

Chapitre 1

Introduction à Java



Sommaire

Introduction à Java

- Qu'est ce que Java ?
- Introduction à l'édition Standard de Java (Java SE)
- Prise en main de l'IDE Eclipse
- Le premier programme Java
- Programmation structurée en Java
- Débogage de programmes Java

Qu'est ce que Java ?

Qu'est-ce que Java ?

Introduction à Java

- Un langage de programmation orienté **objets pur** développé par J.Gosling chez Sun Microsystems en 1995 (Sun a été racheté en 2008 par Oracle Corp)
 - Qualité, productivité, et maîtrise de la complexité d'applications
- Java adopte une architecture de « **Machine Virtuelle** » qui abstrait la machine physique
 - Portabilité
- Un ensemble d'API (Application Programming Interface, ou bibliothèques) riches et variées
 - Networking, accès aux bases de données, distribution, interfaces graphiques, ...
- Les spécifications de Java et les outils de développement JDK sont publics
 - Java se veut un standard pour le développement d'applications d'entreprise
 - Les variantes JDK sont disponibles sur <http://www.oracle.com/technetwork/java/index.html>



James Gosling

Historique de l'évolution de Java

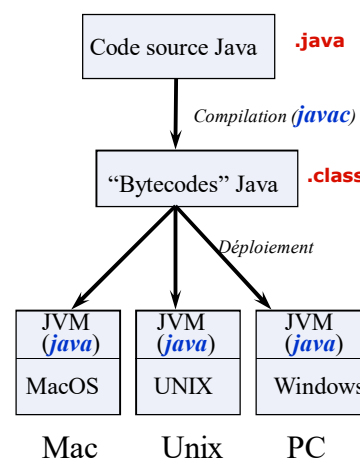
Version Name	Code Name	Release Date
JDK 1.0	Oak	January 1996
JDK 1.1	(none)	February 1997
J2SE 1.2	Playground	December 1998
J2SE 1.3	Kestrel	May 2000
J2SE 1.4	Merlin	February 2002
J2SE 5.0	Tiger	September 2004
Java SE 6	Mustang	December 2006
Java SE 7	Dolphin	July 2011
Java SE 8		March 2014
Java SE 9		September, 21st 2017
Java SE 10		March, 20th 2018
Java SE 11		September, 25th 2018

M.Romdhani, Mars 2019

5

Mécanisme d'exécution des programmes Java

- Le code source java est compilé en un format indépendant des machines appelé bytecode
- Le bytecode est interprété par la machine virtuelle déjà installée sur la machine physique
 - Pour tout type de machine (OS/Processeur), il y a une JVM spécifique. Toutes ces JVM sont disponibles et gratuites
- Le bytecode peut également être converti en un code machine et peut ainsi être exécuté rapidement. Pour cela, il faut compiler avec un compilateur spécial dit Just in Time (JIT).
 - Mais la portabilité est ainsi perdue !



M.Romdhani, Mars 2019

6

Utilisations de Java

■ Langage très utilisé dans le milieu professionnel

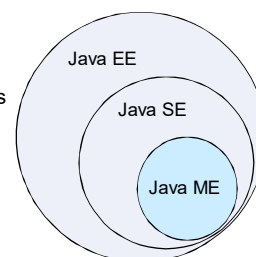
■ Possibilités d'utilisation

- Applications mobiles (avec Java ME et Android)
- Applications en console
- Applications avec interfaces graphiques
- Sites Web dynamiques (avec Java EE)
- Intégration inter-applicative
- Web Services / Micro-services

Les trois éditions de Java

■ Java offre 3 éditions :

1. Java Micro Edition (Java ME, anciennement J2ME)
 - Application pour terminaux mobiles (PDA, Applications mobiles, ...)
2. Java Standard Edition (Java SE, anciennement J2SE)
 - Applications "stand-alone" , Desktop Apps
3. Java Enterprise Edition (Java EE, anciennement J2EE)
 - Applications à large échelle, applications distribuées, applications Web, ...



■ Les trois éditions de Java exploitent les mêmes facilités syntaxiques du langage Java

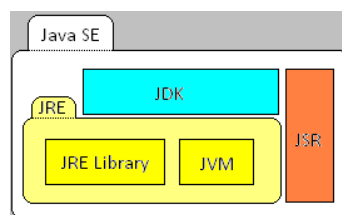
- Chaque édition définit un ensemble de bibliothèques de classes différent

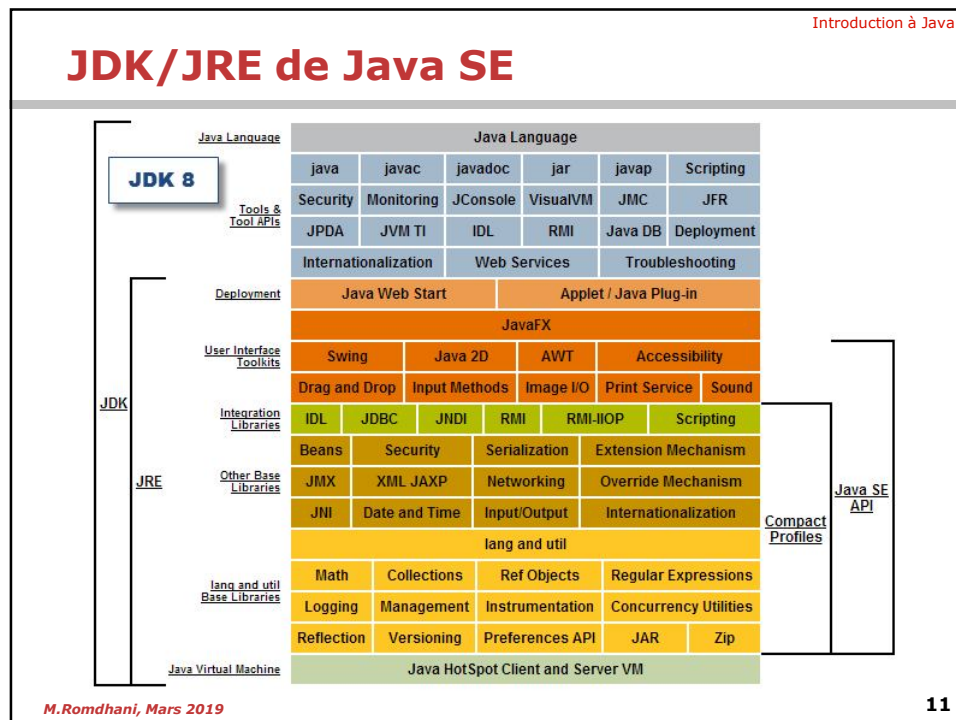
Introduction à l'édition Standard de Java (Java SE)

Java SE Platform

Introduction à Java

- **Java Platform, Standard Edition, ou Java SE (anciennement Java 2 Platform, Standard Edition, ou J2SE), est une spécification de la plateforme Java d'Oracle, destinée typiquement aux applications pour poste de travail**
- **À chaque version de Java SE correspond notamment, comme toutes les éditions Java :**
 - Les Java Specification Requests (JSR), constituant les spécifications de la version considérée ;
 - Un Java Development Kit (JDK), contenant les bibliothèques logicielles et les outils de développement;
 - un Java Runtime Environment (JRE), contenant le seul environnement d'exécution (compris de base dans le JDK).





Introduction à Java

API (Library) de Java SE

■ Le succès de Java tient, entre autres, à la richesse de son API. L'API est organisée en packages chacun couvrant un domaine d'application.

- **java.lang** – Package systématiquement visible contenant des classes fondamentales : Object, System, String, Thread, ..
- **java.util** – Utilitaires de programmation : Date, List, Stack, Vector, ...
- **java.io** – Classes pour les différents modes d'entrée/sortie
- **java.net** – Programmation réseau : Sockets TCP, URL, Datagrammes, ...
- **java.sql** – Programmation des accès aux bases de données relationnelles
- **java.rmi** – Développement d'applicatifs Java distribué sur plusieurs machines
- **javax.xml** – Développement XML
- **java.awt** – Développement d'interfaces graphiques
- **javax.swing** – Développement d'interfaces graphique JFC, amélioration des performances de AWT apparue depuis Java 2.

M. Romdhani, Mars 2019 12

Les outils pour programmer Java

■ L'indispensable JDK (Java Development Kit)

- Récupérer le JDK 8.x d'Oracle pour Windows 64bits et l'installer.
(<https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>)

- Positionner la variable d'environnement **JAVA_HOME**
(Variable Système) de manière à ce qu'elle pointe sur
le dossier d'installation du JDK
(Exemple C:\Program Files\Java\jdk1.8.0_181)

Nom de la variable :

Valeur de la variable :

- Ajouter **%JAVA_HOME%\bin** au contenu
de la variable système **PATH**

Nom de la variable :

Valeur de la variable :

- Vérifier que le JDK soit bien installé

- Lancer une fenêtre Dos Shell (**Windows-R** puis **cmd**) et exécuter **> java -version**

```
C:\> java -version
java version "1.8.0_181"
Java(TM) SE Runtime Environment (build 1.8.0_181-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.181-b13, mixed mode)
```

■ Un environnement de développement intégré (IDE)

- Un IDE n'est pas obligatoire pour programmer Java, mais il amène un confort d'édition, d'accès rapide à la compilation, et d'exécution et de débogage.

- IDE gratuits : Eclipse SDK (www.eclipse.org), Apache NetBeans
(<https://netbeans.apache.org/download/nb100/index.html>)

- IDE propriétaire: IntelliJ IDEA (<https://www.jetbrains.com/idea/>), IBM WebSphere
Rational Application Developer
(<https://www.ibm.com/developerworks/downloads/r/rad/index.html>)

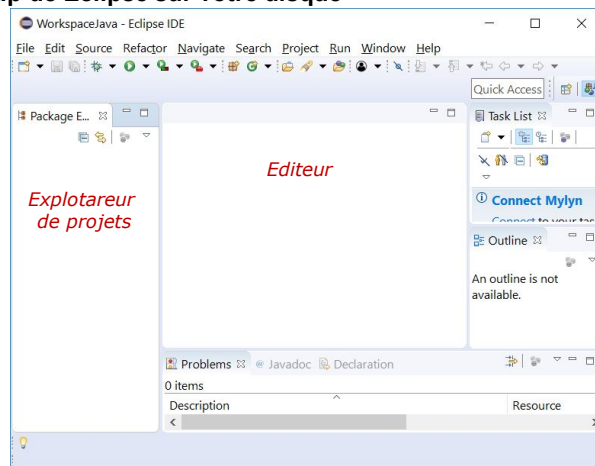
M. Romdhani, Mars 2019

13

Prise en main de l'IDE Eclipse

Introduction à l'IDE Eclipse

- Télécharger le package Eclipse IDE for Java Developers :
<https://www.eclipse.org/downloads/packages/>
- Décompresser le zip de Eclipse sur votre disque
- Aperçu du workbench de Eclipse



M.Romdhani, Mars 2019

15

Workspace et Projets Eclipse

- Le Workspace d'Eclipse
 - Le Workspace est le répertoire (folder) qui permet de stocker tous les projets
 - Le chemin de votre Workspace vous est demandé au lancement du programme.
 - Vous pouvez également le changer manuellement dans Eclipse en cliquant sur File\Switch Workspace\Other
- Un Projet Eclipse est une collection de codes
 - Un projet est stocké dans un répertoire (folder) du Workspace
 - Un projet Eclipse se compose d'un folder src pour le code source et d'un folder bin pour le code compilé
 - Un projet est configuré via .project et .classpath

M.Romdhani, Mars 2019

16

Perspectives Eclipse

■ Une perspective présente une partie du projet de développement selon un certain angle de vue


- Une perspective se compose d'un ensemble de Vues

- Exemple de perspectives :

- Java (Perspective par défaut)
- Debug
- Java Browsing
- Git,

Debug
 Git
 Java (default)
 Java Browsing
 Java Type Hierarchy
 Planning
 Resource
 Team Synchronizing
 XML

■ Changer de perspective

- Icône Open Perspective  dans la barre des outils
- Menu Windows > Perspective> Open Perspective

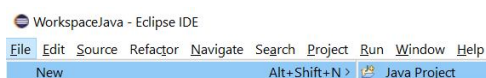
■ Réinitialiser une perspective

- Menu Windows > Perspective> Reset Perspective

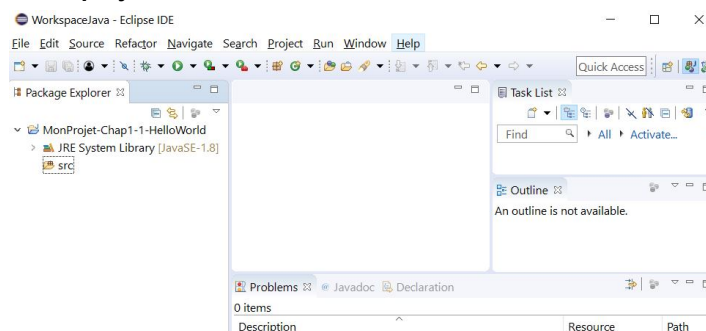
Créer un projet sous Eclipse

■ Pour créer un projet,

- Menu File > New (ALT-SHIFT-N), puis Java Project



■ Structure du projet créé



Le premier programme Java

Ajouter la Classe du programme et son package Introduction à Java

- On suppose que le projet Eclipse est créé. Pour ajouter une classe:
 - Click-droit sur le projet > New > Class
 - Spécifier le nom du package, le nom de la classe, et cocher la case pour générer la méthode main() qui est le point d'entrée

Java Class
Create a new Java class.

Source folder: MonProjet-Chap1-1-HelloWorld/src Browse...

Package: be.businesstraining Browse...

☐ Enclosing type: Browse...

Name: HelloWorld

Modifiers: ☒ public ☐ package ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass: java.lang.Object Browse...

Interfaces: Add...

Which method stubs would you like to create?

☒ public static void main(String[] args)

☐ Constructors from superclass

☒ Inherited abstract methods

? Finish Cancel

Compléter la méthode main()

■ Ajouter à la méthode main() une instruction qui affiche « Hello World !»

```

1 package be.businessstraining;
2
3 public class HelloWorld {
4
5     public static void main(String[] args) {
6         System.out.println("Hello World !");
7     }
8 }

```

- Pour saisir System.out.println(), utiliser le raccourci **sysout** puis CTRL-SPACE
- Eclipse compile au fur et à mesure de l'édition

■ Lancer l'exécution du programme:

- Menu Run > Run (CTRL+F11)
- Click-droit sur le projet > Run As > Java Application
- Le résultat apparaît dans la vue Console

```

Problems Javadoc Declaration Console
<terminated> HelloWorld [Java Application] C:\Program Files\Java\jdk1.8.0_181\bin\javaw.exe
Hello World !

```

M.Romdhani, Mars 2019

21

Commentaires sur le programme Hello world !

■ La POO pure en action ...

- Rien que pour afficher « Hello World » nous avons développé une classe.
- Nous avons utilisé les classes **System** et **String** ; ces classes appartiennent à **java.lang** qui est implicitement importé

■ La méthode main() doit être static

- main doit être une méthode de classe pour qu'on puisse l'invoquer directement à partir de la classe sans besoin de construire une instance de la classe.

■ La méthode main() doit déclarer l'argument string

- main() doit déclarer la chaîne d'arguments même si on lui passera rien lors de l'exécution.

M.Romdhani, Mars 2019

22

Compiler et exécuter à partir de la ligne de commande

Introduction à Java

- Ouvrir la console de commande Windows (cmd.exe)
- Compiler à partir de la ligne de commande
 - Naviguer jusqu'au dossier source **src** (qui contient nos packages)
 - Utiliser la commande :
 - > `javac be/businessstraining/HelloWorld.java`
- Exécuter à partir de la ligne de commande
 - Naviguer jusqu'au dossier **src**(qui contient nos packages)
 - Utiliser la commande :
 - > `java be.businessstraining.HelloWorld`

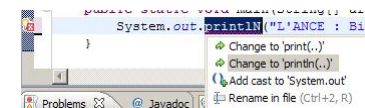
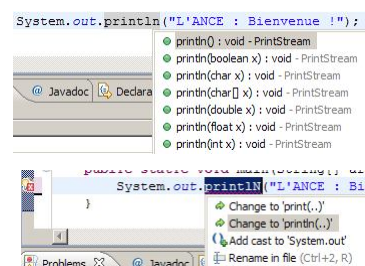
M. Romdhani, Mars 2019

23

Etre productif avec Eclipse

Introduction à Java

- IntelliSense
 - Lors de l'édition d'un programme
- Fixation automatique d'erreurs
- Refactorisation de programmes
 - Faites "Menu Contextuel", puis Refactor
 - Parmi les opérations de refactorisation : Générer les Setters/Getters, Extraction d'interface, ...
- Quelques raccourcis
 - **Ctrl+Space** : Content assist
 - **Ctrl+M** : Maximizes the current view or editor.
 - **Ctrl+O** : Open the Quick Outline view
 - **Ctrl-Shift-O** : Organize imports
 - **Ctrl + /** : commenting, uncommenting lines and blocks
 - **Ctrl-F11** : Run application
 - **CTRL SHIFT L** : avoir la liste de tous les raccourcis



M. Romdhani, Mars 2019

24

Programmation structurée en Java

Introduction à Java

Programmation structurée en Java

■ Types prédéfinis (8 au total)

- Entiers
 - **byte** (8 bits), **short** (16 bits), **int** (32 bits), **long** (64bits)
- Réels
 - **float** (32 bits IEEE 754), **double**(64 bits)
- Caractères
 - **char** (2 octets, unicode)
- Logique
 - **boolean** (1 octet)

■ Structures de contrôle de flux d'exécution (même syntaxe qu'en C/C++)

- Schémas conditionnels
 - if
 - switch-case
- Schémas itératifs
 - for
 - while
 - do-while

Conversions

■ Entre types numériques

```
int myIntVar = 111;
long myLongVar = (long) myIntVar;
```

■ D'une chaîne de caractères vers un type numérique

```
int myIntVar = Integer.parseInt("111");
double myBoubleVar = Double.parseDouble("111.222");
```

■ Vers une chaîne de caractère

```
int myIntVar = 111;
String myStringVar = String.valueOf(myIntVar);
```

Les tableaux

■ Ce sont simplement des ensembles d'éléments de même type.

- La taille du tableau est fixée à la création et ne peut pas être modifiée.
- On peut retrouver les éléments grâce à leur **index**, qui représente en quelque sorte leur place dans le tableau.
- L'**index d'un tableau commence à 0**, ce qui signifie qu'un tableau de 5 élément contiendra des éléments aux index 0, 1, 2, 3 et 4.

■ Déclaration d'un tableau

- Déclaration et initialisation en même temps


```
int[] notes = {15, 16, 17, 18}; // 1
int notesAutrement [] = {15, 16, 17, 18}; // Equivalent à 1
```
- Déclaration et initialisation séparés


```
int[] notes = new int[4];
notes[0]= 15; notes[1]= 16, ...
```

if-else Branching

- The basic format of an if statement is as follows:

```
if (booleanExpression) {
    System.out.println("Inside if statement");
}
```

- The following code demonstrates a legal if-else statement:

```
if (x > 3) {
    System.out.println("x is greater than 3");
} else {
    System.out.println("x is not greater than 3");
}
```

- An else clause belongs to the innermost if statement to which it might possibly belong (in other words, the closest preceding if that doesn't have an else

```
if (exam. done())
    if (exam.getScore() < 0.61)
        System.out.println("Try again.");
    // Which if does this belong to?
else
    System.out.println("Java master!");
```

switch Statements

- Legal Expressions for switch and case

- The general form of the switch statement is:

```
switch (expression) {
    case constant1: code block
    case constant2: code block
    default: code block
}
```

- A switch's expression must evaluate to a **char**, **byte**, **short**, **int**, or, as of Java 5, an **enum**. You won't be able to compile if you use anything else, including the remaining numeric types of long, float, and double. As of Java 7, Switching over Strings is possible.

Using while and do Loops

■ A while statement looks like this:

```
while (expression) {
    // do stuff
}
```

- Any variables used in the expression of a while loop must be declared before the expression is evaluated. In other words, you can't say

```
while (int x = 2) {} // not legal
```

■ The following shows a do loop in action:

```
do {
    System.out.println("Inside loop");
} while (false);
```

■ Take a look at the following examples of legal and illegal while expressions:

```
int x = 1;
while (x) {} // Won't compile; x is not a Boolean
while (x = 5) {} // Won't compile; resolves to 5 (as the result of assignment)
while (x == 5) {} // Legal, equality test
while (true) {} // Legal
```

Using for loops

■ A typical example of a for loop.

```
for (/*Initialization*/ ; /*Condition*/ ; /* Iteration */) {
    /* loop body */
}
```

- The Basic for Loop: Declaration and Initialization

- If you declare more than one variable of the same type, then you'll need to separate them with commas as follows:

```
for (int x = 10, y = 3; y > 3; y++) {}
```

- Basic for Loop: Conditional (Boolean) Expression

- Look out for code that uses logical expressions like this:

```
for (int x = 0; (((x < 10) && (y-- > 2)) || x == 3); x++) {}
```

- The preceding code is legal, but the following is not:

```
for (int x = 0; (x > 5), (y < 2); x++) {} // too many expressions
```

- Basic for Loop: Iteration Expression

- After each execution of the body of the for loop, the iteration expression is executed.

```
for (int x = 0; x < 1; x++) {
    // body code that doesn't change the value of x
}
```


The Enhanced for Loop (for Arrays)

- The enhanced for loop, new to Java 5, is a specialized for loop that simplifies looping through an array or a collection.

```
int [] a = {1,2,3,4};
for(int x = 0; x < a.length; x++) // basic for loop
    System.out.print(a[x]);

for(int n : a) // enhanced for loop
    System.out.print(n);
```

- More formally, let's describe the enhanced for as follows:

for(declaration : expression)

The two pieces of the for statement are

1. **declaration** The newly declared block variable, of a type compatible with the elements of the array you are accessing. This variable will be available within the for block, and its value will be the same as the current array element.
2. **expression** This must evaluate to the array you want to loop through. This could be an array variable or a method call that returns an array. The array can be any type: primitives, objects, even arrays of arrays.

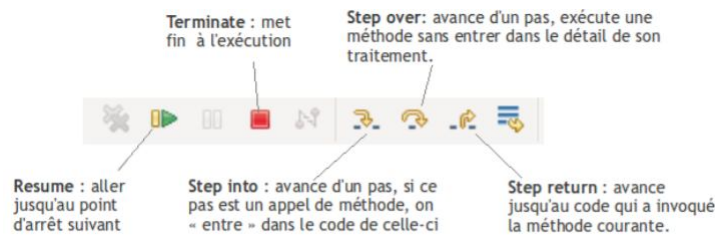
Débogage de programmes Java

Débogage d'applications Java avec Eclipse

Introduction à Java

- Eclipse dispose d'un debugger offrant de nombreuses possibilités qui permettent d'avoir une vue précise de l'exécution d'un programme en n'importe quel point de celui-ci.

- Vous pouvez exécuter une application en mode debugger
 - En choisissant **Debug As→Java Application** après un clic droit sur le nom du fichier de la classe dans la zone de paquetage
 - En utilisant le menu **Run→Debug...** puis configurer comme pour l'exécution d'une application
 - En cliquant sur le bouton représentant une espèce de scarabée (« **bug** ») dans la barre de boutons. Une fenêtre doit vous proposer de passer dans la perspective « Debug ».



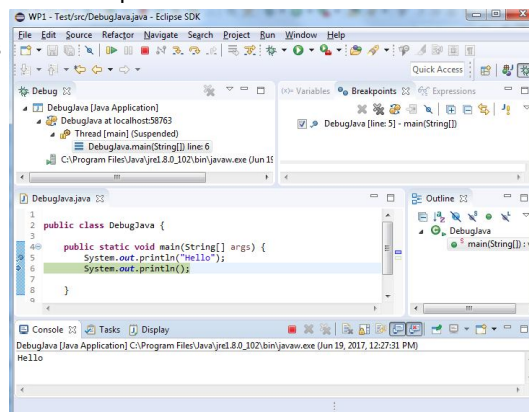
M.Romdhani, Mars 2019

35

Le perspective « Debug »

Introduction à Java

- The debug perspective offers additional views that can be used to troubleshoot an application like Breakpoints, Variables, Debug, Console etc. When a Java program is started in the debug mode, users are prompted to switch to the debug perspective.
- **Debug view** – Visualizes call stack and provides operations on that.
- **Breakpoints view** – Shows all the breakpoints.
- **Variables/Expression view** – Shows the declared variables and their values.
- **Display view** – Allows to inspect the value of a variable, expression or selected text during debugging.



M.Romdhani, Mars 2019

Raccourcis utiles pour le débogage

- Try to memorize the following function keys. They will greatly speed up your debugging by avoiding constantly having to use your mouse.

- CTRL-SHIFT-B – Toggle Breakpoint
- F5 – Step Into
- F6 – Step Over
- F7 – Step Return
- F8 – Run until next breakpoint is reached



Shortcut	Toolbar	Description
F5 (Step Into)		Steps into the call
F6 (Step Over)		Steps over the call
F7 (Step Return)		Steps out to the caller
F8 (Resume)		Resumes the execution
Ctrl + R (Run to Line)		Run to the line number of the current caret position
Drop to Frame		Rerun a part of your program
Shift + F5 (Use Step Filters)		Skipping the packages for Step into
Ctrl + F5 / Ctrl + Alt + Click		Step Into Selection

Watchpoints, Exception Breakpoints, Conditional Breakpoints

- **Watchpoints** - A watchpoint is a special breakpoint that stops the execution of an application whenever the value of a given expression/field changes, without specifying where it might occur. User can specify by **Breakpoint Properties...** if they want the execution to stop when the watch expression is **Accessed**, **Modified** or both
- **Exception Breakpoints** – An exception breakpoint is specified for thrown exception using **Add Java Exception Breakpoint**
- **Condition Breakpoints** – Eclipse users can create conditions to restrict the activation of breakpoints.

