Université Cadi Ayyad
Ecole Supérieure de Technologie – Safi

COURS 1

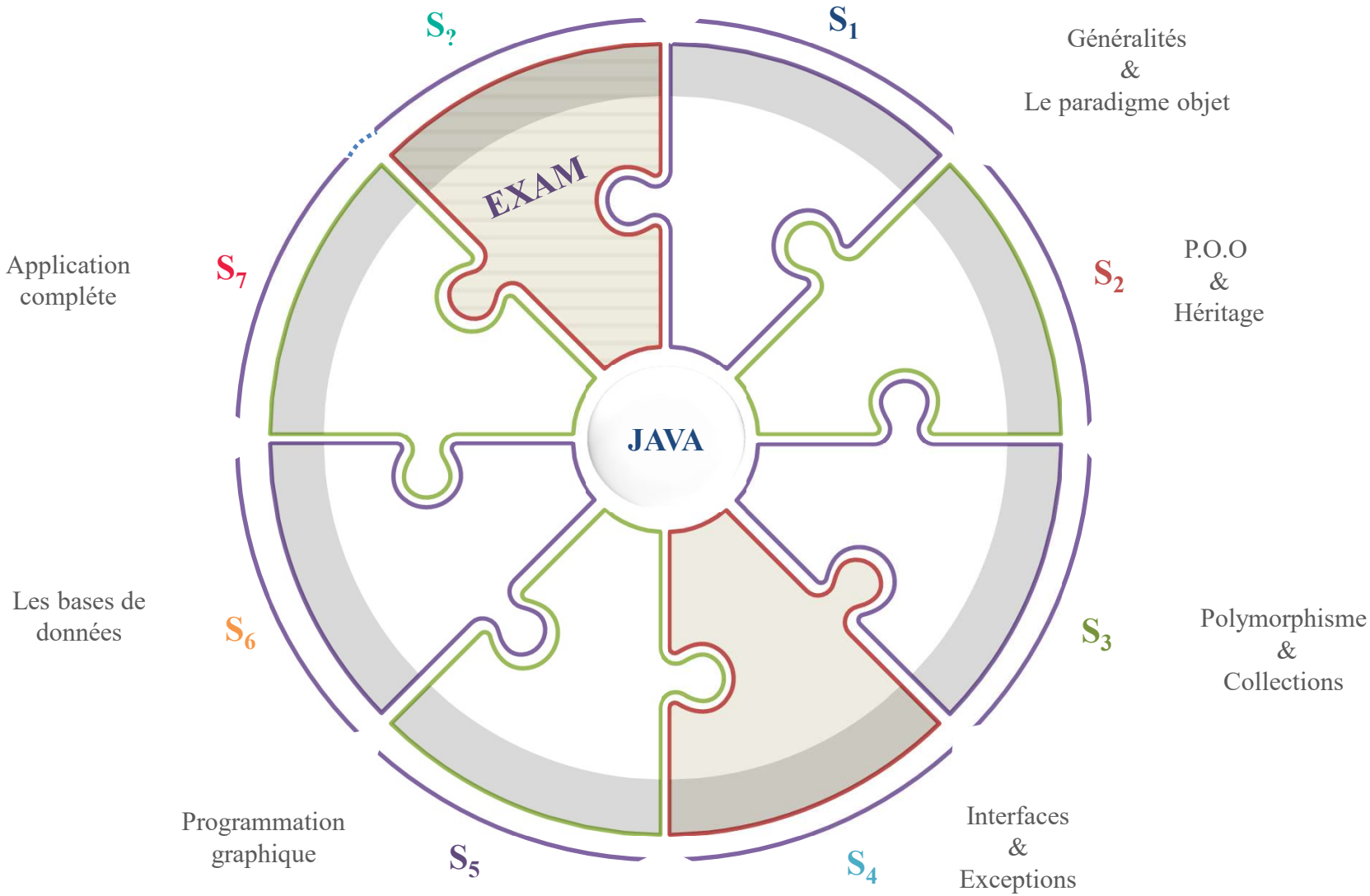
COURS DE LA PROGRAMMATION ORIENTÉE OBJET (JAVA)

Pr. EL ABDELLAOUI SAID

Elabdellaoui.said@yahoo.fr

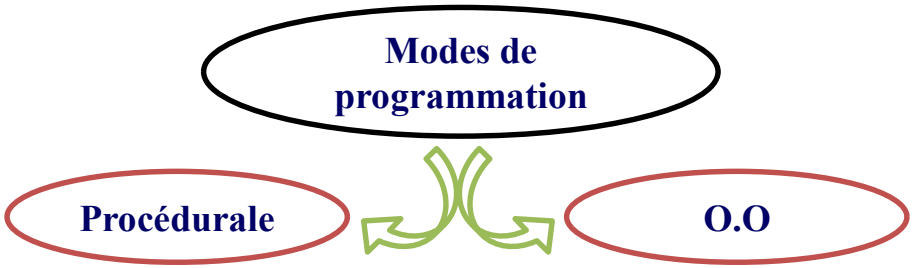
GÉNÉRALITÉS

PLANIFICATION



INTRODUCTION

INTRODUCTION

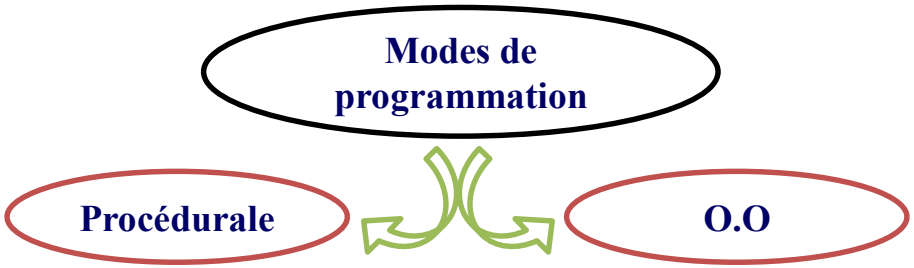


❑ **Procédurale** : Les notions de variables/types de **données** et de **traitement de ces données** étaient séparées :

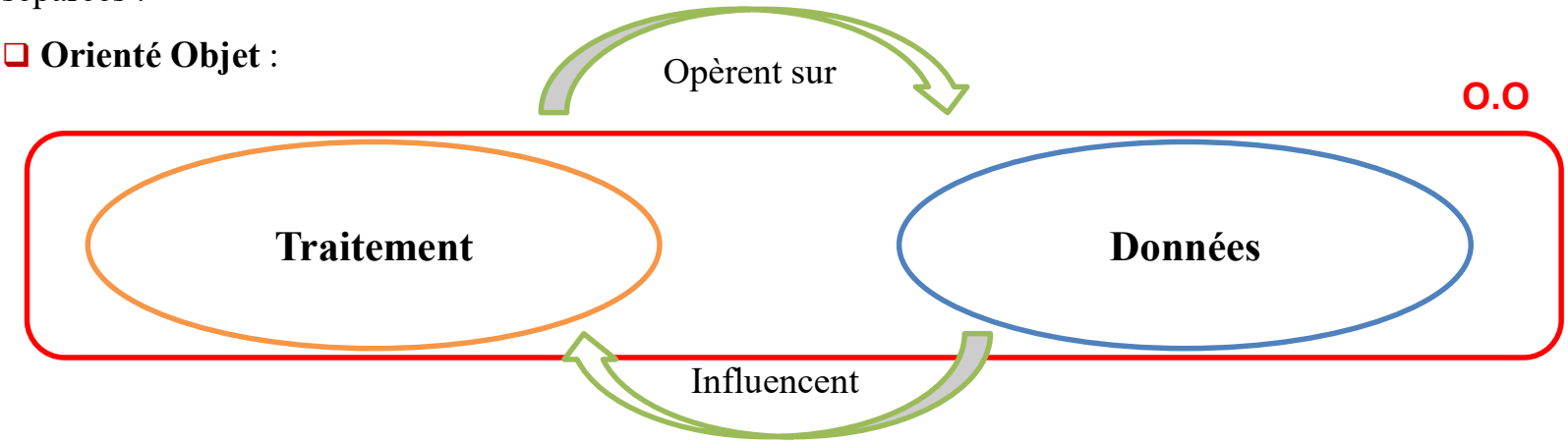
```
    // Méthode !!
   double Surface (double l, double h){
                  return (l × h)
                          }

   main () {
      // Déclarations !!
         double Largeur = 7.0;
         double Hauteur = 2.0 ;
      // Affichage !!
               print (" Surface du rectangle : "+
                        Surface (largeur, hauteur) );

}
```

Syntaxe universelle explicative !!

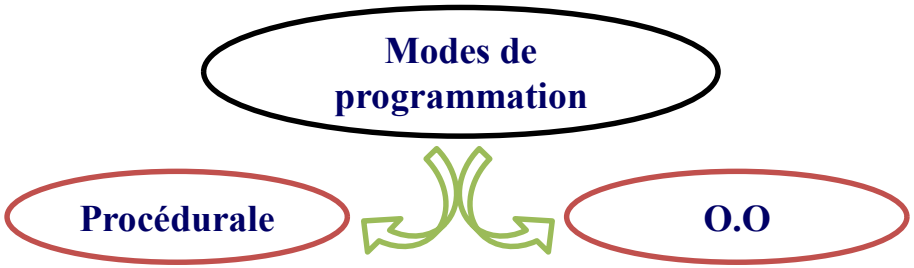
INTRODUCTION



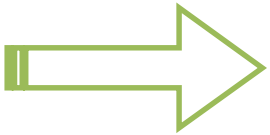
- **Procédurale** : Les notions de variables/types de **données** et de **traitement de ces données** étaient séparées :
- **Orienté Objet** :

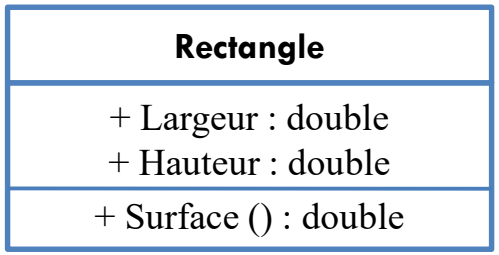


INTRODUCTION




```
Public class Rectangle {
    public static void main (  String[] args   ) {
       // Déclarations !!
          double Largeur = 7.0;
          double Hauteur = 2.0 ;
       // Affichage !!
     System.out.println (" Surface du rectangle : "+
                          Surface (largeur, hauteur) );

                                                  }
      // Méthode !!
     double Surface (double l, double h){
                    return (l × h)
                             }

}
```



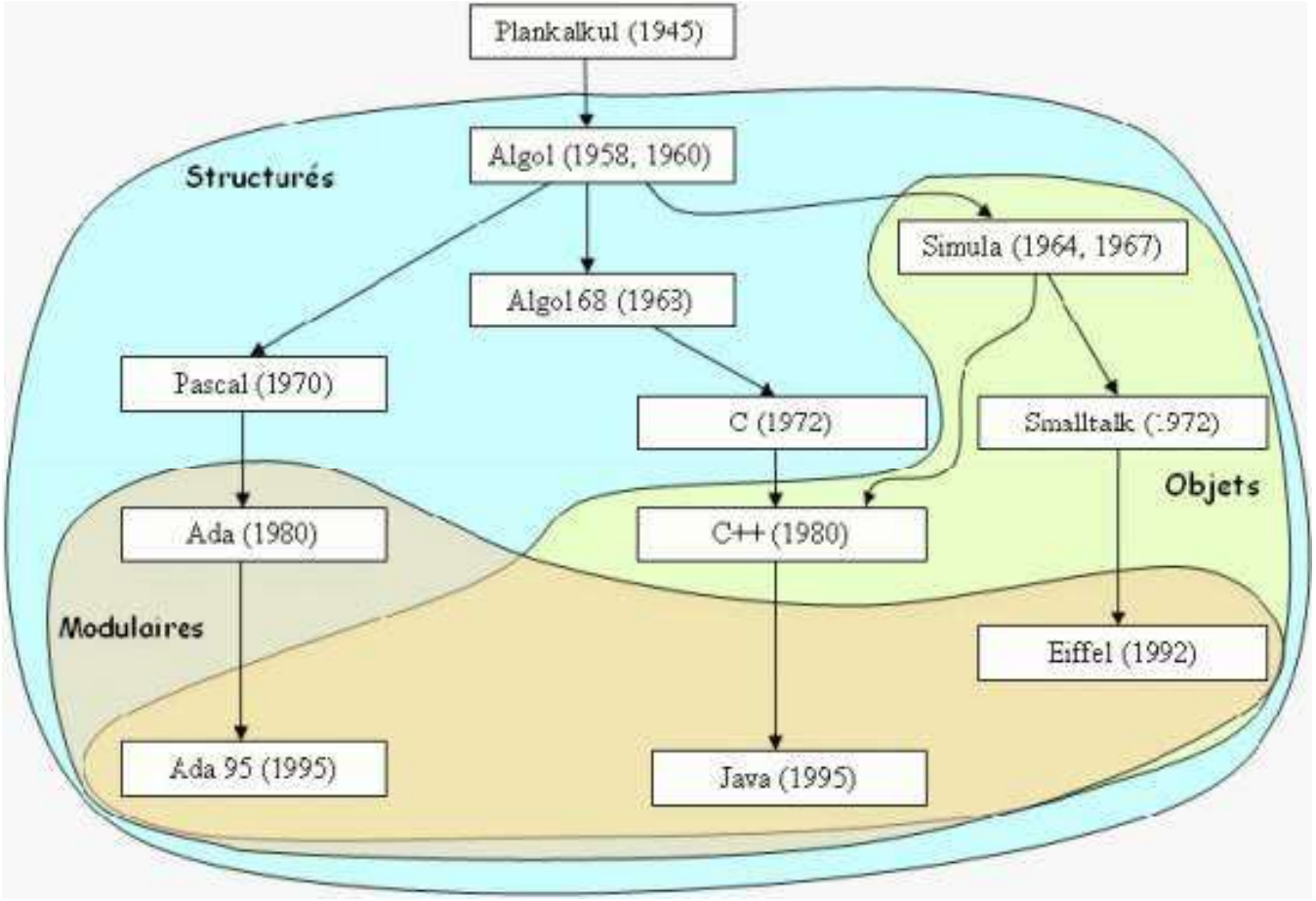


INTRODUCTION

- **Java** utilise les notions usuelles de la programmation orientée objet :
  - Encapsulation
  - Abstraction
  - Héritage
  - Polymorphisme
  - Générésités
  - …….

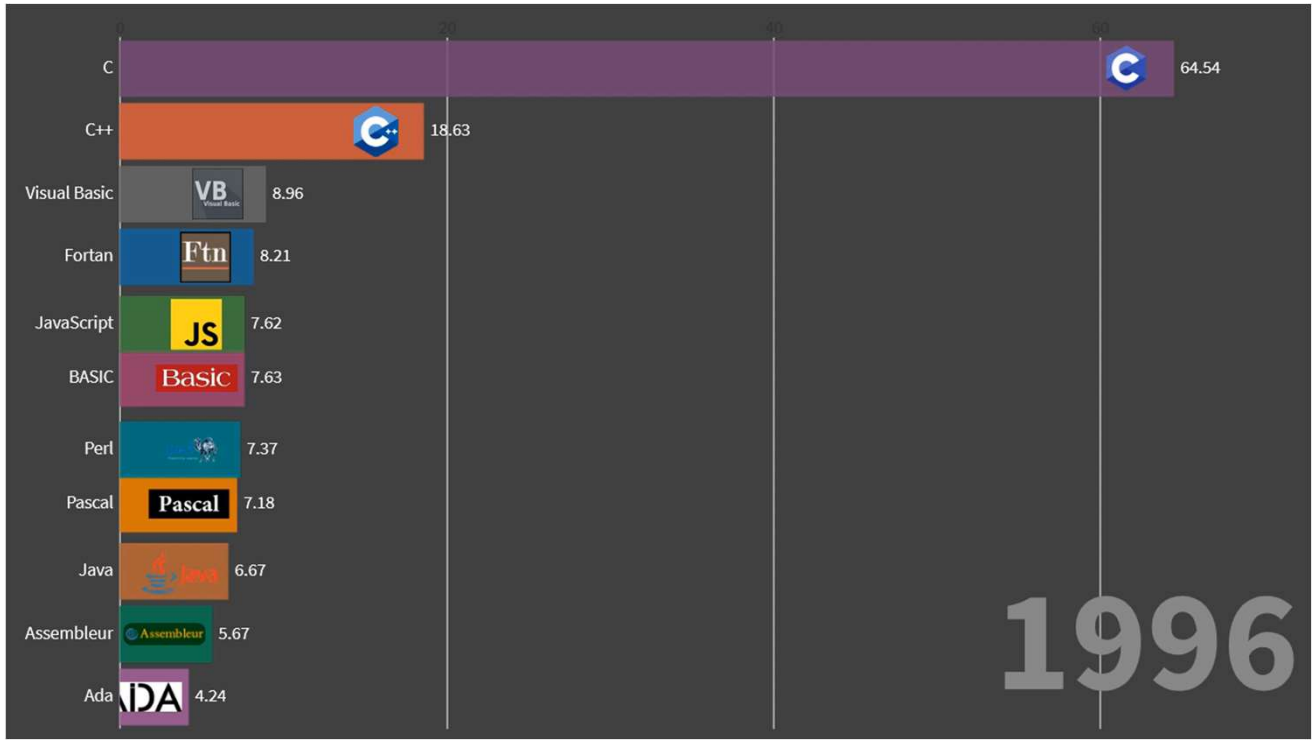
GÉNÉRALITÉS

HISTORIQUE DE JAVA



Différentes langage de programmation

HISTORIQUE DE JAVA



C : 1975-2000
JAVA : 2001-2018
Python : 2019-2021

Evolution de différentes langage de programmation

HISTORIQUE DE JAVA

The Green Project

1991

Lancement

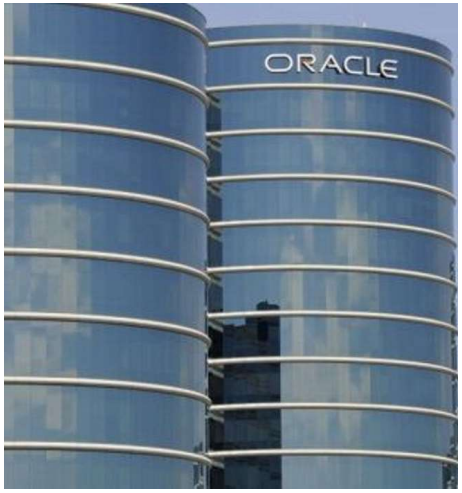
Langage portable **=**

1995

HISTORIQUE DE JAVA

HISTORIQUE DE JAVA

HISTORIQUE DE JAVA



2010

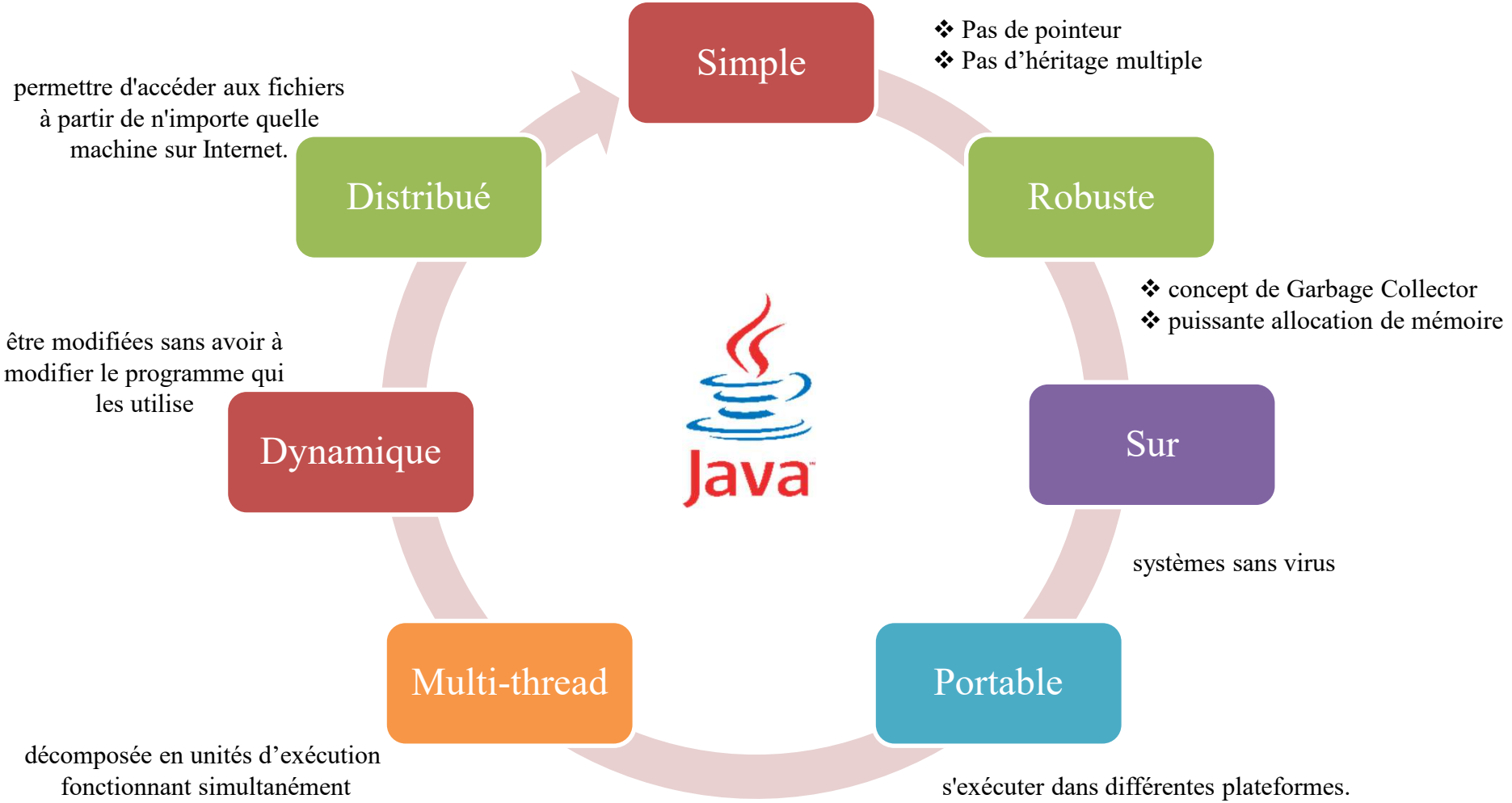
Java

SPARC

HISTORIQUE DE JAVA

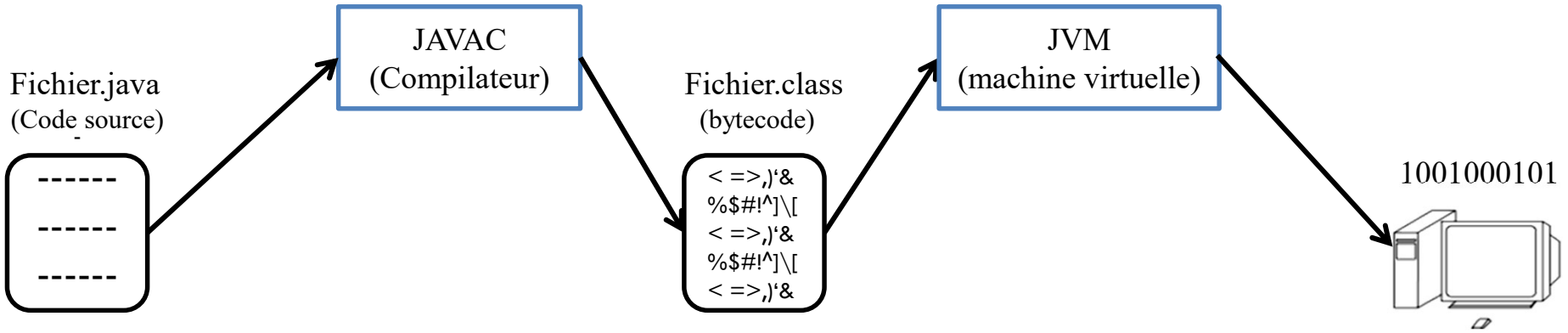
- L'histoire de JAVA a commencée depuis **1991** et lancé **1995** par **SUN Microsystems** sous un projet nommé ("Green Project").
- Son premier nom était **OAK**.
- Langage de programmation inspiré du C++
- Libre depuis novembre 2006
- Racheté par Oracle en 2010
- Un langage accompagné d'un ensemble énorme de bibliothèques standard couvrant de très nombreux domaines.

PRÉSENTATION DE JAVA



PRÉSENTATION DE JAVA

- Il est basé sur une architecture logicielle nécessitant une machine virtuelle java (JVM).



Architecture du langage JAVA

PLATE-FORMES JAVA

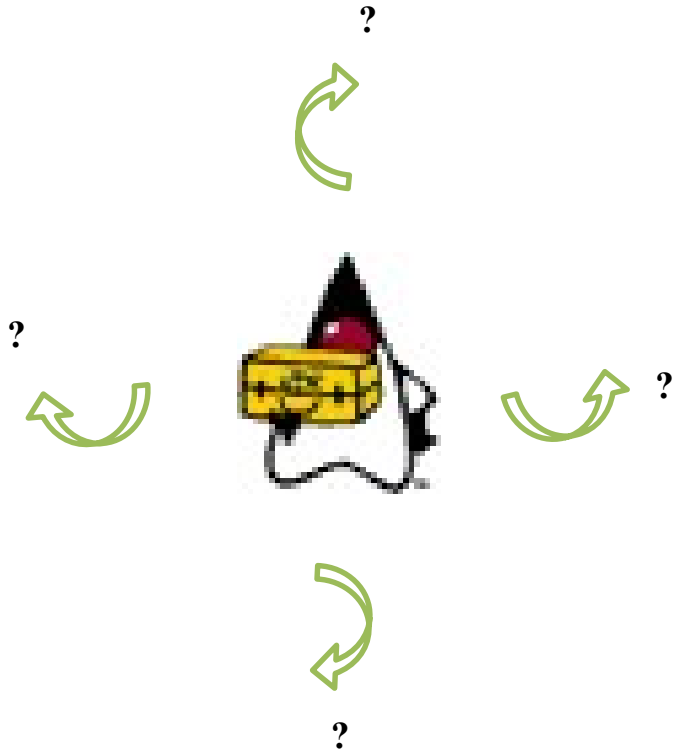
- **Java SE** : "Standard Edition" :

- **Java ME** : "Micro Edition". Edition qui sert à écrire des applications embarquées
  - Ex. : téléphone portable, carte à puce

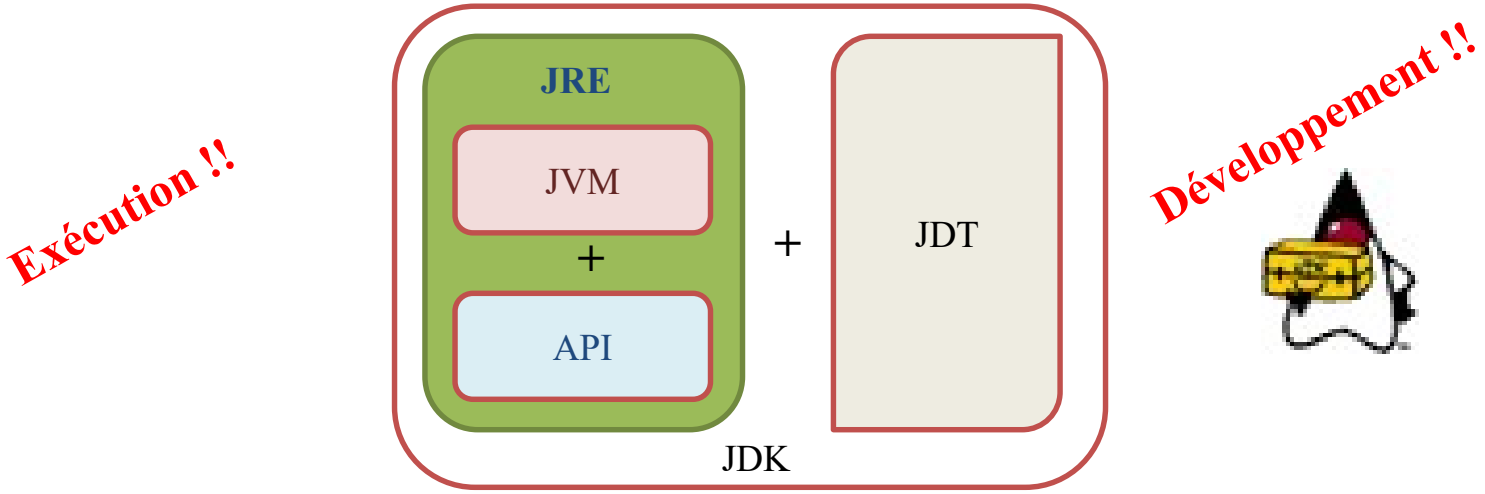
- **Java EE** : "Enterprise Edition". Rajoute certaines API et fonctionnalités pour les entreprises.

- **Java Fx**: Créer des applications Internet riches à l'aide d'une API d'interface utilisateur légère. C'est un ajout récent à la famille des plates-formes Java.

PLATEFORME JAVA - OUTILS -



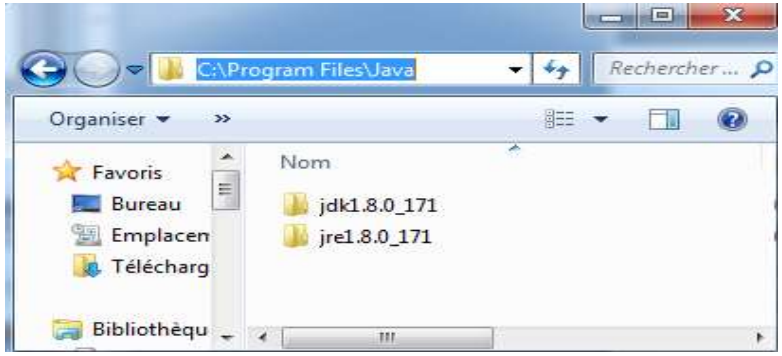
PLATEFORME JAVA



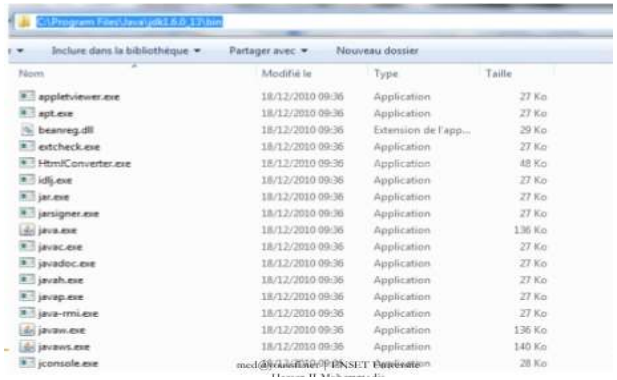
- **JVM** signifie **Java Virtual Machine** permet de traduire **Byte Code** en code de la machine hôte (host machine).
- **JRE** signifie **Java Runtime Environment,** il c… …VM et un certain nombre de bibliothèques **Java** qui aident à exécuter des programmes écrits e…
- **API** : est l'abréviation **Applicat…** …large collection de software composants groupés en bibliothèques appelés …
- **JDK** est l'abréviation de **Java Develo…** …2 composants **JRE** et **JDT.**
- **JDT** est l'abréviation de **Java Dev…** …is fournissent des outils pour le développement d'applications **Java**, tels que **Javac** aide à co… …le code source des programmeurs en **Byte Code.**

PLATE-FORMES JAVA

- Installation :
  - Le JDK peut être téléchargé gratuitement à partir du site de Oracle (www.oracle.com)
  - Exécuter *jdk-8u13-windows-i586-p.exe.* Le JDK sera installé dans le répertoire *c:program files\java* et installe également *jre1.8* dans le même dossier.



- Les programmes nécessaire au développement java sont placés dans le répertoire c:\jdk1.5\bin a savoir:
  - **javac.exe** Compilateur java.
  - **java.exe:** Interpréteur du bytecode java.
  - **appletviewer.exe** Pour tester les applets java.
  - **Jdb.exe** : Débogueur java.
  - **Javap.exe** désassembleur du bytecode.
  - **Javadoc.exe :** Générer la documentation de vos programmes java.



,,

JAVA : ENTRÉES / SORTIES

- L'aspect le plus utile de la classe **System** est les variables qu'elle déclare, qui permettent d'avoir une interaction avec le système. On y trouve les variables *in*, *out* et *err*.
- La variable in représente le flux d'entrée standard du système, alors que la variable out représente le flux de sortie standard. La variable err est le flux d'erreur standard. (les flux vont être détailler prochainement)

  - Pour **afficher** une chaine de caractère :

```
System.out.println("Chaine de caractère") ;
```

  - Pour **afficher** une erreur :

```
System.err.println("Erreur à l'ouverture") ;
```

1ÉRE APPLICATION

SUR MACHINE

NOTRE PREMIER PROGRAMME EN JAVA

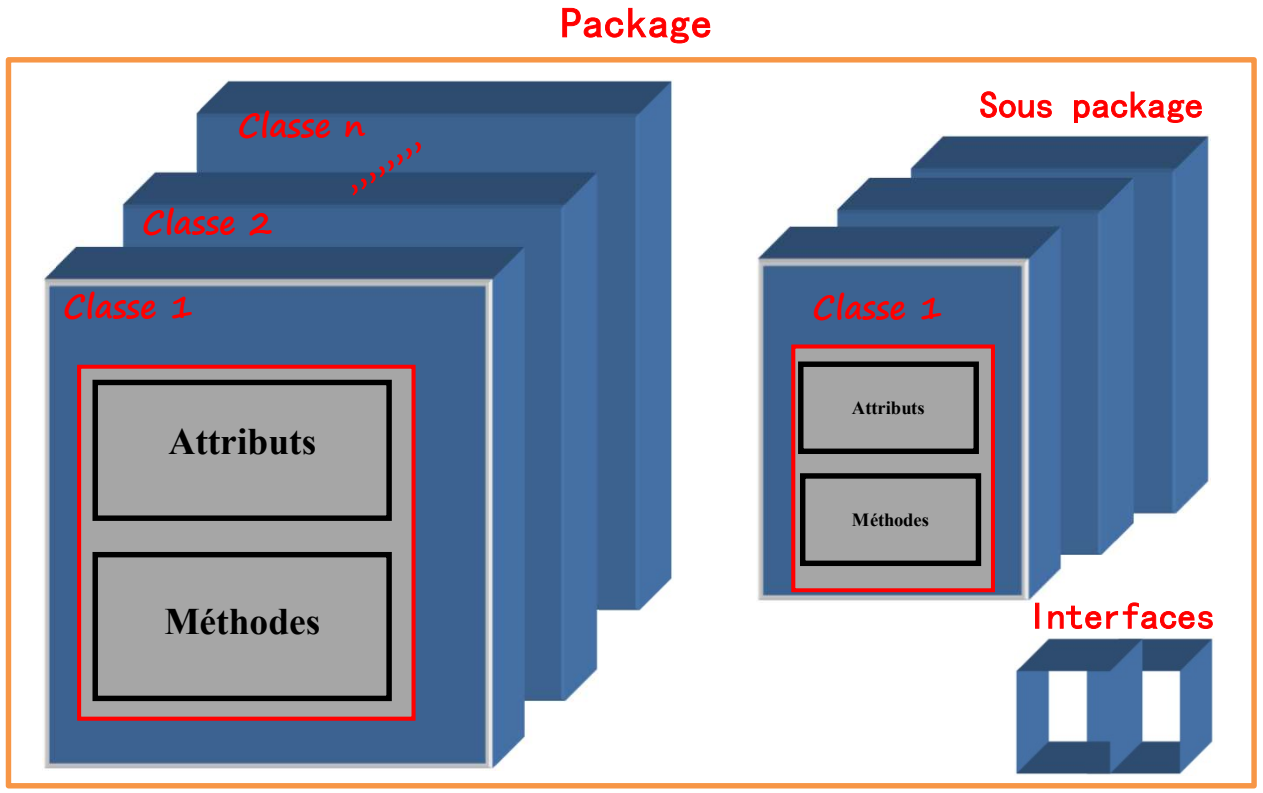
Bonjour.java

```
public class Bonjour {

          public static void main ( String [ ] args ) {
System.out.println ("Bonjour tout le monde !! ");
}
             }
```

- Le **main** est le point d'entrée pour l'exécution d'une application Java.
- La classe contenant la méthode **statique** **main** doit obligatoirement être **public** pour que la machine virtuelle y ait accès.
- Le profil de cette méthode est `public static void main ( String [ ] args )`
  - ***public*** : pour qu'elle puisse être exécutée par la machine virtuelle java (**JVM**).
  - ***static*** : pour que l'exécution puisse avoir lieu avant même la création d'une instance de la classe.
  - ***args*** : tableau d'objets String (chaînes de caractères) contenant les arguments de la ligne de commande

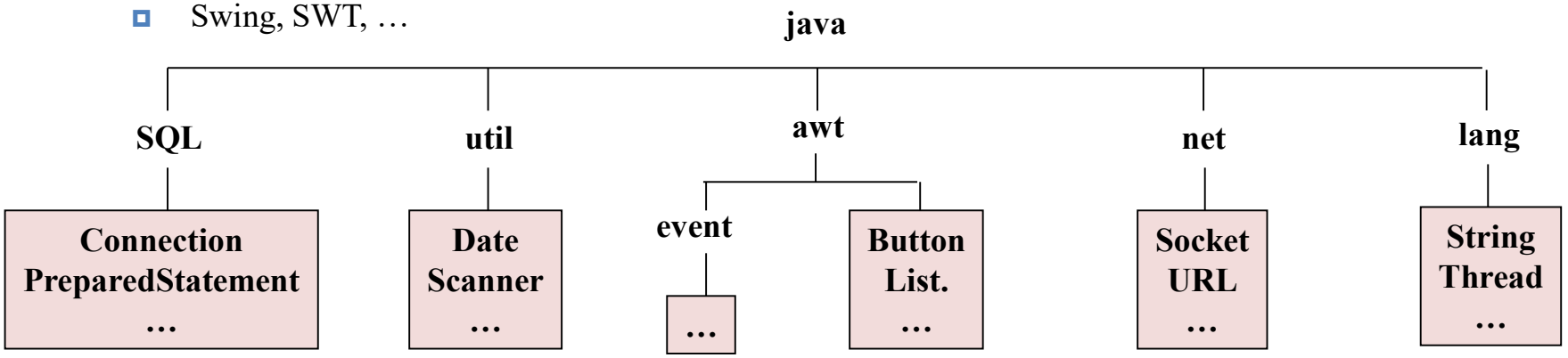
STRUCTURATIONS D'UN PROGRAMME

- **Structure d'un programme en P.O.O**
  - Programme Java utilise un ensemble de classes
  - Une classe regroupe un ensemble d'attributs et de méthodes
  - Les classes sont regroupées par package



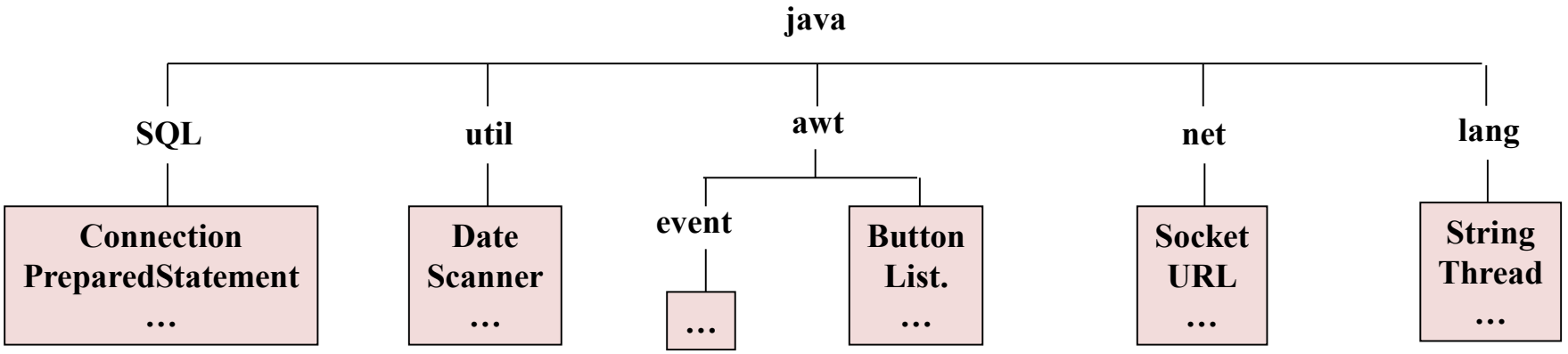
PACKAGES

- Un **Package** est une collection nommé de classes, d'interfaces et de sous-packages. Il permet de les regrouper sémantiquement.
- Exemples :
  - AWT : Abstract Windows Toolkit
  - Réseau : Sockets (serveurs, clients), Web …
  - Entrées/Sorties : JAVA.io
  - SQL
  - Swing, SWT, …



- Exemple
  - java.lang ;
  - java.util.Data ;
  - java.lang.String ;
  - java.awt.event.MouseEvent;

PACKAGES (QUELQUES APIS DE BASE)




```
java.util.Date dateDuJour = new
        java.util.Date();
System.out.println(dateDuJour);
```

Ou

```
import java.util.Date;
Date dateDuJour = new Date();
System.out.println(dateDuJour);
```

Ou

```
import java.util.* ;
Date dateDuJour = new Date();
System.out.println(dateDujour);
```

JAVA : ENTRÉES / SORTIES

- L'aspect le plus utile de la classe **System** est les variables qu'elle déclare, qui permettent d'avoir une interaction avec le système. On y trouve les variables ***in***, ***out*** et ***err***.
- La variable ***in*** représente le flux d'entrée standard du système, alors que la variable ***out*** représente le flux de sortie standard. La variable ***err*** est le flux d'erreur standard. (les flux vont être détailler prochainement)
  - Pour **afficher** une chaine de caractère :

```
System.out.println("Chaine de caractère") ;
```

  - Pour **afficher** une erreur :

```
System.err.println("Erreur à l'ouverture") ;
```

  - Pour **lire** une chaine de caractère :
    - Importer la bibliothèque java.util.Scanner;

```
Scanner s = new Scanner(System.in);
int i = s. nextInt();
```

nextInt
nextLine
nextDouble

2ÉME APPLICATION

SUR MACHINE

Merci pour votre attention