

Java - Initiation

Sources et Références

- <http://docs.oracle.com/javase/8/docs/>
- JAVA 8 – Les fondamentaux du langage Java – Edition ENI
- Jean-Michel Doudoux :
http://www.jmdoudoux.fr/accueil_java.htm

James Gosling et versions de Java

- JDK Alpha and Beta (1995)
- JDK 1.0 (Janvier 1996)
- JDK 1.1 (Février 1997)
- J2SE 1.2 (Décembre 1998)
- J2SE 1.3 (Mai 2000)
- J2SE 1.4 (Février 2002)
- J2SE 5.0 (Septembre 2004)
- Java SE 6 (Décembre 2006)
- Java SE 7 (Juillet 2011)
- Java SE 8 (Mars 2014)
- Java SE 9 (Septembre 2017)



Objectifs du langage Java

- Simple, Objet Oriented and Familiar
- Robust and Secure
- Architecture Neutral and Portable
- High Performance
- Interpreted, Threaded, and Dynamic
- A noter : familier par sa parenté avec le langage C++

Code Robuste

- Ramasse miettes ou gestionnaire de la mémoire :

Contrairement à l'allocation des objets, leur dé-allocation n'est pas à la charge du développeur

- Fortement typé :

Pas d'erreur à l'exécution due à une erreur de type

- Généricité :

Vérification statique, à la compilation, du bon typage

- Exceptions :

Mécanisme de traitements des erreurs, une application ne devrait pas s'arrêter à la suite d'une erreur (ou toutes les erreurs possibles devraient être prises en compte)

Code Portable

Le source Java

Num.java

```
public class Num {  
    ...  
}
```

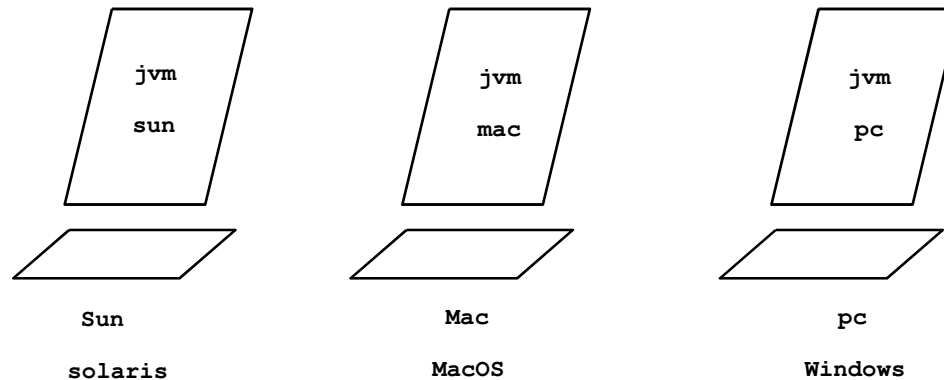
1) **compilation**

Le fichier compilé

Num.class

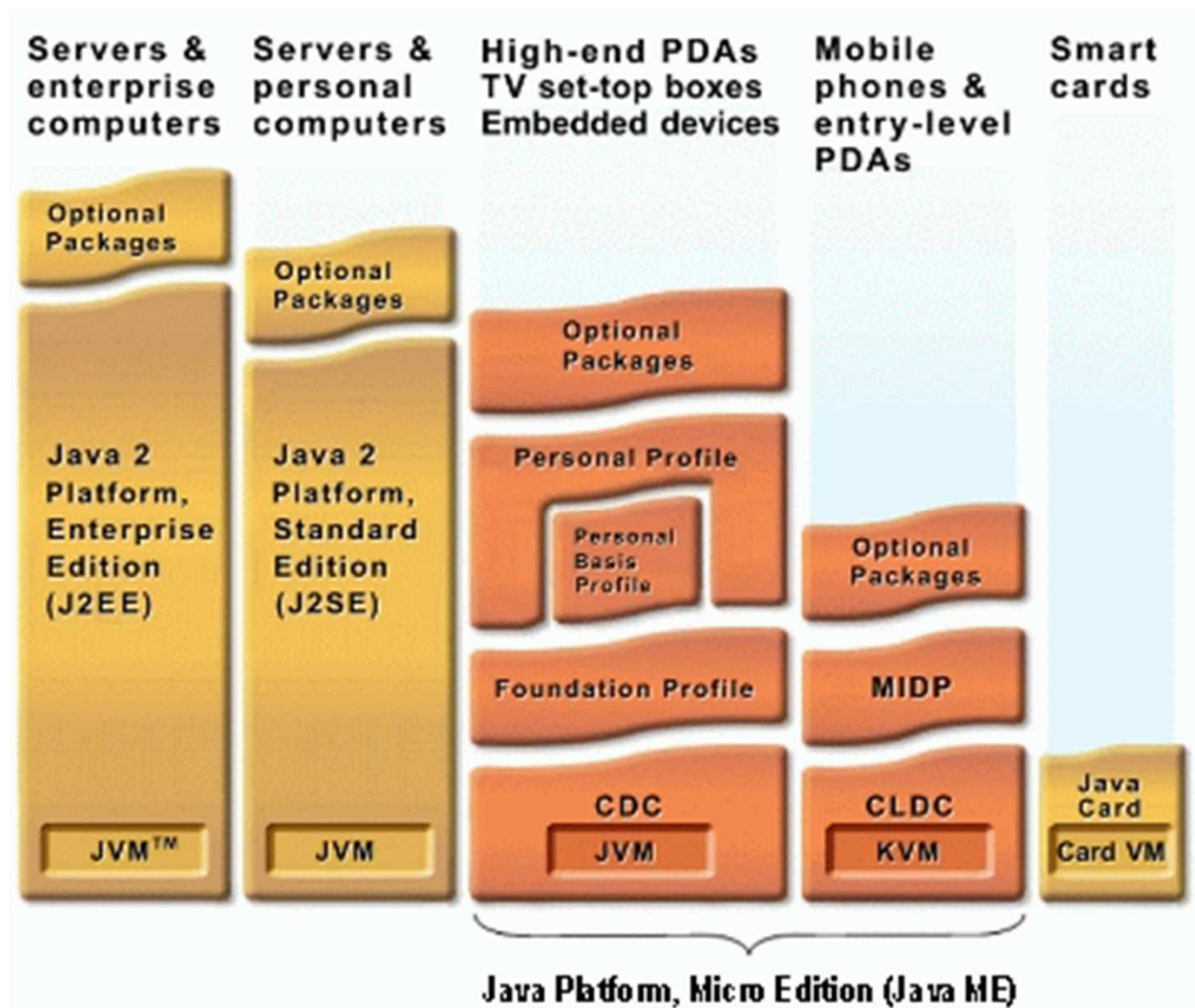
```
1100 1010 1111 1110 1011 1010 1011 1110  
0000 0011 0001 1101 .....  
.....
```

Lecture locale ou distante du fichier

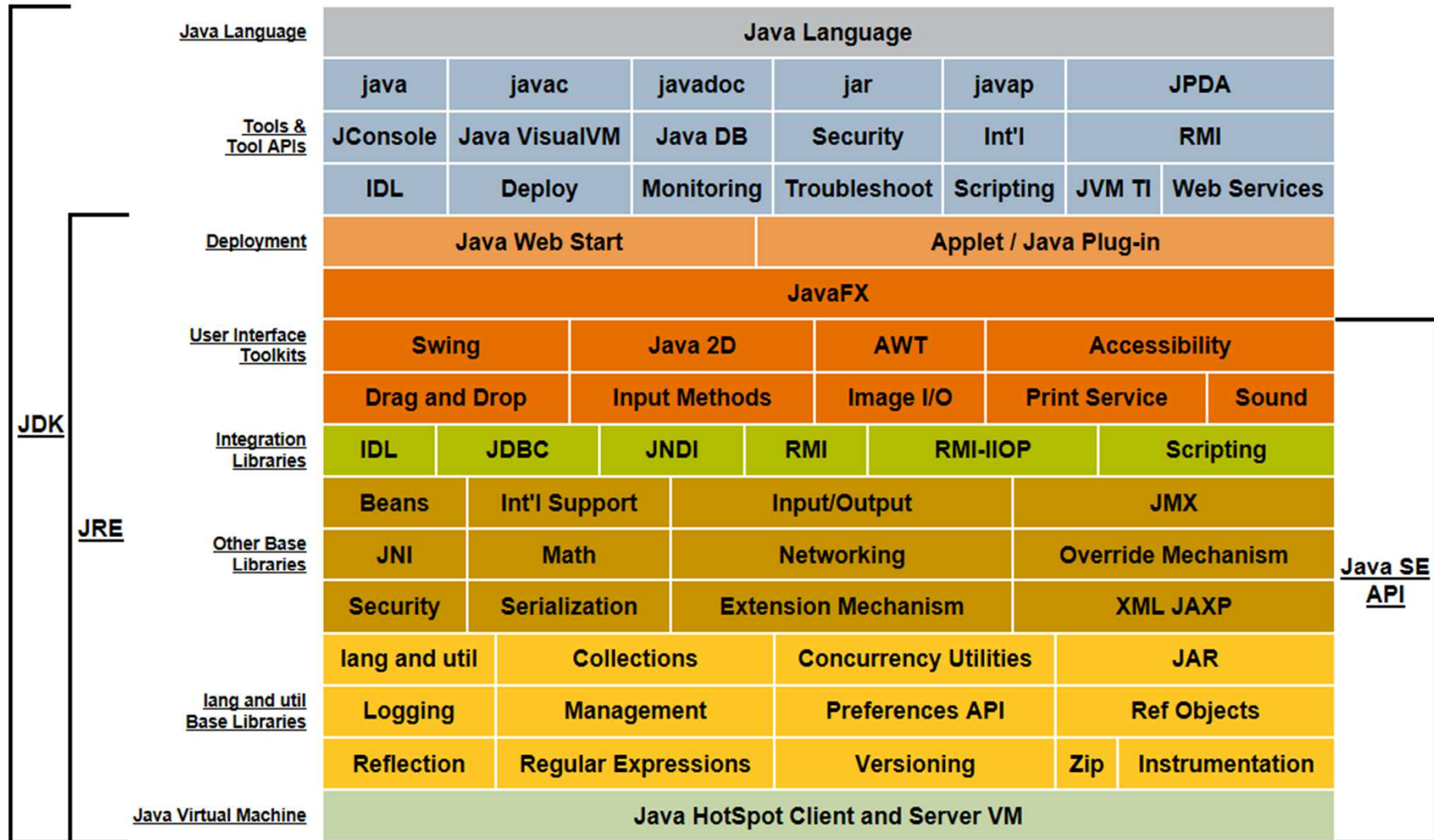


2) **interprétation
/ exécution**

Java EE, SE, ME



Composantes de Java SE 8



Outils Java

Installation du JDK 8

<https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

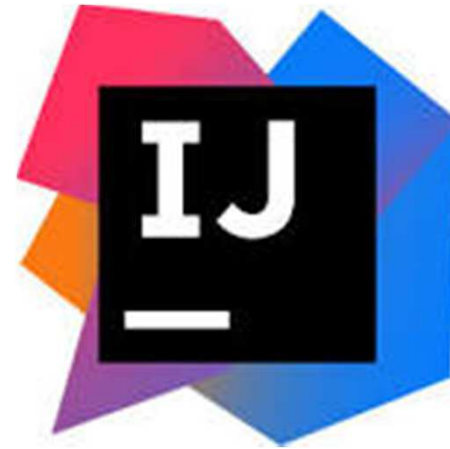
- javac : compilation de fichier(s) Java
- java : exécution de code Java, interprète du code binaire
- javap : decompilateur
- jar : création de fichiers jar
- javadoc : generateur de documentations (au format HTML)
- appletviewer : interprète des applets
- etc.

Environnement de Dev

Installation de l'IDE

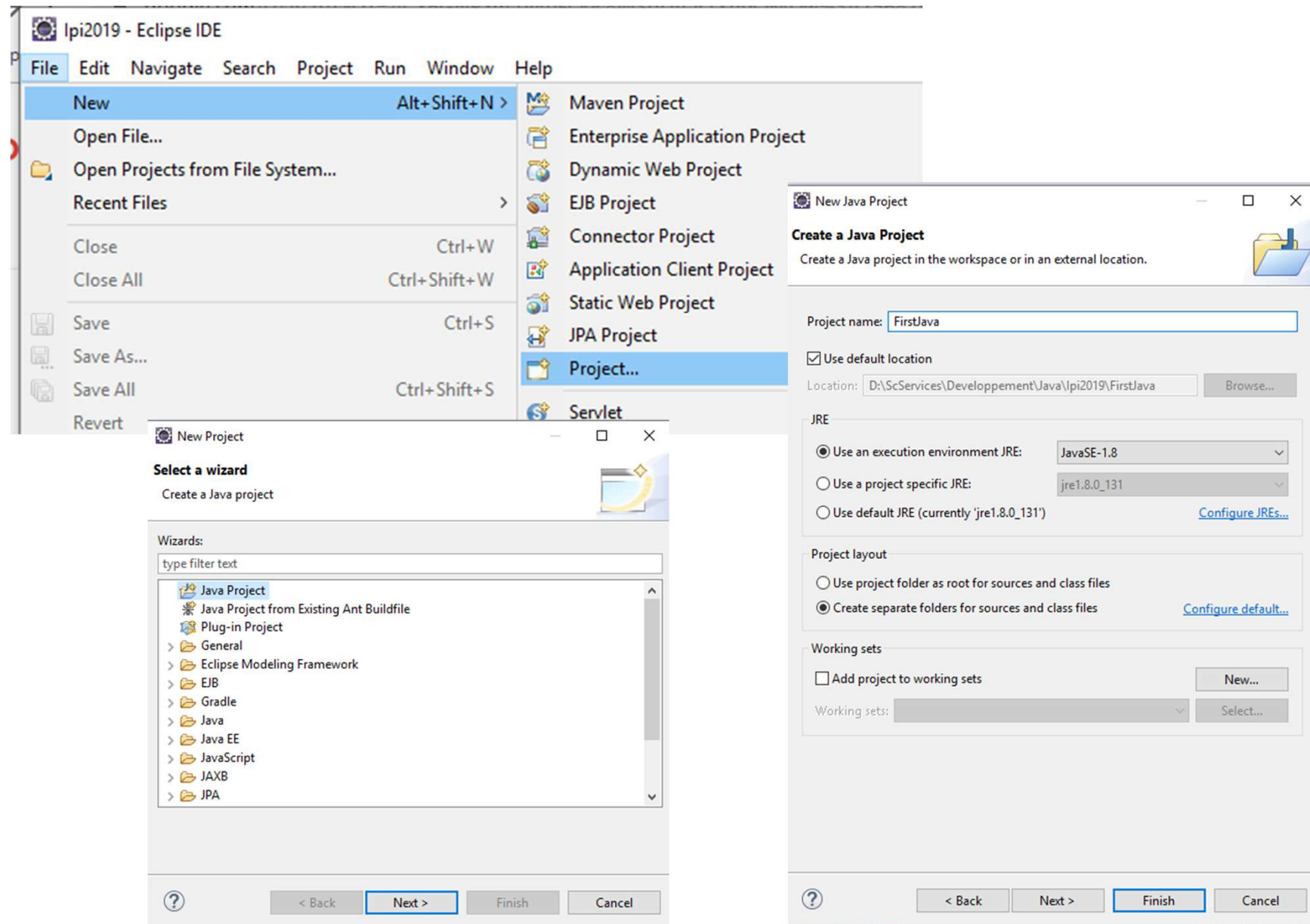


Eclipse

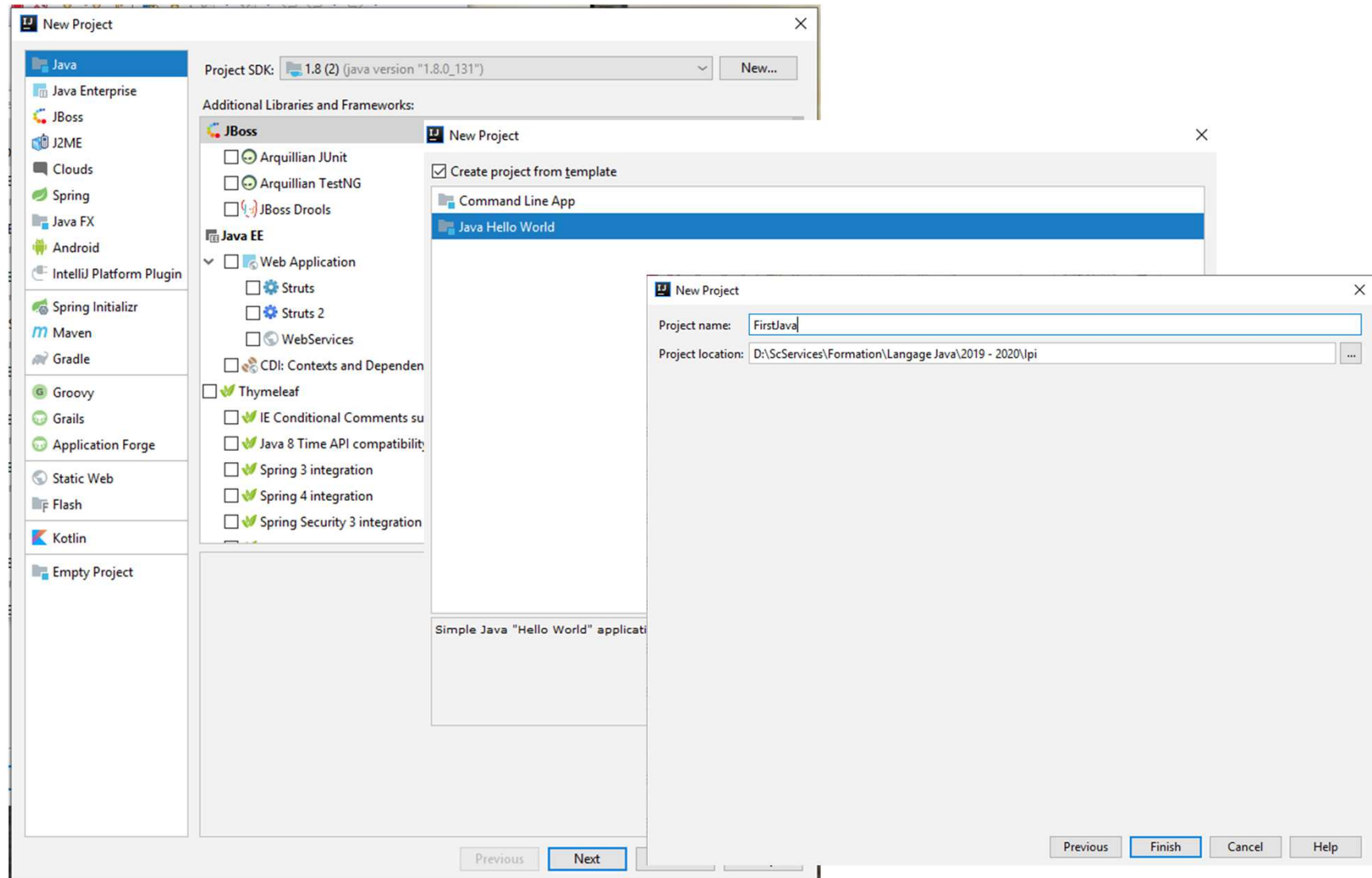


IntelliJ

Création d'un projet sous Eclipse



Création d'un projet sous IntelliJ



La main

```
public class Main {  
  
    public static void main(String[] args) {  
        System.out.println("Hello World!");  
    }  
}
```

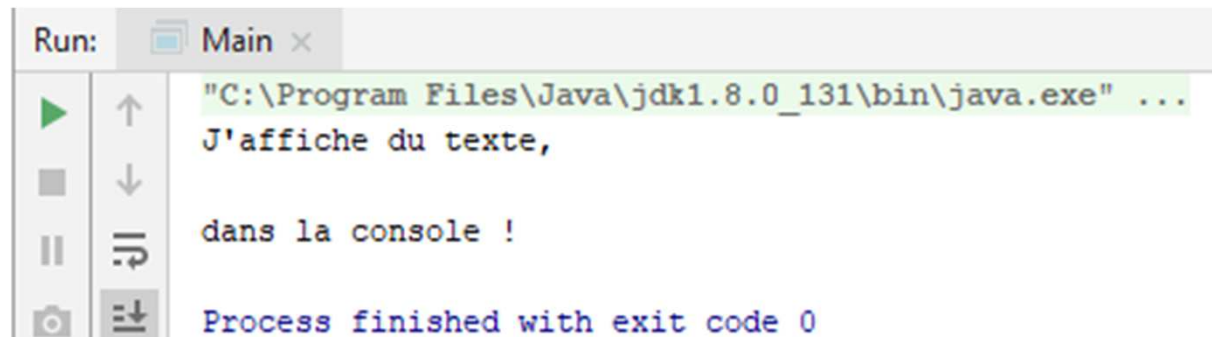
Affichage dans la console

```
public class Main {  
  
    public static void main(String[] args) {  
  
        System.out.print("J'affiche du texte, ");  
        System.out.print(" ");  
        System.out.print("dans la console !");  
    }  
}
```

Run:	Main x	
▶	↑	"C:\Program Files\Java\jdk1.8.0_131\bin\java.exe" ...
■	↓	J'affiche du texte, dans la console !
		Process finished with exit code 0

Affichage dans la console II

```
public class Main {  
  
    public static void main(String[] args) {  
  
        System.out.println("J'affiche du texte, ");  
        System.out.println(" ");  
        System.out.println("dans la console !");  
    }  
}
```



```
Run: Main x  
"C:\Program Files\Java\jdk1.8.0_131\bin\java.exe" ...  
J'affiche du texte,  
  
dans la console !  
Process finished with exit code 0
```

Création de variables

- En Java, on manipule trois choses :
 - Des types primitifs (Java en compte 8)

```
double temperature = 32.5;
```

- Des objets. Rappel : Un objet est une instance d'une classe

```
Date date1 = new Date();
```

- Des tableaux. Les tableaux peuvent contenir des types primitifs ou des objets.

```
int[] tableau1 = new int[5];
```

```
int tableau2[] = new int[5];
```

```
int tableau3[] = {1,2,3,4,5};
```

```
String[] tableau4 = new String[5];
```


Les Types Primitifs

Type	Taille	Fourchette
boolean	1 bit	true ou false
byte	8 bits	-128 à 127
short	16 bits	-32768 à 32767
char	16 bits	\u0000 à \uFFFF
int	32 bits	-2147483648 à 2147483647
float	32 bits	1.401e-045 à 3.40282e+038
double	64 bits	2.22507e-308 à 1.79769e+308
long	64 bits	-9223372036854775808 à 9223372036854775807

Typage et assignation

Typage

```
String testString;  
int testEntier;  
boolean testBoolean;
```

Nom de la variable

Symbole de l'assignation

```
testString = "";  
testEntier = 15;  
testBoolean = true;
```

Valeur de la variable

Méthodes : déclarations

Typage du retour de méthode

```
boolean doSomething(int number, String boop) {  
    return true;  
}
```

Valeur de retour

Paramètres

The diagram illustrates the components of a Java method declaration. The code snippet is: `boolean doSomething(int number, String boop) { return true; }`. Annotations include:

- Typage du retour de méthode**: Points to the `boolean` return type.
- Valeur de retour**: Points to the `return true;` statement.
- Paramètres**: Points to the parameter list `(int number, String boop)`.

Règles de portée

- Une variable n'est accessible qu'au sein de son bloc de déclaration
- Corolaire : une variable est accessible au sein d'un bloc situé à l'intérieur de son bloc de déclaration

Règles de portée : exemples

```
String s;
```

```
void doSomething() {  
    s = "booop";  
}
```

```
void doSomethingElse() {  
    s = "bip";  
}
```

```
void doSomething() {  
    String s = "booop";  
}
```

```
void doSomethingElse() {  
    s = "bip";  
}
```

Cannot resolve symbol 's'

La saisie utilisateur

```
Scanner sc = new Scanner(System.in);

System.out.println("Quel est votre numéro favori ?");
int favoriteNumber = sc.nextInt();
System.out.println("Quel est héros ?");
String favoriteHero = sc.next();

System.out.println("Votre numéro favori est :");
System.out.println(favoriteNumber);
System.out.println("Votre héros préféré est :");
System.out.println(favoriteHero);
```

Exercices

- Demander à l'utilisateur son nom, prénom, email, téléphone
- Afficher les valeurs saisies

Les opérateurs logiques

● Opérateur boolean

- && : « Et »
- || : « Ou »

```
boolean isTrue = true;  
boolean isFalse = false;  
System.out.println(isTrue || isFalse);  
  
true
```

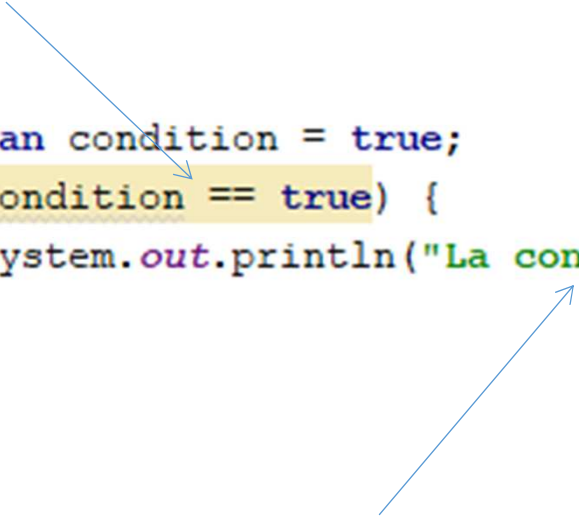
● Opérateur d'égalité

- == : égalité
- != : différent
- <= : inférieur ou égal
- >= : supérieur ou égal
- < : strictement inférieur
- > : strictement supérieur

Les test de conditions : if

La condition

```
boolean condition = true;  
if (condition == true) {  
    System.out.println("La condition est vraie");  
}
```



Instructions exécutées si la condition est vraie

Les test de conditions : if else

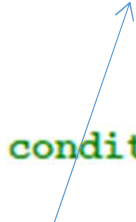
```
boolean condition = true;  
if (condition == true) {  
    System.out.println("La condition est vraie");  
}  
else {  
    System.out.println("La condition est fausse");  
}
```



Instructions exécutées si la condition est fausse

Les test de conditions : if else if

```
boolean condition1 = false;
boolean condition2 = true;
if (condition1 == true) {
    System.out.println("La condition 1 est vraie");
}
else if (condition2){
    System.out.println("La condition 2 est vraie");
}
else {
    System.out.println("Les conditions 1 et 2 sont fausses");
}
```



Instructions exécutées si la première condition
Est fausse, et que la seconde est vraie

Exercice

- Créer une méthode qui demande à l'utilisateur d'entrer une valeur et affiche si cette valeur est positive, ou négative ou égale à 0.
- Créer une méthode qui demande à l'utilisateur d'entrer une note. Afficher « ajourné » si la note est inférieure à 8, « rattrapage » entre 8 et 10, « admis » au-dessus de 10. positive ou négative ou égale à 0.
- Créer une méthode qui demande à l'utilisateur de saisir :
 - Deux valeurs a et b, de type int
 - Un opérateur op de type string, vérifier qu'il s'agit d'une des valeurs suivante : +, -, /, *Afficher le résultat de l'opération a op b.

Les test de conditions : le swtich

Variable testée

```
Scanner sc = new Scanner(System.in);  
String s = sc.next();  
  
switch(s) {  
    case "boop":  
        System.out.println("Le mot choisi est boop !");  
        break;  
    case "bip":  
        System.out.println("Le mot choisi est bip !");  
        break;  
    default:  
        System.out.println("Le mot choisi n'est ni boop, ni bip !");  
}
```

Scénarios testés

Signe la fin des instructions

Instructions réalisées si aucun des autres scénarios n'est vérifié

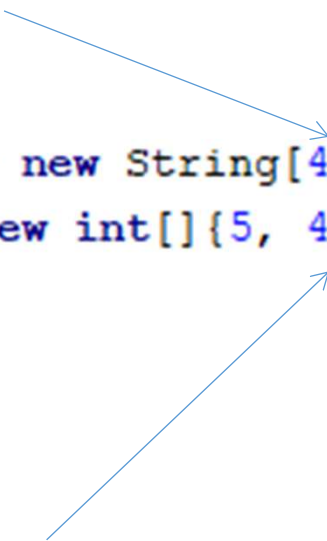
Exercices

- Refaire l'exercice de la calculatrice avec un switch

Déclaration d'un tableau

Taille du tableau

```
String[] strings = new String[4];  
int[] integers = new int[]{5, 4, 3, 2, 1};
```

A diagram with two blue arrows. One arrow starts at the text 'Taille du tableau' and points to the number '4' in the first line of code. The other arrow starts at the text 'Valeurs des différentes cases du tableau' and points to the list of numbers '{5, 4, 3, 2, 1}' in the second line of code.

Valeurs des différentes cases du tableau

La boucle for

```
int[] integers = new int[]{5, 4, 3, 2, 1};  
for(int i = 0; i < integers.length; i++) {  
    System.out.print("i " + i + " : ");  
    System.out.println(integers[i]);  
}
```


La boucle while

```
int i = 0;
while (i < integers.length) {
    System.out.print("i " + i + " : ");
    System.out.println(integers[i]);
}
```

La boucle do while

```
Scanner sc = new Scanner(System.in);
String string;
do {
    System.out.println("Voulez vous quitter la boucle ? (veuillez répondre par O ou N)");
    string = sc.next();
}
while (!string.toLowerCase().equals("o"));
```

Exercices

● Créer un programme en Java qui permet de gérer un tableau de taille fixe. Le programme doit utiliser les fonctions suivantes :

1. une fonction remplir qui remplit un tableau avec des entiers.
2. une fonction afficher qui affiche les éléments d'un tableau.
3. une fonction somme qui retourne la somme des éléments du tableau.
4. une fonction max qui renvoie le max de deux entiers
5. en utilisant la fonction précédente, écrire une fonction max qui renvoie le max des éléments du tableau.

Exercices

6. une fonction recherche qui permet de chercher un élément dans le tableau en affichant le message « Élément trouvé » si l'élément existe ou « Élément non trouvé » sinon.
7. une fonction qui permet d'ajouter un élément dans une position donnée.
8. une fonction qui permet de supprimer un élément donné par sa position.
9. Une fonction tri qui permet de trier le tableau en utilisant l'algo de tri bulle optimisé :
https://fr.wikipedia.org/wiki/Tri_%C3%A0_bulles

Classe

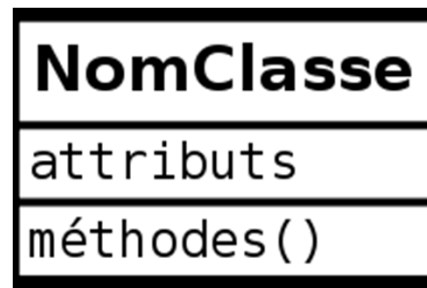
- Une classe peut être vu comme un générateur d'objets de même structure (variables d'instance) et de même comportements (méthodes) = **(Un moule)**
- Une classe se déclare ainsi :

```
public class App { }
```

Dans cet exemple, cette classe se nomme App

Une classe en UML

- Une classe possède un nom, des attributs et des méthodes. (*Caractéristiques et comportement*)



- Attention à bien respecter la convention d'écriture (CamelCase)

Exemple de la classe Personne

- Une classe = des attributs + des méthodes
- Pour chaque attribut de la classe, un accesseur et un mutateur peuvent être défini :

```
private String nom;  
public String getNom() {  
    return nom;  
}  
public void setNom(String nom) {  
    this.nom = nom;  
}
```

Constructeur

- Un constructeur implicite dans tout objet, sans paramètre

```
public Personne() {  
    return nom;  
}
```

- Possibilité de définir autant de constructeur que voulu (*surcharge*) (Obligation d'exprimer le constructeur implicite si nécessaire)

```
public Personne(String nom) {  
    this.nom = nom;  
}
```


Exemple de la classe Personne

```
package fr.tnf.business;
```

```
public class Personne {
```

```
    private String nom;
```

```
    private String prenom;
```

```
    public String getNom() { return nom; }
```

```
    public void setNom(String nom) { this.nom = nom; }
```

```
    public String getPrenom() { return prenom; }
```

```
    public void setPrenom(String prenom) { this.prenom =  
prenom; }  
}
```

Attributs privés de la classe Personne

getNom() est le nom d'une méthode

prenom est le paramètre de la méthode setPrenom

Héritage

- Une classe peut hériter d'une autre classe :

public class Auteur **extends** Personne

- En Java, toutes les classes (de manière implicite) héritent de la classe `java.lang.Object`

.String toString();

.String getClass();

.Object clone();

.boolean equals(Object);

.void finalize();

Classe Abstraite

- Une classe abstraite est une classe dont l'implémentation de certaines méthodes est absente
- L'implémentation est laissée à la responsabilité des sous-classes
- Elle n'est pas instanciable (aucun objet de la classe abstraite *Animal* de ne peut exister)
- Exemple : `java.lang.Number` :
- `public abstract class Number`

Classe Abstraite

```
public abstract class ClasseAbstraite {  
  
    abstract void a();  
    int i; // éventuellement données d'instance  
    static int j; // ou variables de classe  
    static void b(){  
        System.out.println("Hello, it's me");  
    }  
}
```