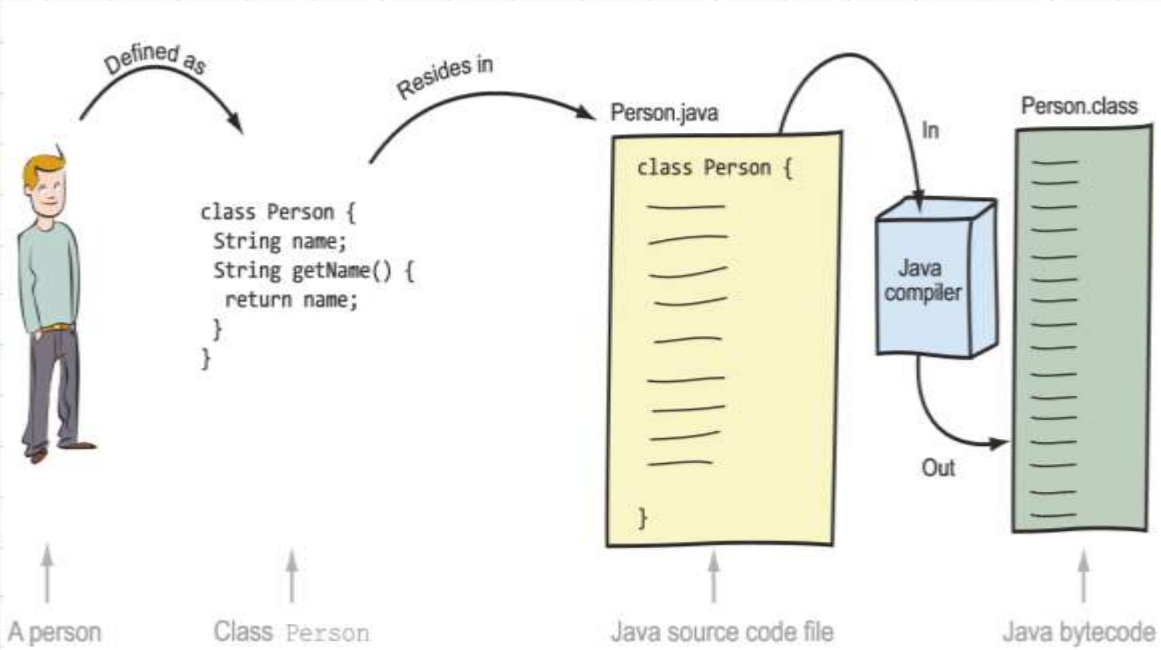
PROGRAMMATION ORIENTÉE OBJET CHAP4:
LE LANGAGE JAVA

L’environnement JAVA

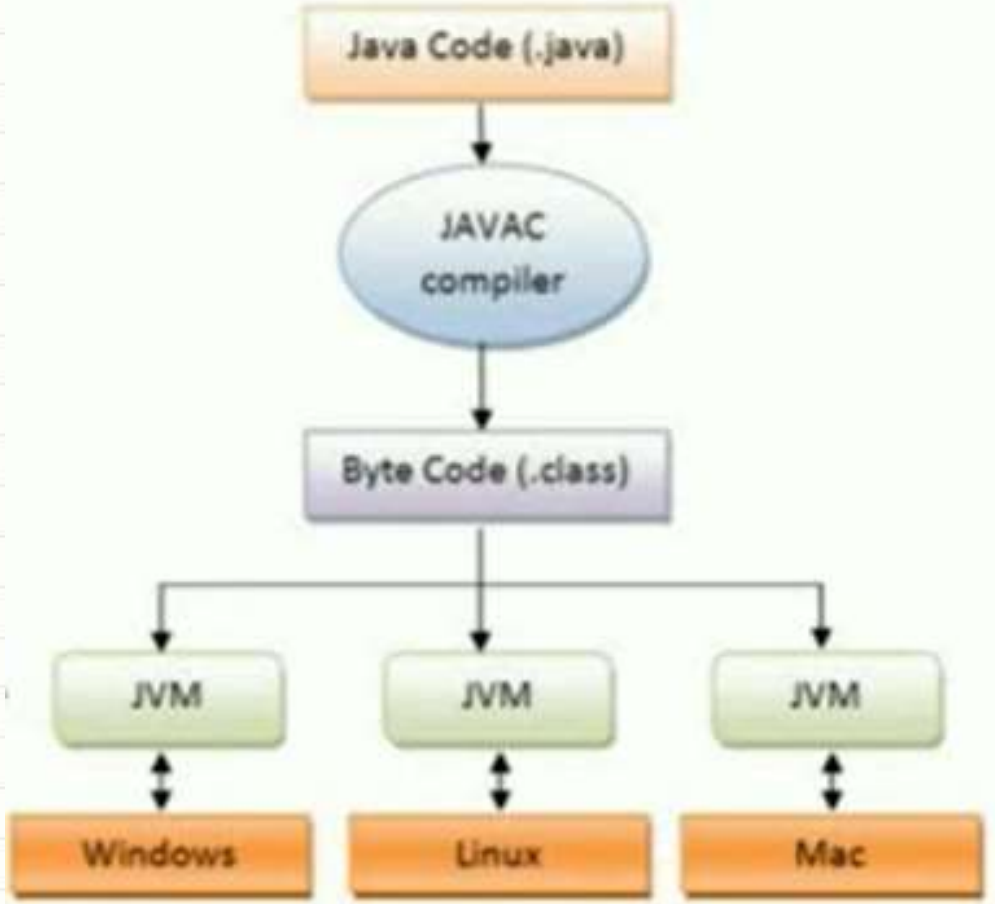


Source: C. Thomas WU. « An Introduction to Object-Oriented Programming with Java ». 2010

- Un programmeur Java écrit son code source, sous la forme de classes, dans des fichiers dont l’extension est **.java**.
- Ce code source est alors compilé par le compilateur **javac** en un langage appelé bytecode et enregistre le résultat dans un fichier dont l’extension est **.class.**

L’environnement JAVA

- Le bytecode ainsi obtenu n’est pas directement utilisable.
- Il doit être interprété par la **machine virtuelle de Java (JVM)** qui transforme alors le code compile en code machine comprehensible par le système d’exploitation.
- C’est la raison pour laquelle **Java est un langage portable**: le byte code reste le même quelque soit l’environnement d’exécution.



Java : "Compile once, run everywhere"

L'environnement Java

- Considérons par exemple ce fichier « *A3.java* » situé dans le répertoire « *DossierJava* »

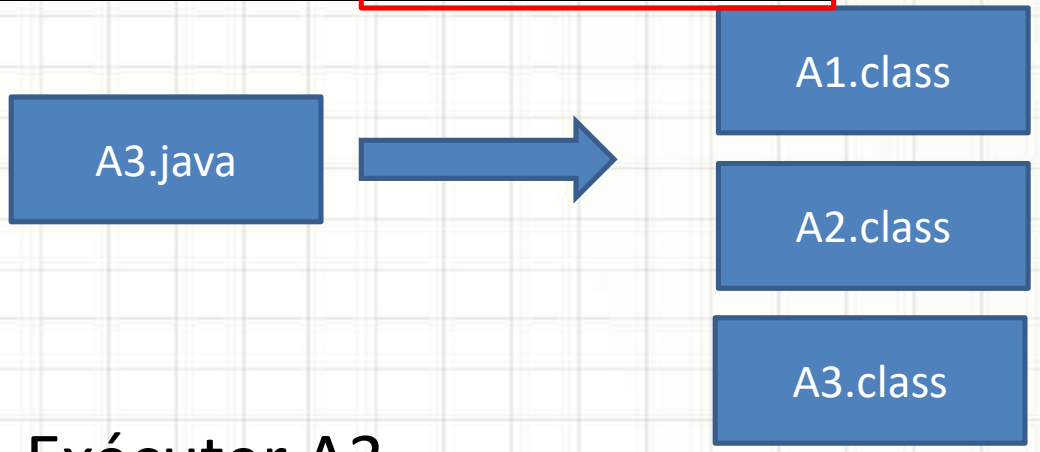
```
class A1{
    static void afficher(){
        System.out.println("La classe A1");
    }
}
class A2{
    static void afficher(){
        System.out.println("La classe A2");
    }
}
public class A3{

    public static void main(String[] args){
        System.out.println("La classe A3");
        A1.afficher();
        A2.afficher();

    }
}
```

- Définir le chemin du JDK

```
\DossierJava>set path=C:\Program Files\Java\jdk-9.0.4\bin
```

- Compiler A3.java

```
DossierJava>javac A3.java
```



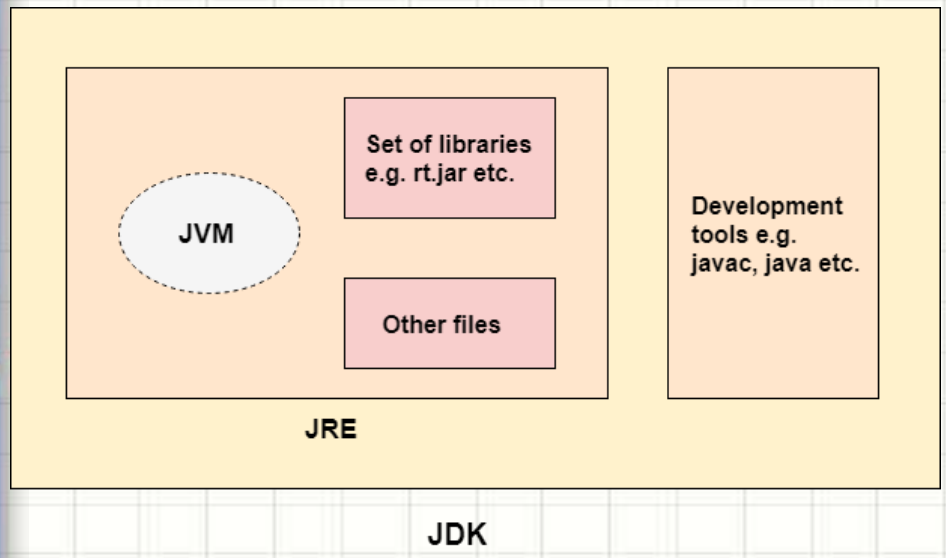
- Exécuter A3

```
DossierJava>java A3
La classe A3
La classe A1
La classe A2
```

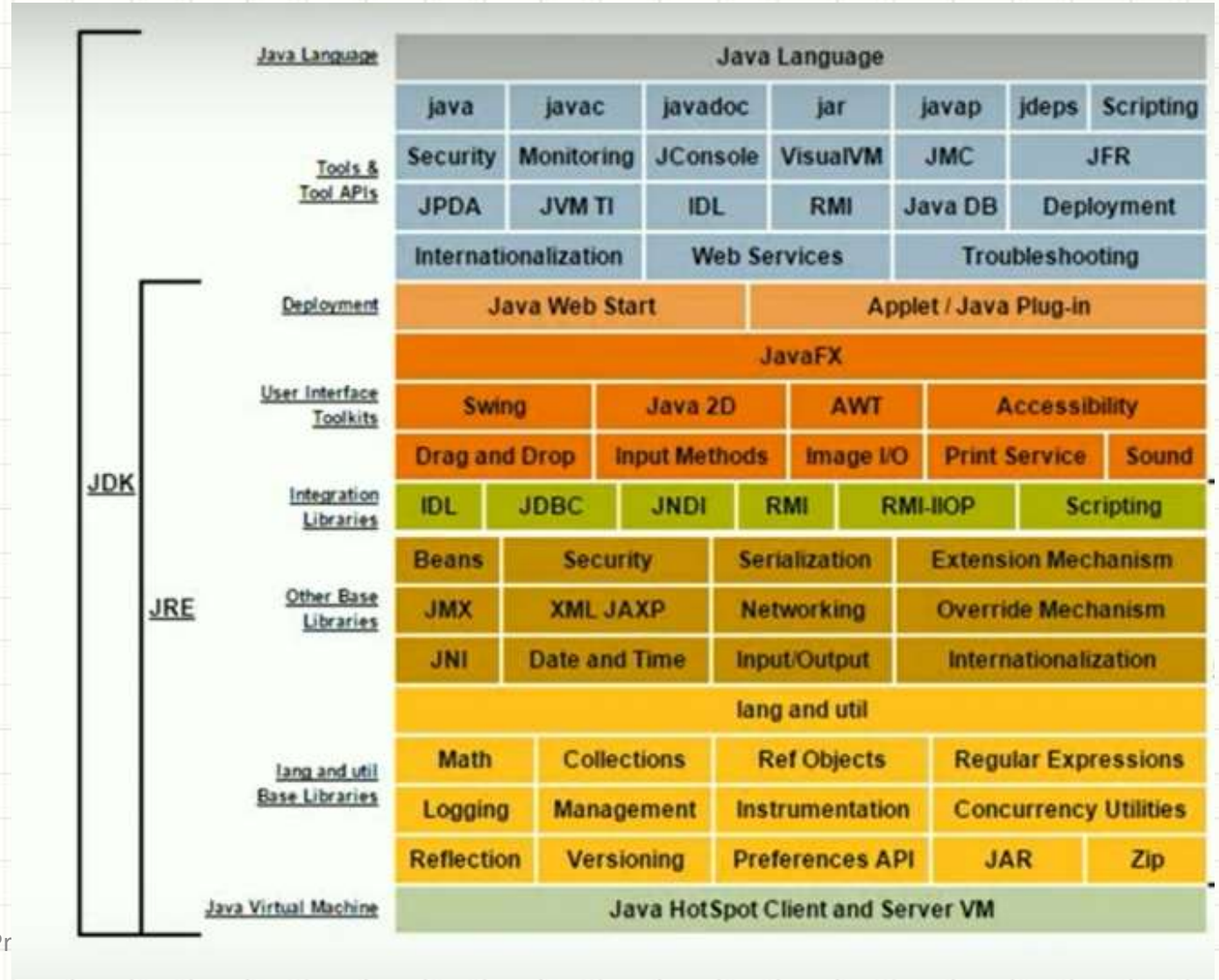
L’environnement Java

- Java est un **langage interprété**, ce qui signifie qu’un programme compilé n’est pas directement exécutable par le système d’exploitation mais il doit être interprété par un autre programme, qu’on appelle interpréteur. **L’interpréteur** est **la machine virtuelle Java (JVM)**.
- Le **Java Runtime Environment (JRE)** se compose d'une machine virtuelle, de bibliothèques logicielles utilisées par les programmes Java et d'un plugin pour permettre l'exécution de ces programmes depuis les navigateurs web.
- En 2009, Sun Microsystems est racheté par Oracle Corporation qui fournit les outils de développement Java SE (Standard Edition) contenus dans le **Java Development Kit (JDK)**.

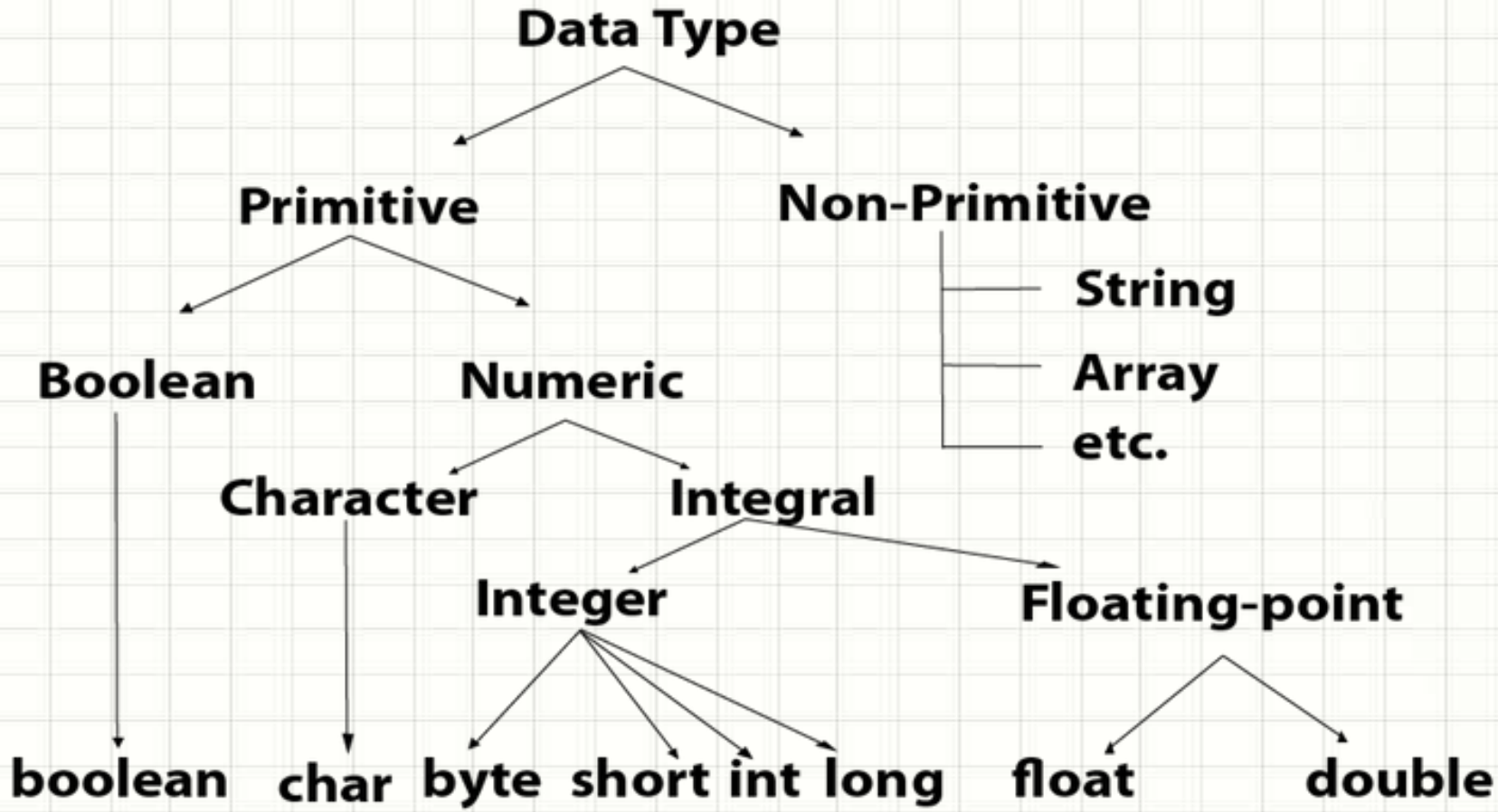
L’environnement Java



Source: https://www.javatpoint.com



Les types de données



Source: https://www.javatpoint.com/java-data-types

Les types de données

- Les types primitifs:

Type	Classe éq.	Valeurs	Portée
`boolean`	`Boolean`	`true` ou `false`	N/A
`byte`	`Byte`	entier signé	{-128..128}
`char`	`Character`	caractère	{\uoooo..\uFFFF}
`short`	`Short`	entier signé	{-32768..32767}
`int`	`Integer`	entier signé	{-2147483648..2147483647}
`long`	`Long`	entier signé	$\{-2^{31}..2^{31}-1\}$
`float`	`Float`	réel signé	$\{-3,4028234^{38}..3,4028234^{38}\}$ $\{-1,40239846^{-45}..1,40239846^{-45}\}$
`double`	`Double`	réel signé	$\{-1,797693134^{308}..1,797693134^{308}\}$ $\{-4,94065645^{-324}..4,94065645^{-324}\}$

LES TYPES NON PRIMITIFS : LES TABLEAUX ET LES CHAÎNES DE CARACTÈRES

Les tableaux

- Un tableau est une structure regroupant des éléments de même type dans un espace contigu en mémoire.
- Un tableau d'entiers par exemple est déclaré ainsi:

```
int[] monTab;
int []monTab;
int monTab[];
```

- Un tableau est un **objet** et a toujours une **taille fixe** qui doit être précisée lors de son instanciation; avant l'affectation de valeurs à ses indices:

```
int[] monTab = new int[20];
```

- Les indices d'un tableau de taille n varient de 0 à n-1.
- La taille d'un tableau est stockée dans une variable **length** accessible ainsi: **monTab.length**

Les tableaux

```
public class MainTab {
    public static void main(String[] args)
{
        //déclaration et instanciation
        int[] a = new int[5];
        //initialisation
        a[0]=10;
        a[1]=20;
        a[2]=70;
        a[3]=40;
        a[4]=50;
        //affichage du contenu du tableau
        for(int i=0;i<a.length;i++)
            System.out.println(a[i]);
    }
}
```

```
10
20
70
40
50
```

```
public class MainTab {
    public static void main(String[] args) {
        /* déclaration, instanciation et
        initialisation */
        int[] a = {10, 20, 70, 40, 50};
        //affichage du contenu du tableau
        for(int i=0;i<a.length;i++)
            System.out.println(a[i]);
    }
}
```

```
public class MainTab {
    public static void main(String[] args) {
        /* déclaration, instanciation et
initialisation */
        int[] a = {10, 20, 70, 40, 50};
        //affichage du contenu du tableau
        for (int element:a)
            System.out.println(element);
    }
}
```

Les tableaux
Passage en paramètre à une méthode

```
class TestTab {
    static void min(int tab[]) {
        int min = tab[0];
        for (int i = 1; i < tab.length; i++)
            if (min > tab[i]) min = tab[i];
        System.out.println(min);
    }
}
public class MainTab {
    public static void main(String[] args) {
        int[] monTab = {12, 3, 45, 22};
        // appel de fonction
        TestTab.min(monTab);
    }
}
```

3

Les tableaux: les tableaux anonymes

```
class TestTab {
    static void afficheTab(int[] tab) {
        for (int i=0; i<tab.length; i++)
            System.out.println(tab[i]);
    }
}
public class MainTab {
    public static void main(String[] args) {
        TestTab.afficheTab(new int[]{12, 3, 45, 22});
    }
}
```

```
12
3
45
22
```

```
class TestTabAnonyme {
    static int[] get() {
        return new int[]{12, 3, 45, 22};
    }
}
public class Main {
    public static void main(String[] args) {
        int[] monTab = TestTabAnonyme.get();

        for (int i=0; i<monTab.length; i++)
            System.out.println(monTab[i]);
    }
}
```

Les tableaux:
Les tableaux à 2 dimensions

- Un tableau (de double par exemple) à deux dimensions en Java est déclaré comme suit:

```
double[][] monTab;
double [][]monTab;
double monTab[][];
double []monTab[];
```

- Un tableau de 2 lignes, 3 colonnes est instancié comme suit:

```
double[][] monTab = new double[2][3];
```

```
public class Main {
    public static void main(String[] args) {
        double[][] monTab = {{2.7, 3.4, 2.1},
{5.8, 8.2, 1.9}};
        for (int i=0; i<2; i++)
        {
            for(int j=0; j<3; j++)
                System.out.print(monTab[i][j]+" ");

            System.out.println();
        }

    }
}
```

```
2.7 3.4 2.1
5.8 8.2 1.9
```

Les tableaux

- La classe java.util.Arrays regroupe un ensemble de méthodes pour le remplissage, le tri, la comparaison, la recherche la recopie, … des tableaux.
- Exemple: **public static void** arraycopy(Object src, **int** srcPos,Object dest, **int** destPos, **int** length)

```
public class Main {
    public static void main(String[] args) {
        char[] tabSrc = { 'm', 'o', 'n', ' ', 'p', 'r', 'o',
                'g', 'r', 'a', 'm', 'm', 'e' };
        //declaring a destination array
        char[] tabDst = new char[9];
        System.arraycopy(tabSrc, 4, tabDst, 0, 9);
        System.out.println(String.valueOf(tabDst));
    }
}
```

programme

Les chaînes de caractères

- Les chaînes de caractères en Java ne sont pas considérées comme un type primitif ou comme un tableau.
- On utilise une classe particulière nomée String fournie par le package java.lang.
- Les variables de type String ont les caractéristiques suivantes :
  - Leur valeur ne peut être modifiée.
  - On peut utiliser l'opérateur + pour concaténer 2 chaines de caractères.

```
char[] ch={'j','a','v','a'};
String s=new String(ch);
⇔
String s="java";
```

```
String s1="Hello";
String s2="World";
String s3=s1+" "+s2;
```

Les chaînes de caractères

- Un ensemble de méthodes de la classe **java.lang.String** permettent d'effectuer des operations ou des tests sur une chaîne de caractères.

```
public class MainChaines {
    public static void main(String[] args) {
        String txt = "Hello";
        int len = txt.length();
        System.out.println("la longueur de la chaîne est: "+len);
        System.out.println(txt.toUpperCase());
        System.out.println(txt.toLowerCase());

        String txt2 = txt.concat(" World");
        System.out.println(txt2);
        System.out.println(txt2.indexOf("World"));
    }
}
```

```
la longueur de la chaîne est: 5
HELLO
hello
Hello World
6
```

Lecture des entrées clavier en Java

- Pour lire les entrées du clavier, on fait appel à un objet de type Scanner.
- La classe Scanner est une classe du package java.util
- Lorsqu'on écrit *System.out.println("Hello World");* on applique println() sur la sortie standard ***System.out***.
- Pour la lecture des entrées clavier, nous instancions l'objet Scanner en utilisant l'entrée standard ***System.in***.

```
import java.util.Scanner;

public class MainClass {
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        System.out.println("Entrer la marque de la voiture");
        String marqueVoiture = sc.nextLine();
        System.out.println("La marque saisie est :"+marqueVoiture);
    }
}
```

Entrer la marque de la voiture
Honda
La marque saisie est :Honda

Lecture des entrées clavier en Java

- Plusieurs méthodes sont prévues au niveau de la classe Scanner pour saisir les différents type de données.

Method	Example
nextByte( )	byte b = scanner.nextByte( );
nextDouble( )	double d = scanner.nextDouble( );
nextFloat( )	float f = scanner.nextFloat( );
nextInt( )	int i = scanner.nextInt( );
nextLong( )	long l = scanner.nextLong( );
nextShort( )	short s = scanner.nextShort( );

Lecture des entrées clavier en Java

- Exemple:

```
import java.util.Scanner;

public class MainClass {
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        System.out.println("Entrer l'année et le prix de la voiture");
        int anneeV = sc.nextInt();
        double prixV = sc.nextDouble();
        System.out.println("L'annee: "+anneeV+" Le prix: "+prixV);
    }
}
```

Entrer l'année et le prix de la voiture
2007
25
L'annee: 2007 Le prix: 25.0

Lecture des entrées clavier en Java

- Scanner, ne prévoit pas par contre une méthode pour la saisie du type **char**.
- Pour contourner ce problème, on peut utiliser la méthode **nextLine()** pour la saisie d'une chaîne de caractères et on accède au caractère souhaité grâce à **charAt()**.

```
import java.util.Scanner;

public class MainClass {
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        System.out.println("Entrer la marque de la voiture");
        String marqueV = sc.nextLine();
        char m = marqueV.charAt(0);
        System.out.println("La marque est :"+marqueV+" et elle commence par: " +m);
}}
```

Entrer la marque de la voiture
BMW
La marque est :BMW et elle commence par: B

Lecture des entrées clavier en Java

- Remarque:
  - nextLine() récupère le contenu de toute la ligne saisie et replace la « tête de lecture » au début d'une autre ligne.
  - Lorsqu'on invoque nextInt() ou nextDouble(), … et qu'on veuille par la suite appeler nextLine(), cette dernière ne permettra pas de saisir une chaîne de caractères mais elle videra la ligne utilisée par les autres instructions.
  - Pour contourner ce problème, il faut invoquer nextLine() 2 fois: une pour vider la ligne et une autre fois pour saisir la chaîne entrée au clavier.

Lecture des entrées clavier en Java

- Exercice:
  - Implémenter une classe Personne définie à travers les attributs: nom, prenom, age et sexe (de type char).
  - Ecrire :
    - un constructeur par défaut pour les objets de type Personne.
    - une méthode saisir() qui saisit les attributs d'un objet de type Personne.
    - une méthode afficher() qui affiche les valeurs des attributs.
    - la méthode main() dans une classe MainPersonne dans le même fichier que la classe Personne (fichier: MainPersonne.java).
  - Tester les méthodes saisir() et afficher() de la classe Personne.

La classe Math

- Classe Math du package java.lang offre un ensemble de méthodes qui permettent de faire des calculs basiques tels que la valeur absolue, le cosinus, le sinus, le max …

```
la valeur absolue de -3 = 3
la racine carrée de 25 = 5.0
2^3 = 8.0
l'arrondi de 3.67 = 3
une valeur aléatoire = 0.3343821683583068
le plus grand = 10
le plus petit = 7
PI = 3.141592653589793
```

```
public class MainClass {
    public static void main(String[] args){
        System.out.println(" la valeur absolue de -3 = " + Math.abs(-3));
        System.out.println(" la racine carrée de 25 = " + Math.sqrt(25));
        System.out.println(" 2^3 = " + Math.pow(2,3));
        System.out.println(" l'arrondi de 3.67 = " + Math.round(3.19));
        System.out.println(" une valeur aléatoire = " + Math.random());
        System.out.println(" le plus grand = " + Math.max(5, 10));
        System.out.println(" le plus petit = " + Math.min(7, 14));
        System.out.println(" PI = " + Math.PI);
}}
```