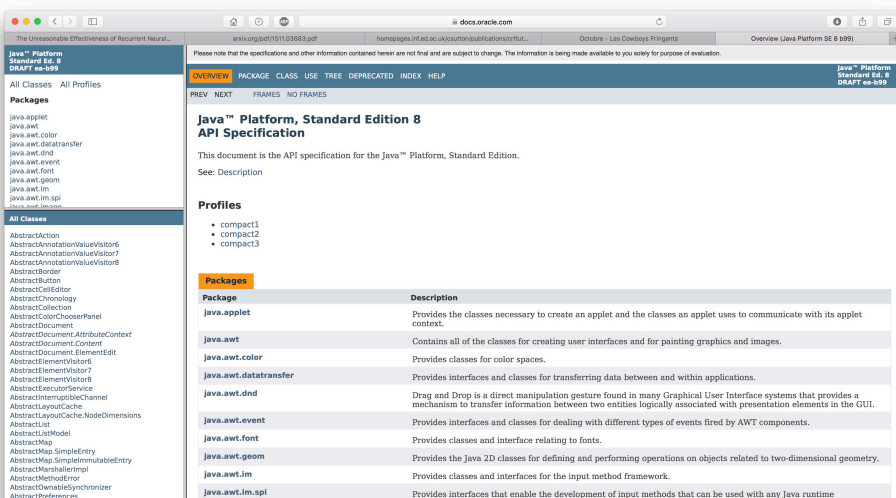
```
1 public class Segment{
2     private Point p1, p2;
3
4     public Segment(){
5         Point p1 = new Point(1,2);
6         Point p2 = new Point(3, 4);
7     }
8
9     public void decaler(){
10        p1.move(1, 1);
11        p2.move(1, 1);
12    }
13
14 }
```

```
1 public class Test{
2     public static void main(String [] args){
3         Segment s1 = new Segment();
4         s1.decaler();
5     }
6 }
```

⇒ Ca compile, ça s'exécute... Mais qu'est ce que ça donne?

Java est un langage très bien documenté et plein d'outils:

<https://docs.oracle.com/javase/8/docs/api/index.html>



The screenshot shows the Java Platform Standard Edition 8 API Specification website. The browser address bar displays `docs.oracle.com`. The page title is "Java™ Platform, Standard Edition 8 API Specification". The left sidebar contains a navigation menu with "All Classes" and "All Profiles". The main content area shows the "Overview" tab selected, displaying the title "Java™ Platform, Standard Edition 8 API Specification" and a description: "This document is the API specification for the Java™ Platform, Standard Edition." Below this, there is a section for "Profiles" listing compact1, compact2, and compact3. A "Packages" section is also visible, listing various Java packages and their descriptions.

Please note that the specifications and other information contained herein are not final and are subject to change. The information is being made available to you solely for purpose of evaluation.

Overview (Java Platform SE 8 b99)

OVERVIEW PACKAGE CLASS USE TREE DEPRECATED INDEX HELP

PREV NEXT FRAMES NO FRAMES

## Java™ Platform, Standard Edition 8 API Specification

This document is the API specification for the Java™ Platform, Standard Edition.

See: Description

### Profiles

- compact1
- compact2
- compact3

### Packages

Package	Description
java.applet	Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.
java.awt	Contains all of the classes for creating user interfaces and for painting graphics and images.
java.awt.color	Provides classes for color spaces.
java.awt.datatransfer	Provides interfaces and classes for transferring data between and within applications.
java.awt.dnd	Drag and Drop is a direct manipulation gesture found in many Graphical User Interface systems that provides a mechanism to transfer information between two entities logically associated with presentation elements in the GUI.
java.awt.event	Provides interfaces and classes for dealing with different types of events fired by AWT components.
java.awt.font	Provides classes and interface relating to fonts.
java.awt.geom	Provides the Java 2D classes for defining and performing operations on objects related to two-dimensional geometry.
java.awt.im	Provides classes and interfaces for the input method framework.
java.awt.im.spi	Provides interfaces that enable the development of input methods that can be used with any Java runtime

Java est un langage très bien documenté et plein d'outils:

<https://docs.oracle.com/javase/8/docs/api/index.html>

## ■ Cas 1: objet connu, usage inconnu

- Quels sont les constructeurs d'une `ArrayList`?
- Comment faire un cosinus (en se doutant que c'est dans la classe `Math`)?
- ...

⇒ Chercher directement dans la classe au niveau de la javadoc

All Methods Static Methods Concrete Methods

Modifier and Type	Method and Description
static double	<b>abs</b> (double a) Returns the absolute value of a double value.
static float	<b>abs</b> (float a) Returns the absolute value of a float value.
static int	<b>abs</b> (int a) Returns the absolute value of an int value.
static long	<b>abs</b> (long a) Returns the absolute value of a long value.
static double	<b>acos</b> (double a) Returns the arc cosine of a value; the returned angle is in the range 0.0 through $\pi$ .

Java est un langage très bien documenté et plein d'outils:

<https://docs.oracle.com/javase/8/docs/api/index.html>

## ■ Cas 1: objet connu, usage inconnu

- Quels sont les constructeurs d'une ArrayList?
- Comment faire un cosinus (en se doutant que c'est dans la classe Math)?
- ...

## ■ Cas 2 : recherche d'une fonctionnalité sans point d'entrée

ex : Comment créer une image en JAVA?

- Google...
  - recherche d'un point d'entrée  $\Rightarrow$  BufferedImage  $\Rightarrow$  retour à la javadoc
  - recherche d'un tutoriel (idéalement officiel):

<https://docs.oracle.com/javase/tutorial/2d/images/>



De manière générale, **on programme pour les autres...**

⇒ documenter son code pour le rendre utilisable

- 1 premier niveau: choisir des noms de classes, méthodes et variables **explicites**.
- 2 deuxième niveau: faire des classes et des méthodes courtes, utiliser des méthodes privées... Eviter à tout prix les copier-coller.
- 3 troisième niveau: ajouter des commentaires pour créer une documentation.
  - Outil intégré dans JAVA: commentaires spéciaux + création automatique d'une page web

```
1 /**
2  * @author Vincent Guigue
3  * Cette classe permet de gerer des points en 2D
4  */
5 public class Point {
6     /**
7      * Attributs correspondant aux coordonnees du point
8      */
9     private double x, y;
10    /**
11     * Constructeur standard a partir de 2 reels
12     * @param x : abscisse du point
13     * @param y : coordonnee du point
14     */
15    public Point(double x, double y) {
16        this.x = x;
17        this.y = y;
18    }
19    /**
20     * @return l'abscisse du point
21     */
22    public double getX() {
23        return x;
24    }
25 }
```

```
$ javadoc Point.java
```

- De manière générale: vérifier la documentation

```
$ javadoc -h
```

- Pour gérer les accents:

```
$ javadoc -encoding utf8 -docencoding utf8 -charset utf8 [fichier.java]
```

- Pour sélectionner le répertoire de stockage du html:

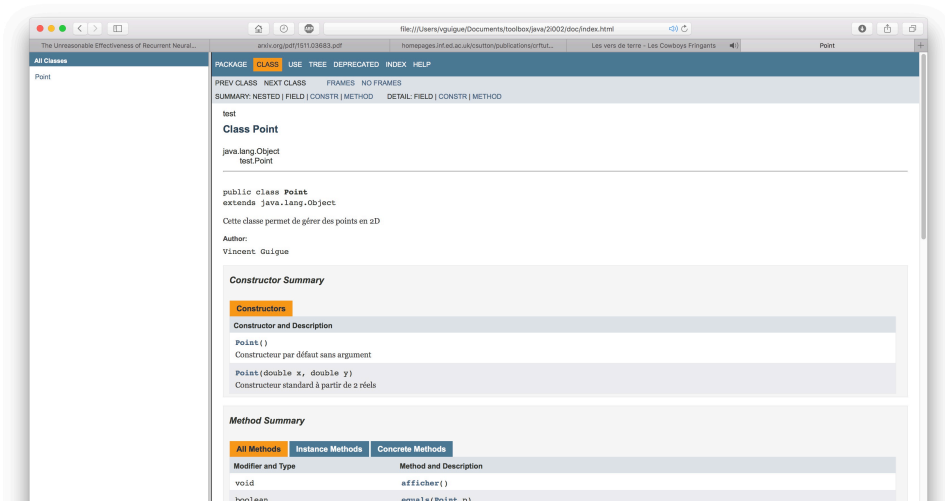
```
$ javadoc -d <directory> [fichier.java]
```

- Représentation public/private

(par défaut, représentation de la partie public seulement)

```
$ javadoc -public/-private [fichier.java]
```

- Classe Point, présentation conforme à la javadoc standard (présence des liens hypertextes...)

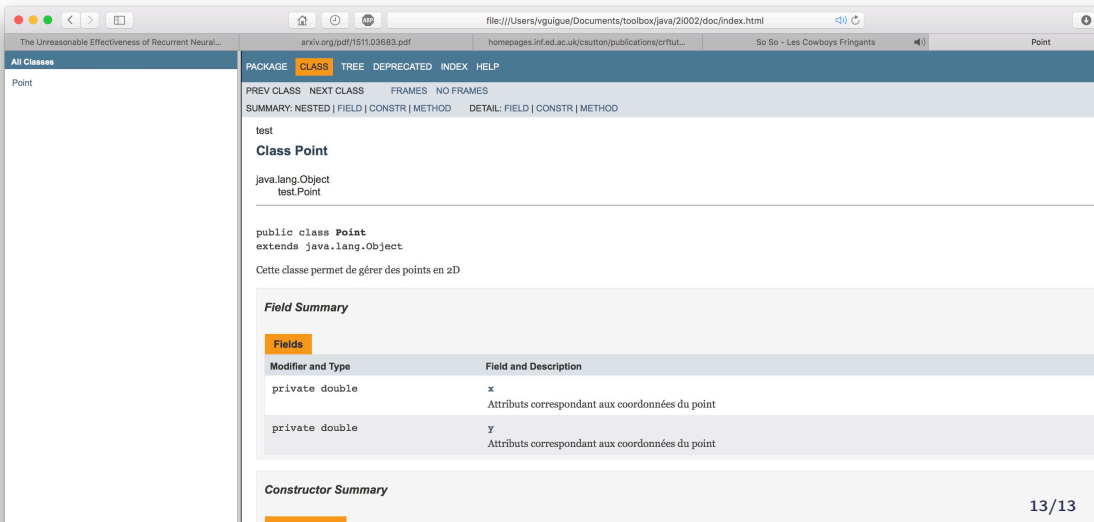


## ■ Classe Point, description des méthodes (entrées, sorties...)

The screenshot shows a web browser window with the address bar displaying `file:///Users/vguigue/Documents/toolbox/java/2i002/doc/index.html`. The browser has several tabs open, including "The Unreasonable Effectiveness of Recurrent Neural...", "arxiv.org/pdf/1511.03683.pdf", "homepages.inf.ed.ac.uk/csutton/publications/crftut...", and "Les vers de terre - Les Cowboys Fringants". The main content area displays the Javadoc for the `Point` class. On the left, a sidebar titled "All Classes" lists `Point`. The main content area shows the following methods:

- toString**  
`public java.lang.String toString()`  
Overrides:  
`toString` in class `java.lang.Object`
- move**  
`public void move(double dx, double dy)`  
Méthode de déplacement d'un point  
Parameters:  
`dx`: - déplacement en x  
`dy`: - déplacement en y
- afficher**  
`public void afficher()`

## ■ Classe Point, représentation privée



The screenshot displays a web browser window showing the Javadoc documentation for the `Point` class. The browser's address bar indicates the file path: `file:///Users/vguigue/Documents/toolbox/java/2i002/doc/index.html`. The page features a navigation bar with links to `PACKAGE`, `CLASS` (highlighted), `TREE`, `DEPRECATED`, `INDEX`, and `HELP`. Below the navigation bar, there are links for `PREV CLASS`, `NEXT CLASS`, `FRAMES`, and `NO FRAMES`. The main content area shows the class signature `public class Point` and a brief description: "Cette classe permet de gérer des points en 2D". Below this, the `Field Summary` section is visible, containing a table with two fields: `x` and `y`, both of type `private double`. The table is as follows:

Modifier and Type	Field and Description
<code>private double</code>	<code>x</code> Attributs correspondant aux coordonnées du point
<code>private double</code>	<code>y</code> Attributs correspondant aux coordonnées du point

The bottom of the page shows the beginning of the `Constructor Summary` section.