Présentation de Java

P.PISZYNA – SN2 – septembre 2020

Sommaire (1)

- Historique de Java
- Avantages
- Java dans le référentiel du BTS SN
- Classement
- Principe d'exécution d'un programme Java
- Principe de la machine virtuelle Java
- Cibles pour Java
- Etapes de développement

Sommaire (2)

- Rappels sur la POO
- Quelques classes de base
- Programmation réseau avec les sockets
- Programmation multi thread
- Programmation d'interfaces graphiques avec Swing
- Utilisation de la documentation

Sommaire (3)

- Création et intégration de librairies : les fichiers JAR
- Cas des bases de données
- Cas des jauges SteelSeries

Historique de Java

- Langage orienté objet apparu dans les années 1990 en même temps que Python et la standardisation du C++
- Langage issu du projet Oak (langage pour l'électronique grand public)
- La version actuelle du langage est SE (Standard Edition) 14 depuis 2020

Avantages

- Langage entièrement objet
- S'exécute sur n'importe quelle plateforme
- Un seul code source universel
- MultiThread (plusieurs fils d'exécution pour une même application)
- Robuste (pas de pointeurs)
- Simple (un seul fichier par classe)
- Gestion de la mémoire transparente pour l'utilisateur
- Facile à maitriser quand on connaît le C++

Java dans le référentiel SN

S4.7. Langages de programmation	C++	3	1
	Utilisation d'un langage objet (Java, C#, C++, etc.)	3	2
	SQL	3	2
	Web statique : HTML / XML	3	3
	Web dynamique : PHP, JavaScript	2	2
	Circuits programmables (graphique, descriptif, etc.)		3
	Langages graphiques par flux de données (simulation et instrumentation virtuelle)		3

Classement des langages de programmation

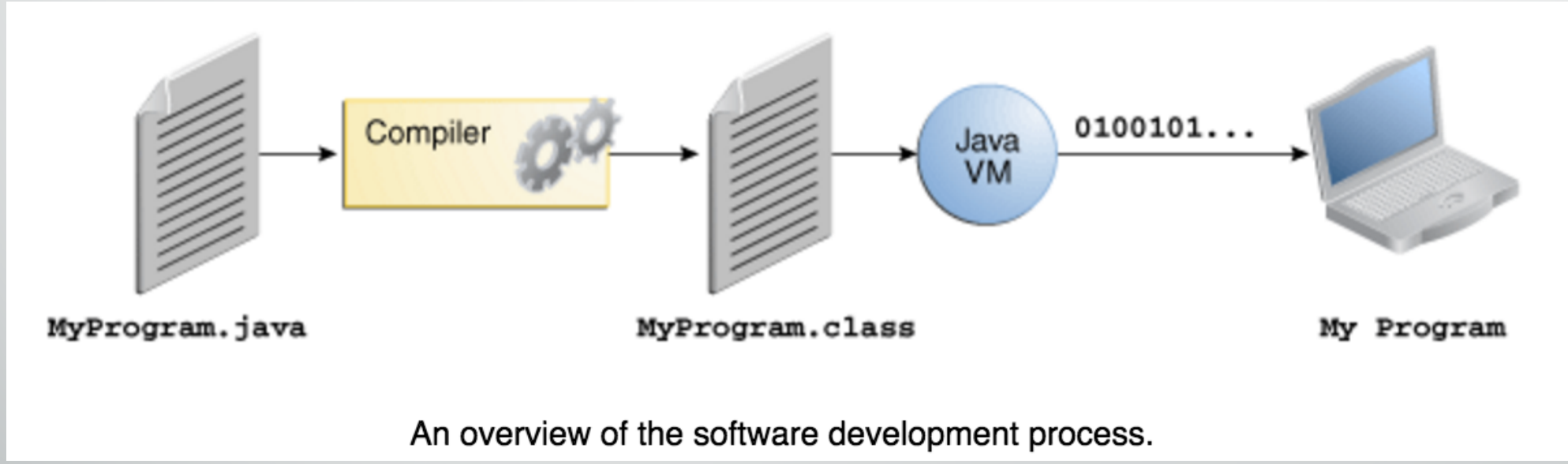
http://pypl.github.io/PYPL.html

Worldwide, Sept 2020 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Python	31.56 %	+2.9 %
2		Java	16.4 %	-3.1 %
3		Javascript	8.38 %	+0.3 %
4		C#	6.5 %	-0.8 %
5		PHP	5.85 %	-0.5 %
6		C/C++	5.8 %	+0.0 %
7		R	4.08 %	+0.3 %
8		Objective-C	2.79 %	+0.2 %
9		Swift	2.35 %	-0.1 %
10		TypeScript	1.92 %	+0.1 %
11		Matlab	1.65 %	-0.1 %
12		Kotlin	1.61 %	+0.1 %

Principe d'exécution d'un programme Java

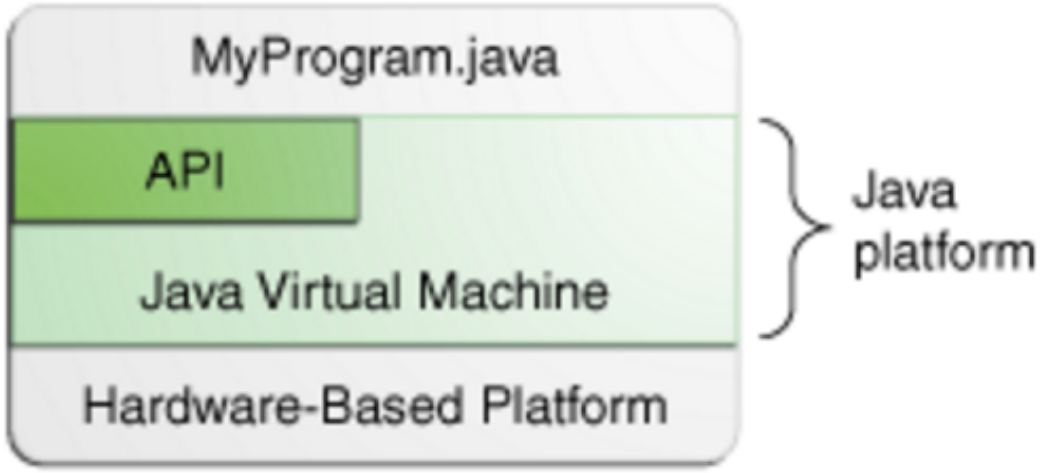
(source : Oracle)



An overview of the software development process.

Principe de la machine virtuelle Java

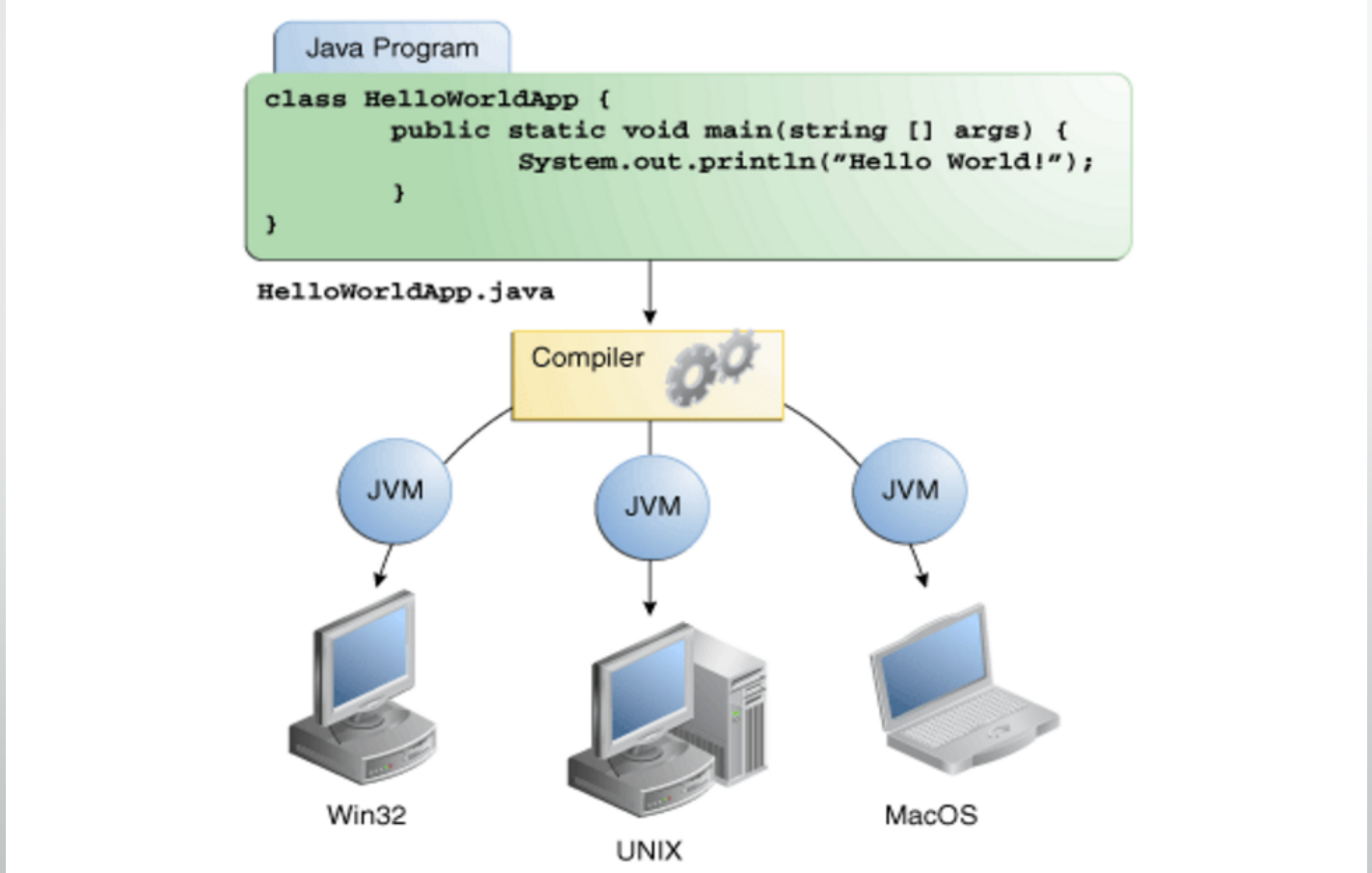
(source : Oracle)



The API and Java Virtual Machine insulate the program from the underlying hardware.

Cibles pour Java (1)

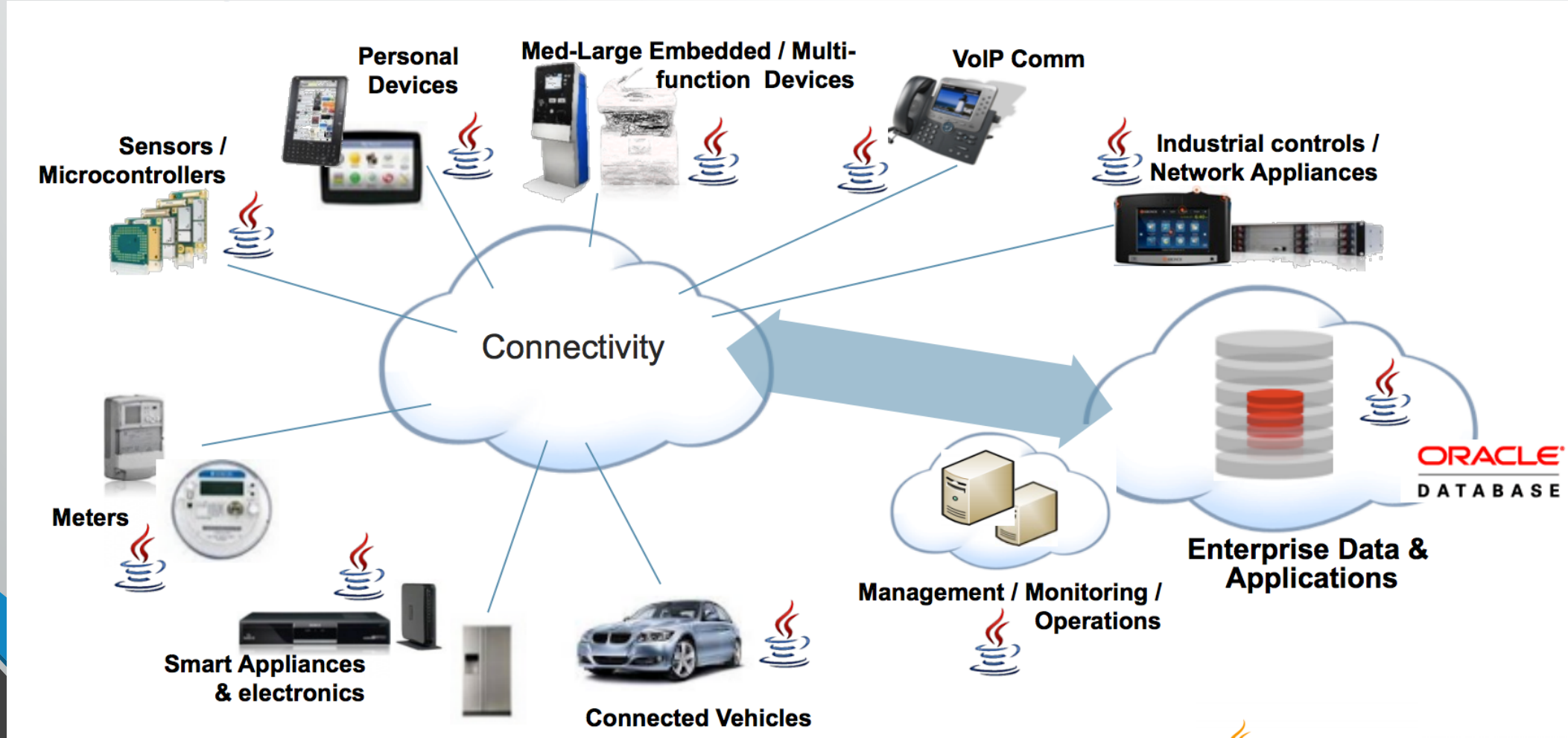
(source : Oracle)



Through the Java VM, the same application is capable of running on multiple platforms.

Cibles pour Java (2)

(source : Oracle)



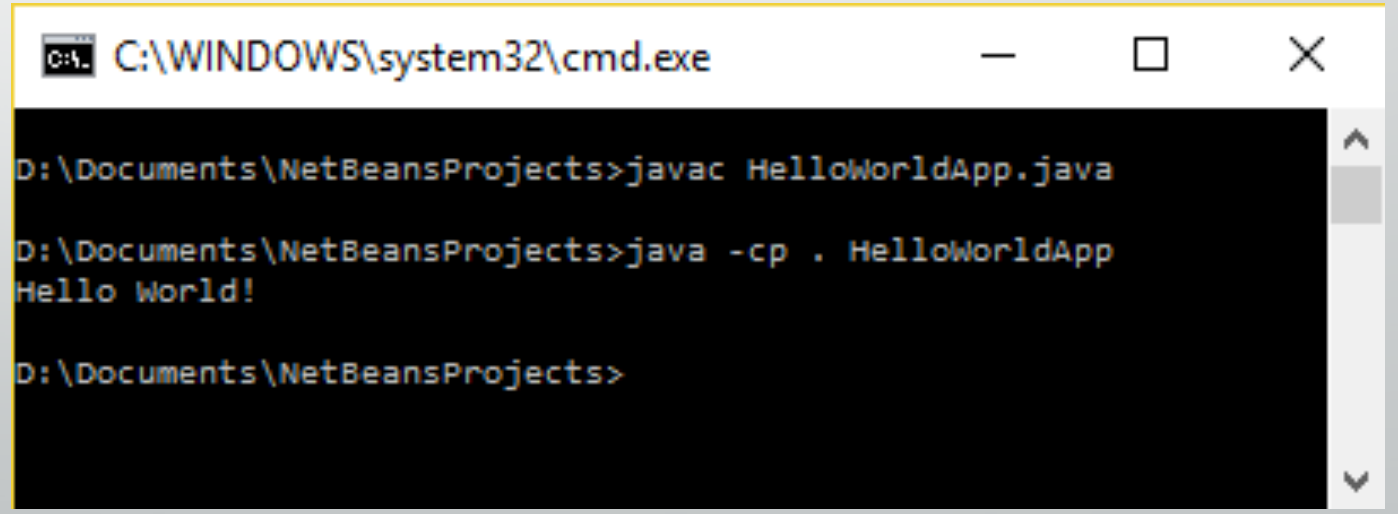
Etapes de développement

1. Création du code source
   - A partir des spécifications (UML, algorithme…)
   - Avec un éditeur de texte ou un EDI
2. Compilation en Byte-Code
   - A partir du code source
   - Avec le compilateur Java
3. Diffusion sur la cible (optionnel)
   - A partir du Byte-Code
   - Sur la cible locale ou avec le réseau
4. Exécution sur la cible
   - A partir du Byte-Code
   - Avec la JVM

Etapes de développement

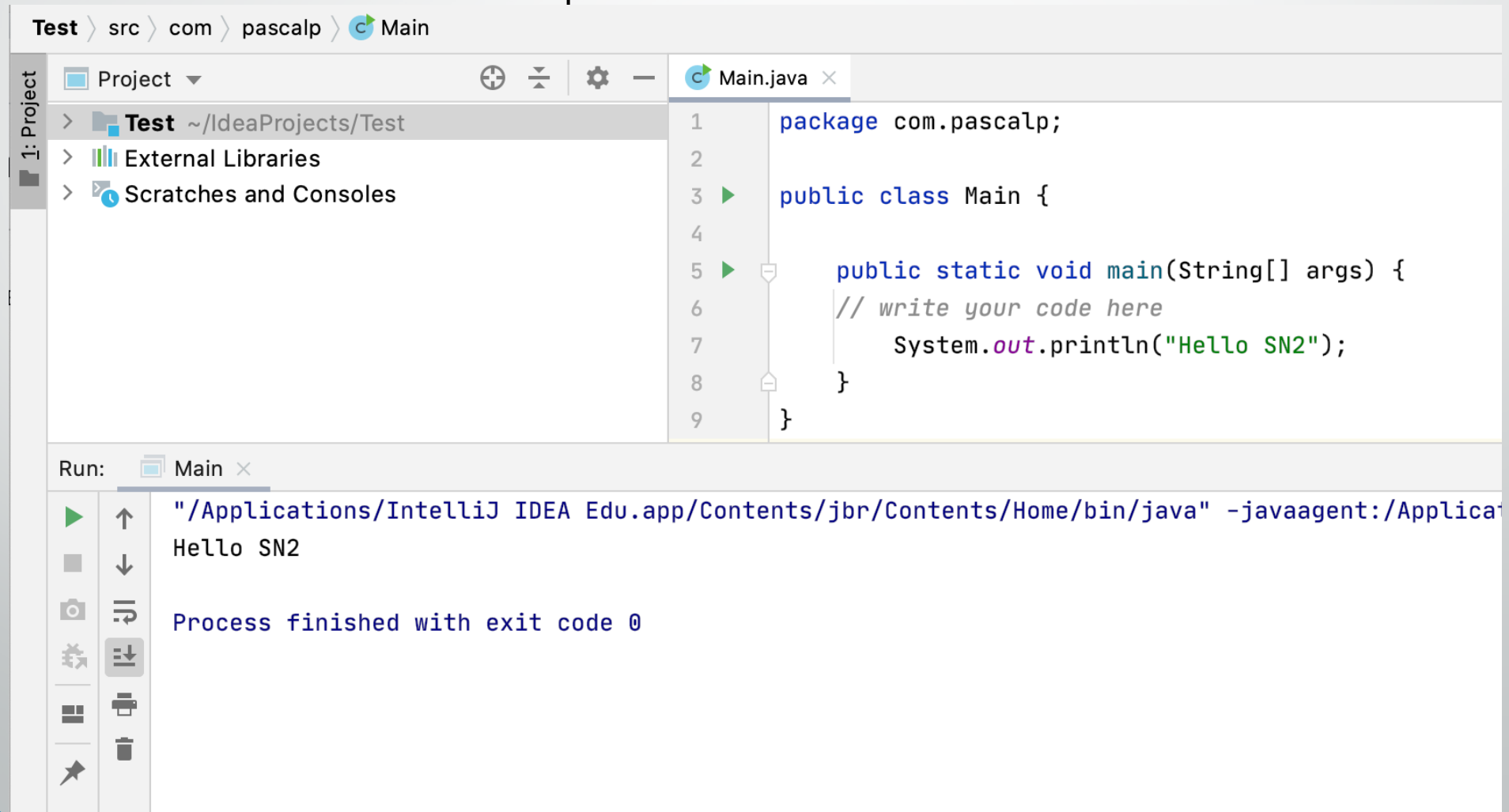
Exemple en ligne de commande

```java
class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello World!"); // Display the string.
    }
}
```



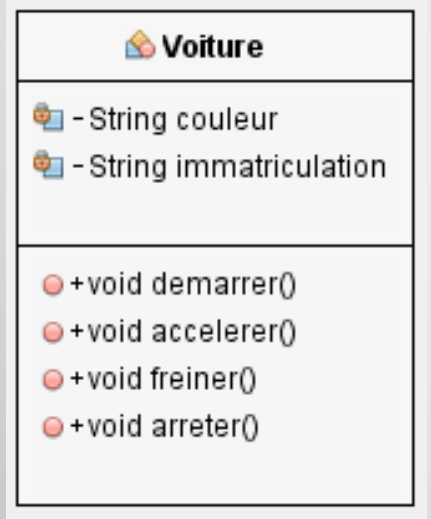
Etapes de développement

Exemple avec Intelli J



Quelques rappels sur la POO

Les classes et les objets



Une classe est un modèle de définition d'objets ayant en commun :

- Des propriétés (attributs)
- Des comportements (méthodes)

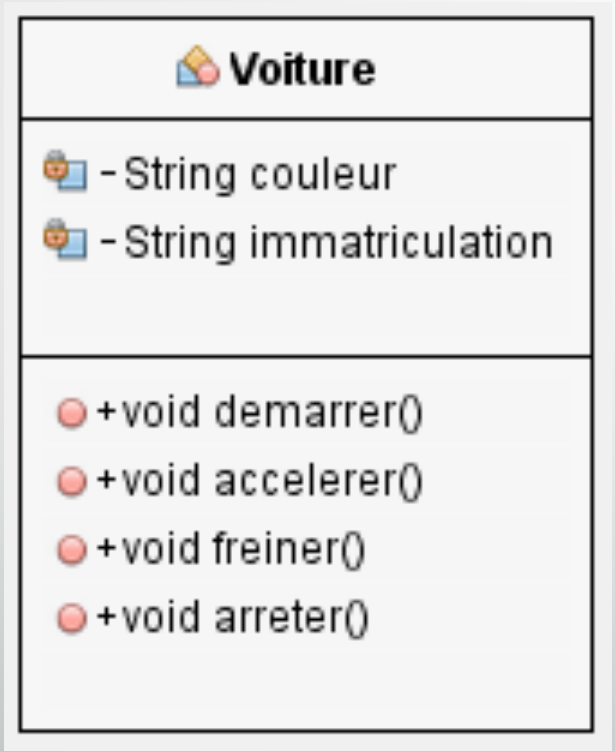
Un objet est une instance d'une classe

Par exemple, la classe Voiture permet de créer (instancier) plusieurs objets :

- maLeaf
- laMercedesDeLewis
- laTeslaDeMusk
- …

Quelques rappels sur la POO

Les classes en Java




```java
public class Voiture {
    private String couleur;
    private String immatriculation;

    public void demarrer() {
    }
    public void accelerer() {
    }
    public void ralentir() {
    }
    public void arreter() {
    }
}
```

Quelques rappels sur la POO

Conventions en Java

- Conventions de noms
  - CeciEstUneClasse
  - celaEstUneMethode(…)
  - jeSuisUneVariable
  - JE_SUIS_UNE_CONSTANTE
- Un fichier par classe, une classe par fichier
  - Classe « Voiture » décrite dans le fichier Voiture.java

Quelques rappels sur la POO

Les classes en Java – Démo Netbeans & EasyUML

- Création d'un projet UML et d'une classe Voiture
- Création d'une application Java
- Génération du code de la classe voiture
- Modification du code de la classe voiture :
  - Ajout d'un constructeur
  - Ajout d'un booléen « demarree » et de son getter
- Génération du diagramme de la classe modifiée
- Génération du diagramme de classes du projet
- Modification de la classe VoitureApp : l'objet « maLeaf » devient un attribut

Quelques rappels sur la POO

Les classes en Java – Implémentation d'un cas d'utilisation

- Une voiture roule à une vitesse comprise entre 0 et 100 km/h
- L'accélération et le freinage se font par paliers de 10 km/h
- L'accélération et le freinage ne peuvent se faire que si la voiture est démarrée
- Compléter la classe Voiture et la fonction main() pour :
  - Démarrer la voiture
  - Accélérer jusqu'à 90 km/h.
  - Chaque action provoque un affichage
- Implémenter et tester la méthode freiner()

Quelques classes de base

La classe System (source: http://www.jmdoudoux.fr/)

Cette classe possède de nombreuses fonctionnalités pour utiliser des services du système d'exploitation.

La classe System définit trois variables statiques qui permettent d'utiliser les flux d'entrée/sortie standards du système d'exploitation.

Variable	Type	Rôle
in	InputStream	Entrée standard du système. Par défaut, c'est le clavier
out	PrintStream	Sortie standard du système. Par défaut, c'est le moniteur
err	PrintStream	Sortie standard des erreurs du système. Par défaut, c'est le moniteur

Exemple :

```
1. System.out.println("bonjour");
```

Quelques classes de base

La classe String (source: http://www.jmdoudoux.fr/)

- Une chaîne de caractères est contenue dans un objet de la classe java.lang.String.
- On peut initialiser une variable String sans appeler explicitement un constructeur : le compilateur se charge de créer un objet.

Exemple : deux déclarations de chaines identiques.

```
1. String uneChaine = "bonjour";
2. String uneChaine = new String("bonjour");
```

Exemple :

```
1. uneChaine = chaine.toUpperCase().trim();
```

Quelques classes de base

La classe String (source: Bessem BOURAOUI)

- Ce sont des objets traités comme des types simples...
- Initialisation

```
String maChaine = "Bonjour!"; // Cela ressemble à un type simple
```

- Longueur

```
maChaine.length(); // Avec les parenthèses car c'est une méthode
```

- Comparaison

```
maChaine.equals("Bonjour!"); // Renvoi vrai
```

- Concaténation

```
String essai = "ess" + "ai";
String essai = "ess".concact("ai");
```

Faites attention à la comparaison de chaînes de caractères.

```
maChaine == "toto";
```

Comparaison sur les références !!

Quelques classes de base

La classe String (source: Bessem BOURAOUI)

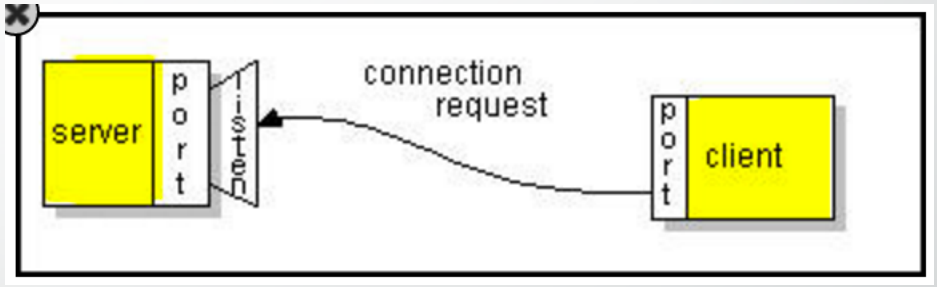
```
String s = "C' est " ; // création
String t = s + " le moment " ; // concaténation
String t1 = s + " " + 2.54 ; // conversion
int len = t . length ( ) ; // t a i l l e d ' une chaîne
String sub = t . substring ( 3 , 5 ) ; // extraction d'une sous-chaîne
char c = t . charAt ( 2 ) ; // extraction d'un caractère
boolean b1 ;
b1 = t . equals ( " h e l l o " ) ; // test d'égalité
int r ; r = t . compareTo ( " bonjour " ) ; // comparaison alphabétique
r = t . indexOf ( 't' ) ; // index d'éléments
r = t . indexOf ( 't' , r+1) ;
r = t . lastIndexOf ( 't' ) ;
r = t . lastIndexOf ( 't' , r-1);
String nouv = t . replace ( ' a ' , ' h ' ) ;  //remplacement sans modification de
    la chaîne initiale
```

Programmation réseau avec les sockets

La classe Socket (source : Oracle)

Création de la Socket :

```
Socket socket = new Socket(hostName, portNumber);

// Crée la socket et se connecte au serveur
```



Flux d'entrée/sortie pour envoyer et recevoir des données :

```
PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
```

Programmation réseau avec les sockets

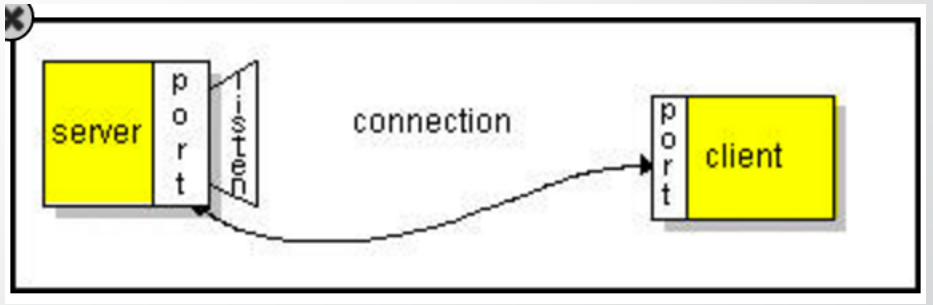
La classe ServerSocket (source : Oracle)

Création de la socket d'écoute:

```
ServerSocket serverSocket = new ServerSocket(portNumber);
// Crée la socket et se met à l'écoute
```

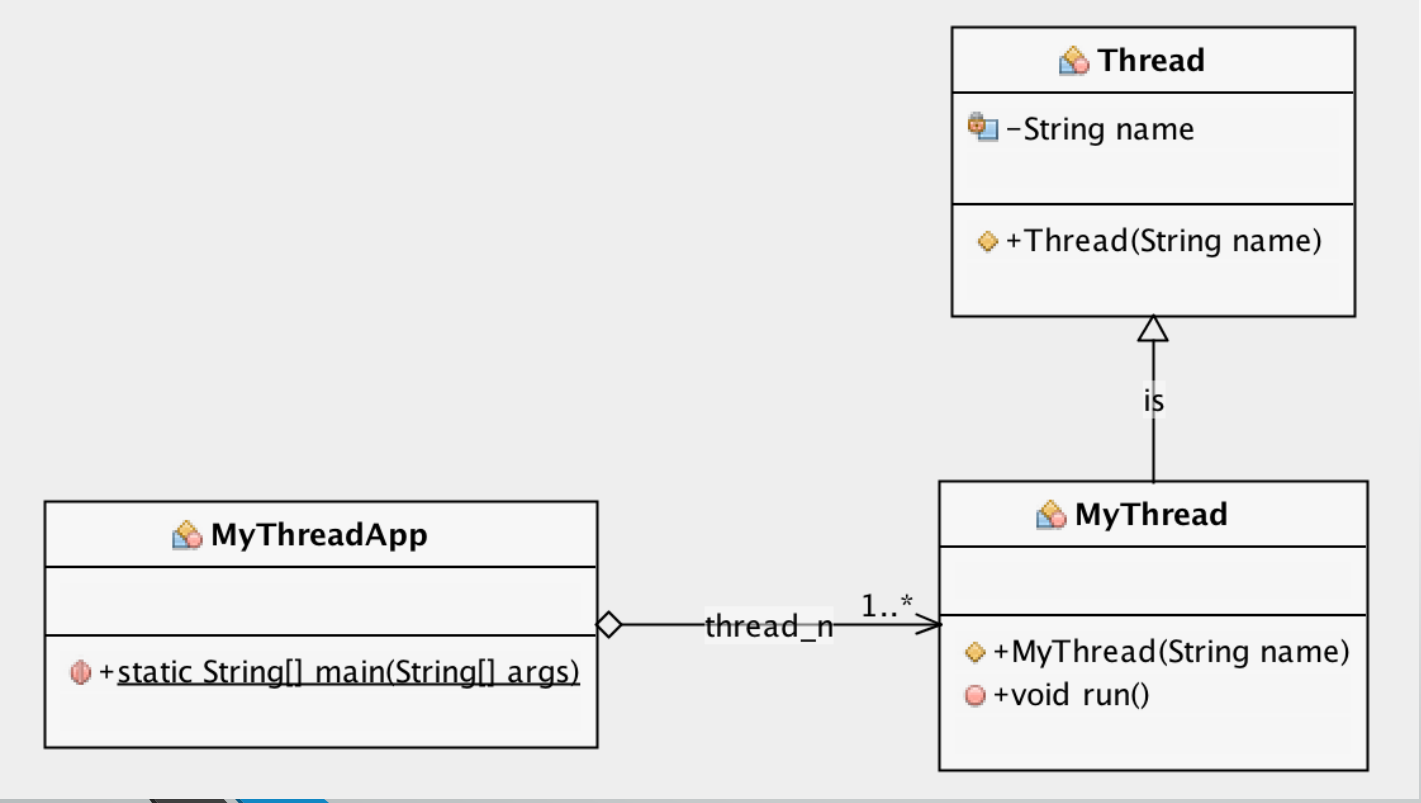
Création de la socket de données :

```
Socket clientSocket = serverSocket.accept();
// On attend la demande de connexion du client
```



Programmation multithread

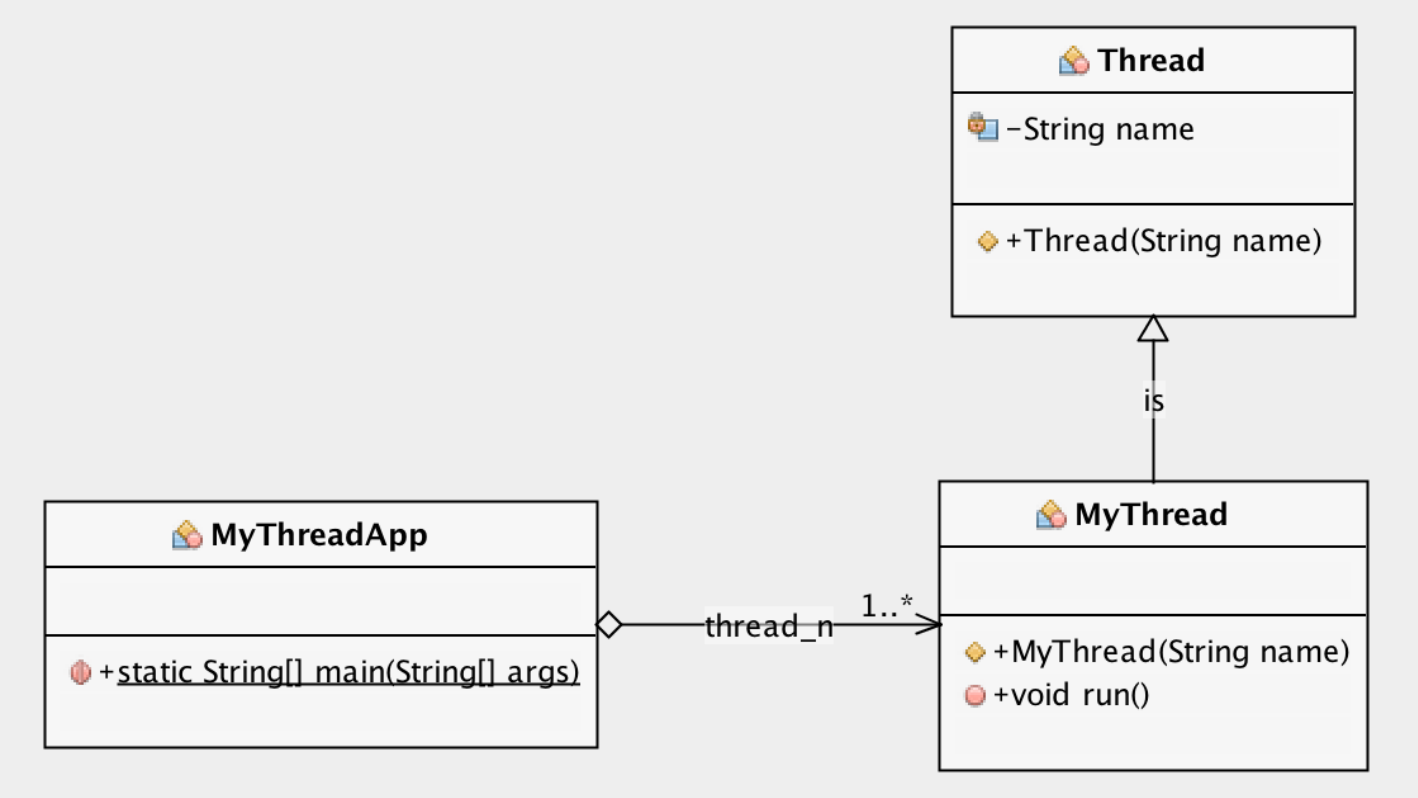
(Par héritage)




```java
public class MyThread extends Thread {

    public MyThread(String name) {
        super(name);
        start();
    }

    @Override
    public void run() {
        System.out.println(getName() + " démarre");
        // Instructions à exécuter
        System.out.println(getName() + " terminé");
    }
}
```

Programmation multithread

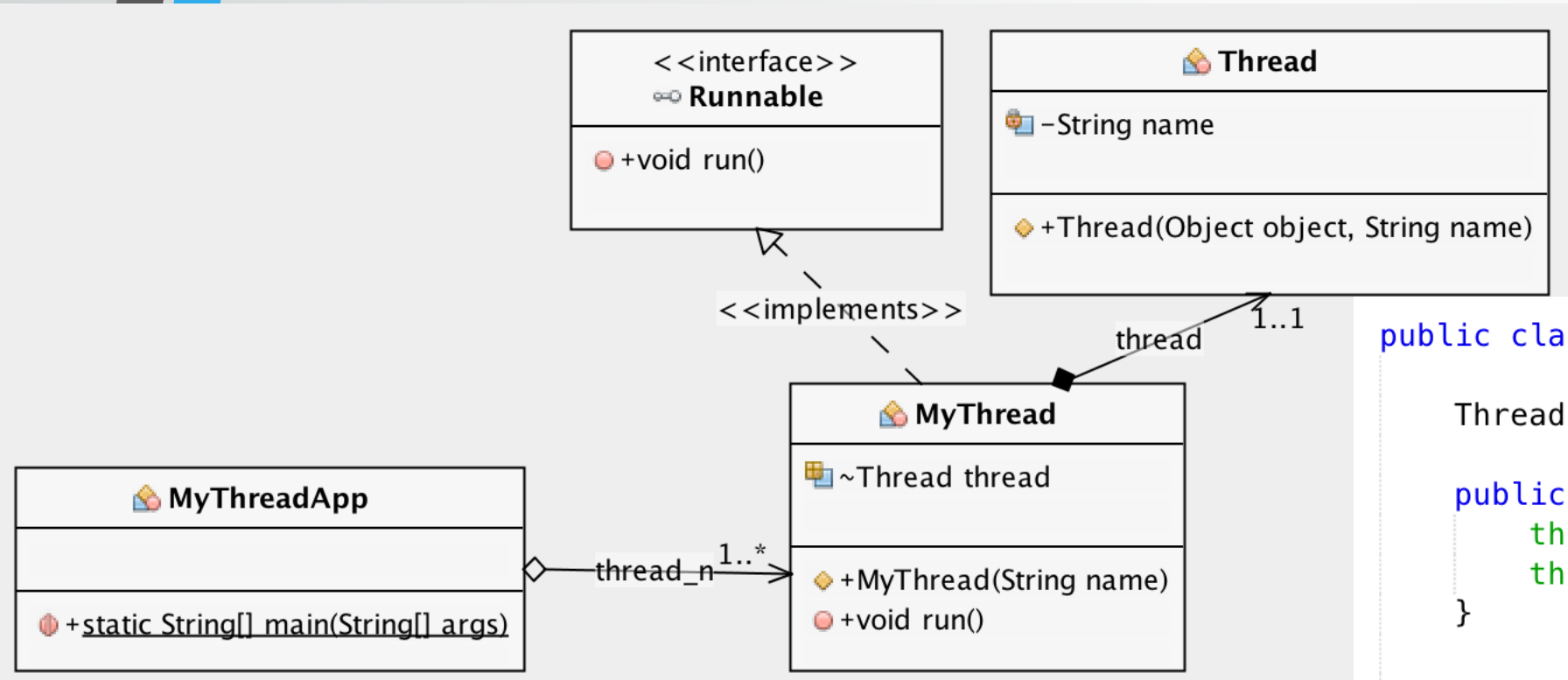
(Par héritage)




```java
public class MyThreadApp {

    public static void main(String[] args) {
        System.out.println("Main démarre");
        MyThread thread1 = new MyThread("thread 1");
        System.out.println("Main terminé");
    }

}
```

Programmation multithread

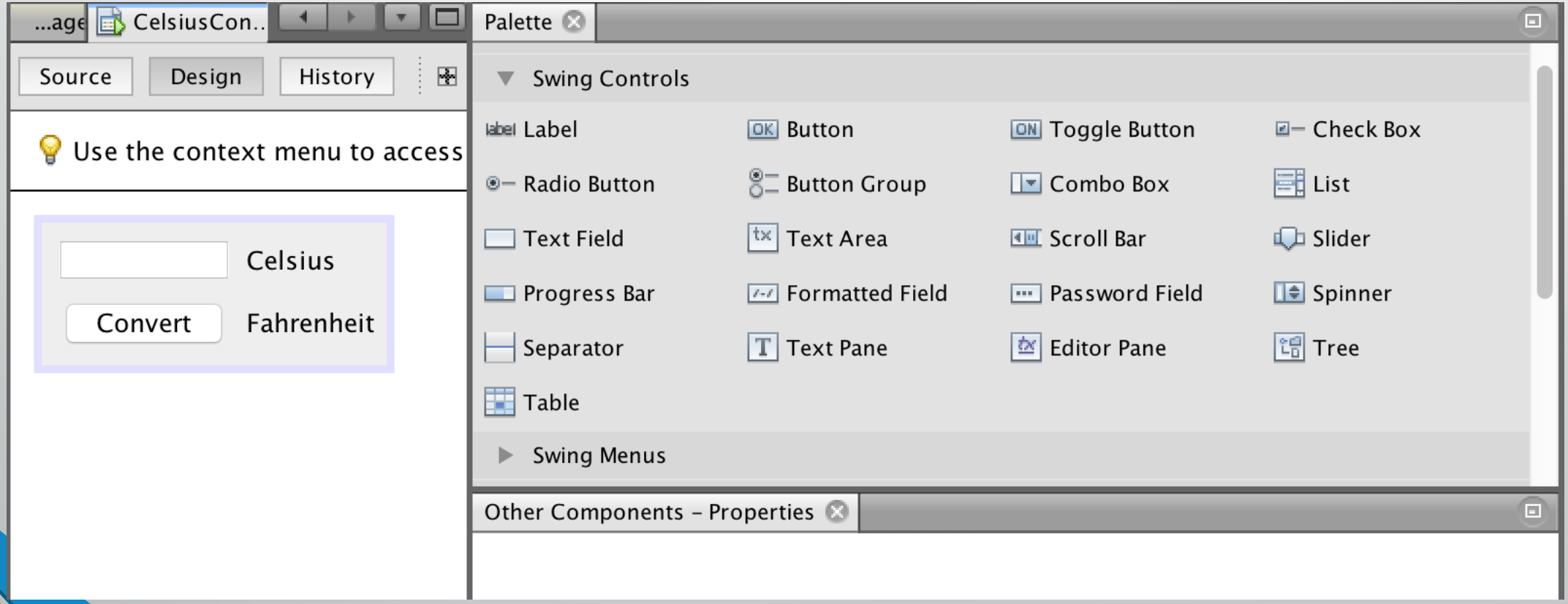
(Par composition)




```java
public class MyThread implements Runnable {

    Thread thread;

    public MyThread(String name) {
        thread = new Thread(this, name);
        thread.start();
    }

    @Override
    public void run() {...5 lines }
}
```

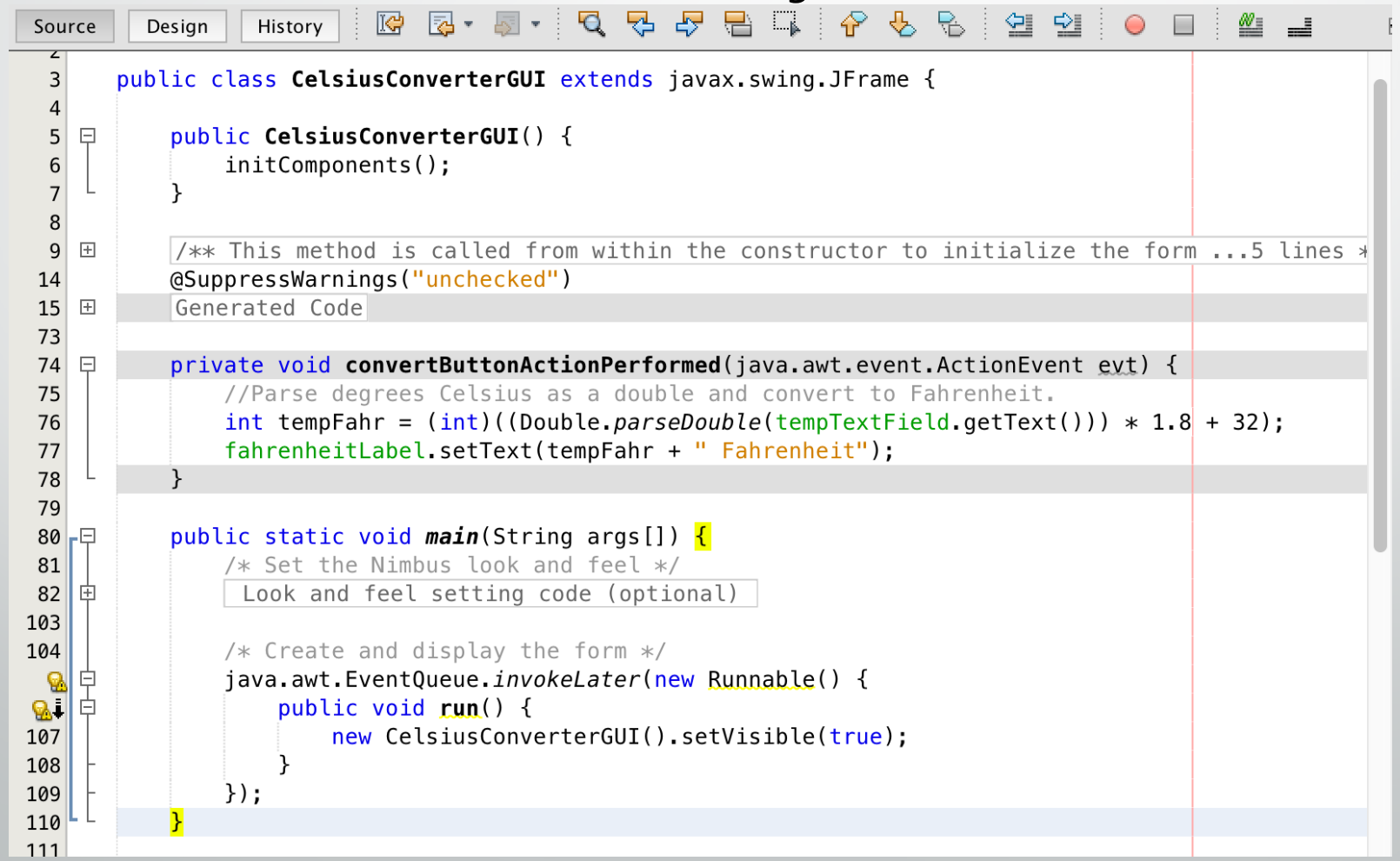
Programmation d'interfaces graphiques

Avec Swing

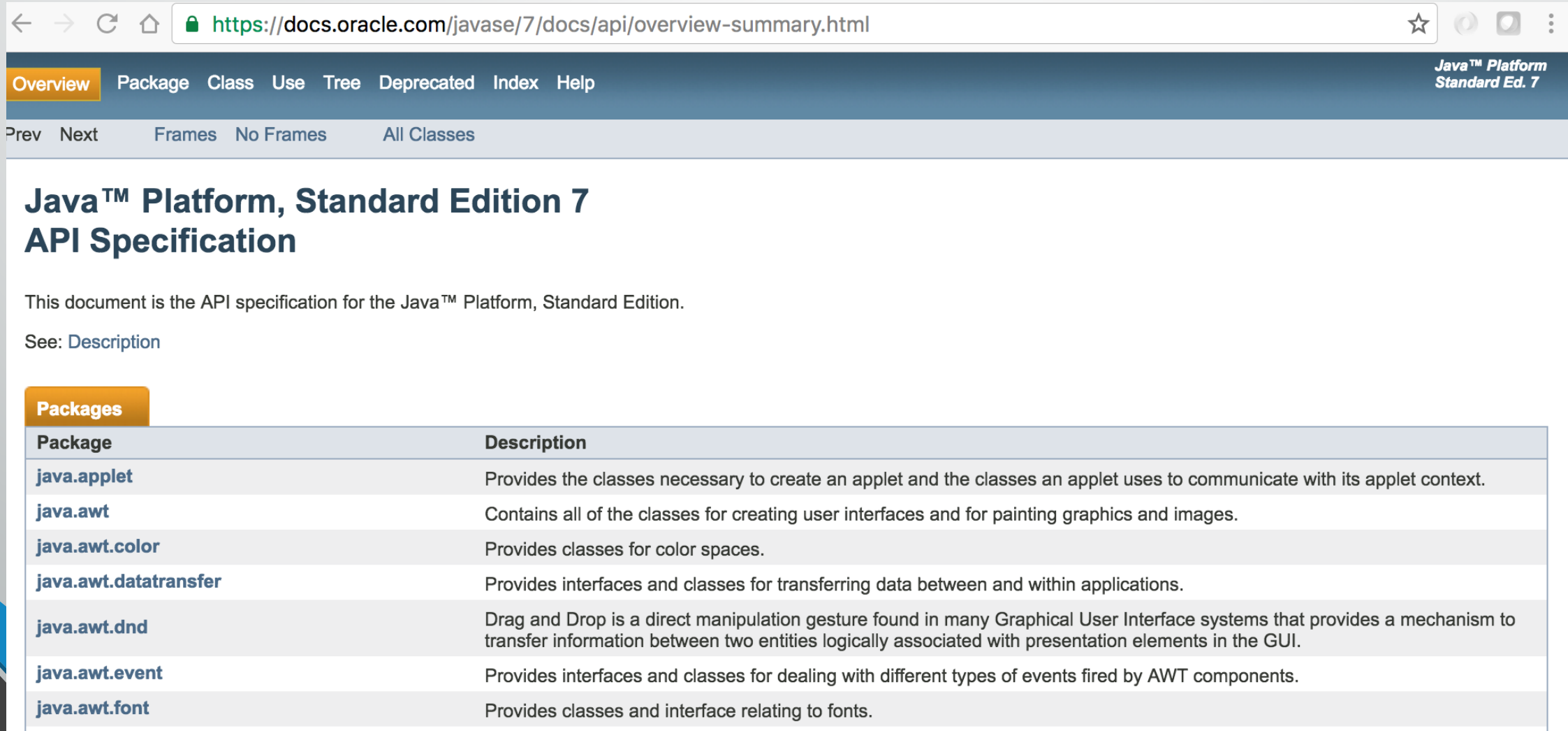


Programmation d'interfaces graphiques

Avec Swing



Utilisation de la documentation



Création et intégration de librairies

Les fichiers JAR (Java Archive)

- Démonstration simple :
  - Création d'un projet « Java library » avec la classe Voiture
  - Création d'un projet « Java application »
  - Importation de la classe Voiture
  - Ajout de documentation avec Javadoc

Création et intégration de librairies

Les fichiers JAR (Java Archive)

- Démonstration avec les bases de données :
  - Rappel sur les bases de données
  - Création d'une base de données « Utilisateurs »
  - Modélisation avec MySQLWorkbench
  - Génération des tables
  - Peuplement des tables
  - Création d'un projet « Java application »
  - Importation du connecteur JDBC (https://dev.mysql.com/downloads/connector/j/)
  - Accès à la base « Utilisateurs »

Création et intégration de librairies

Les fichiers JAR (Java Archive)

- Démonstration avec des jauges en mode graphique:
  - Création d'un projet « Java application » avec Swing
  - Importation de la librairie SteelSeries
  - Création d'une jauge pour visualiser la puissance apparente (projet TCE)