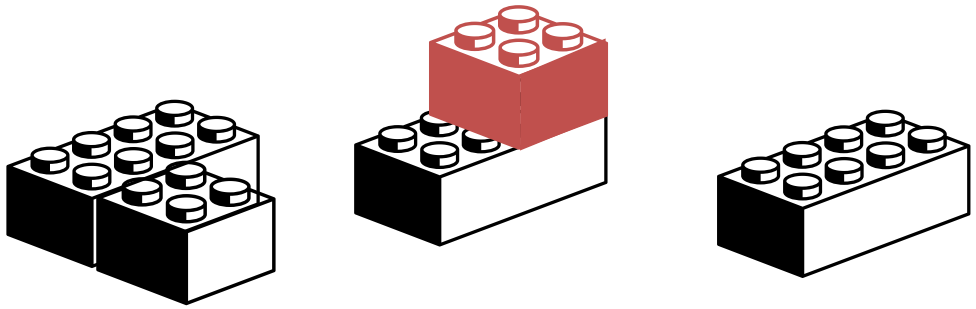
POO
Programmation Orientée Objets

Mise en œuvre par le langage JAVA

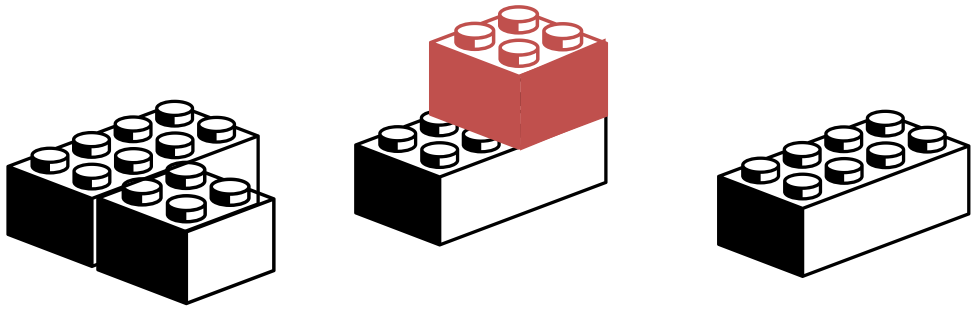
J. Saraydaryan



Les Besoins de conceptions

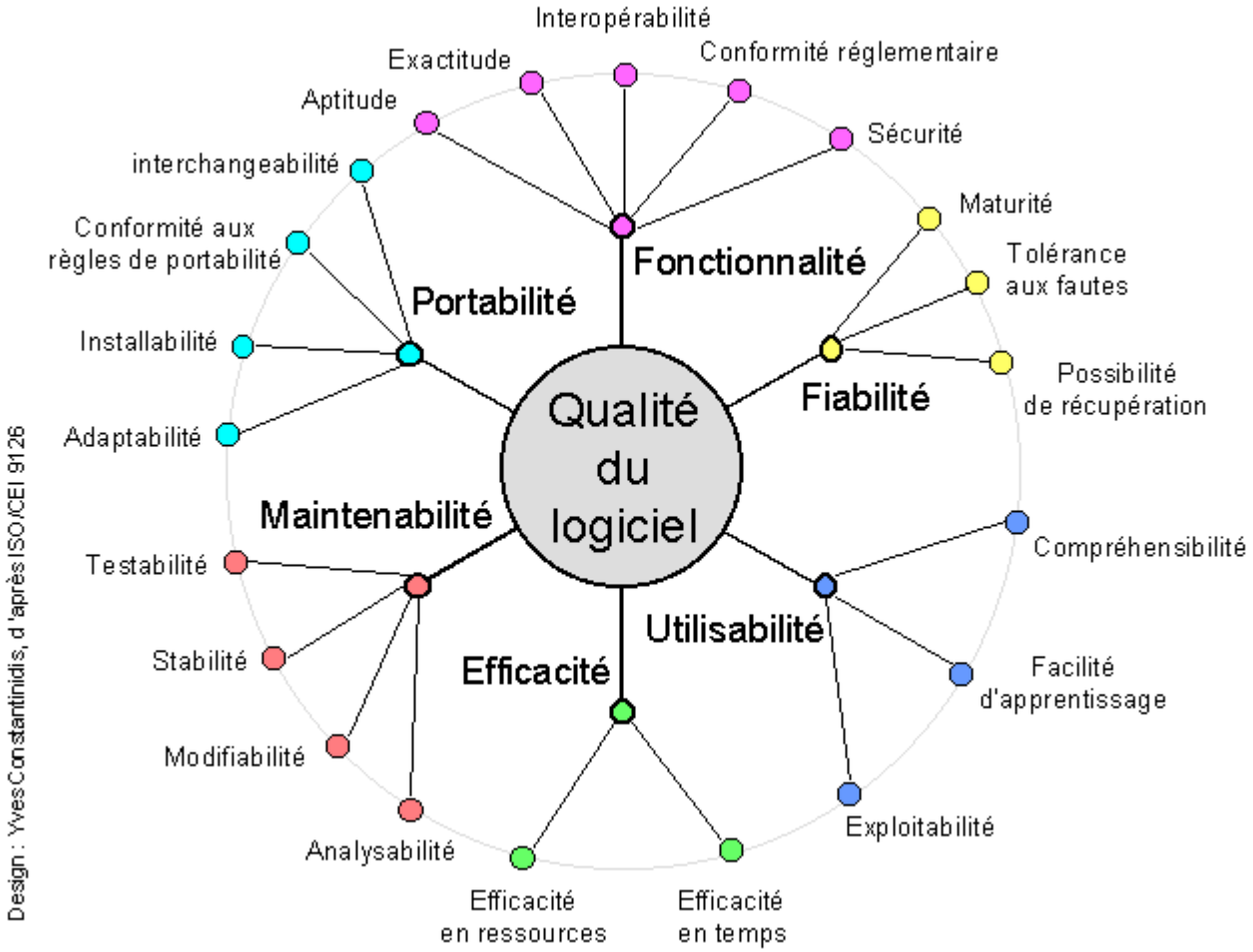
Mise en œuvre par le langage JAVA

J. Saraydaryan

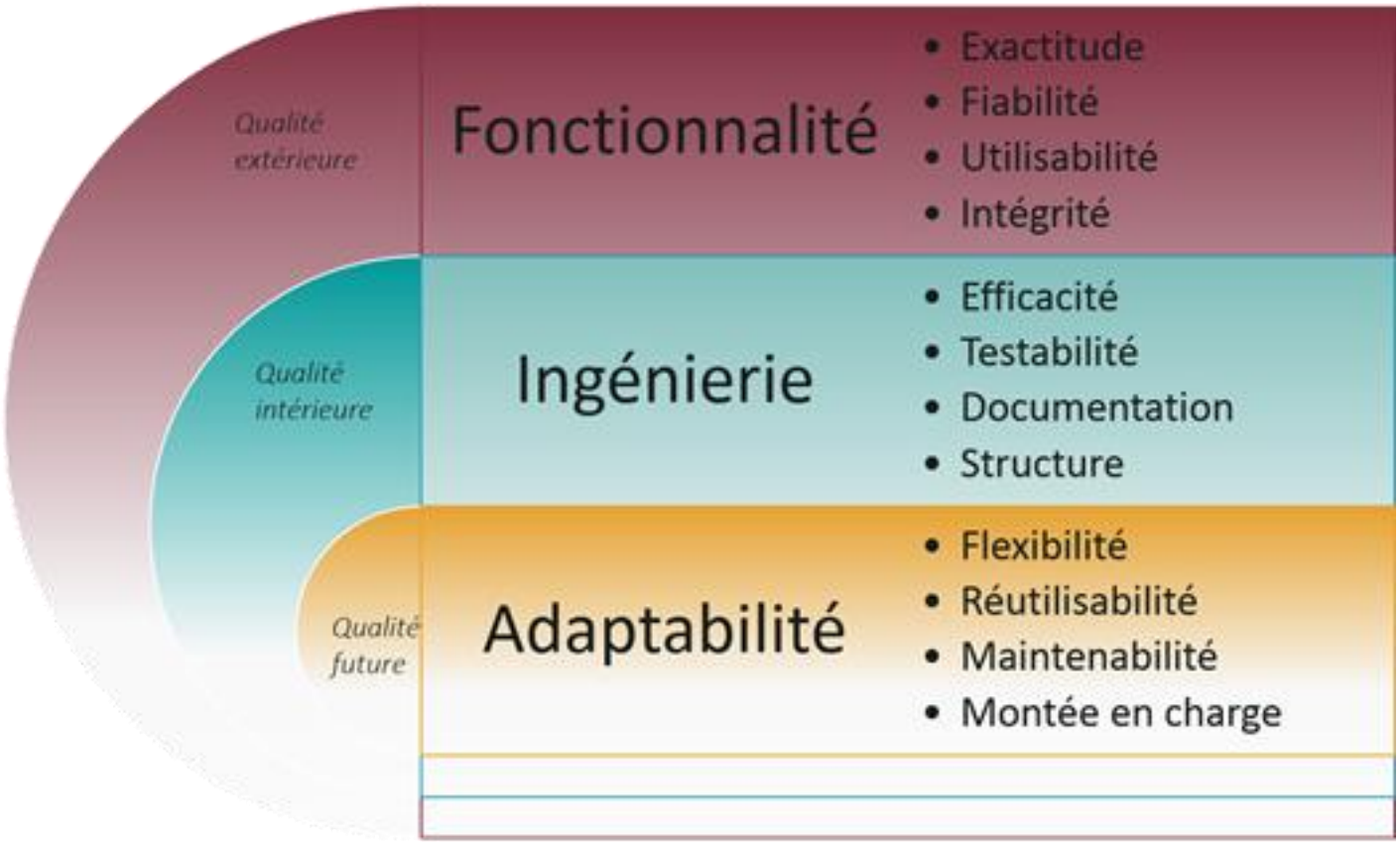


Motivation

Qualité d'un logiciel

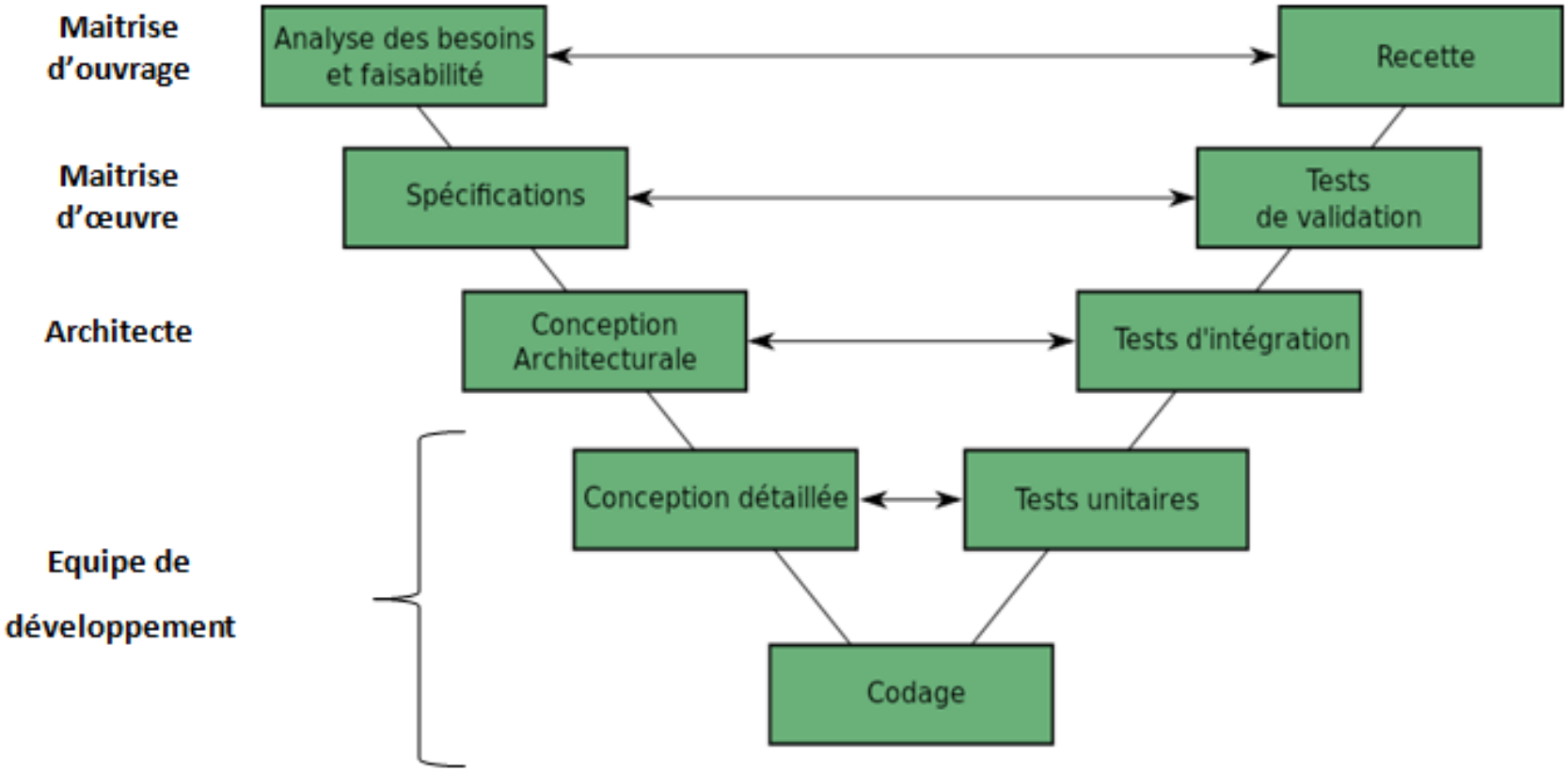


Qualité d'un logiciel



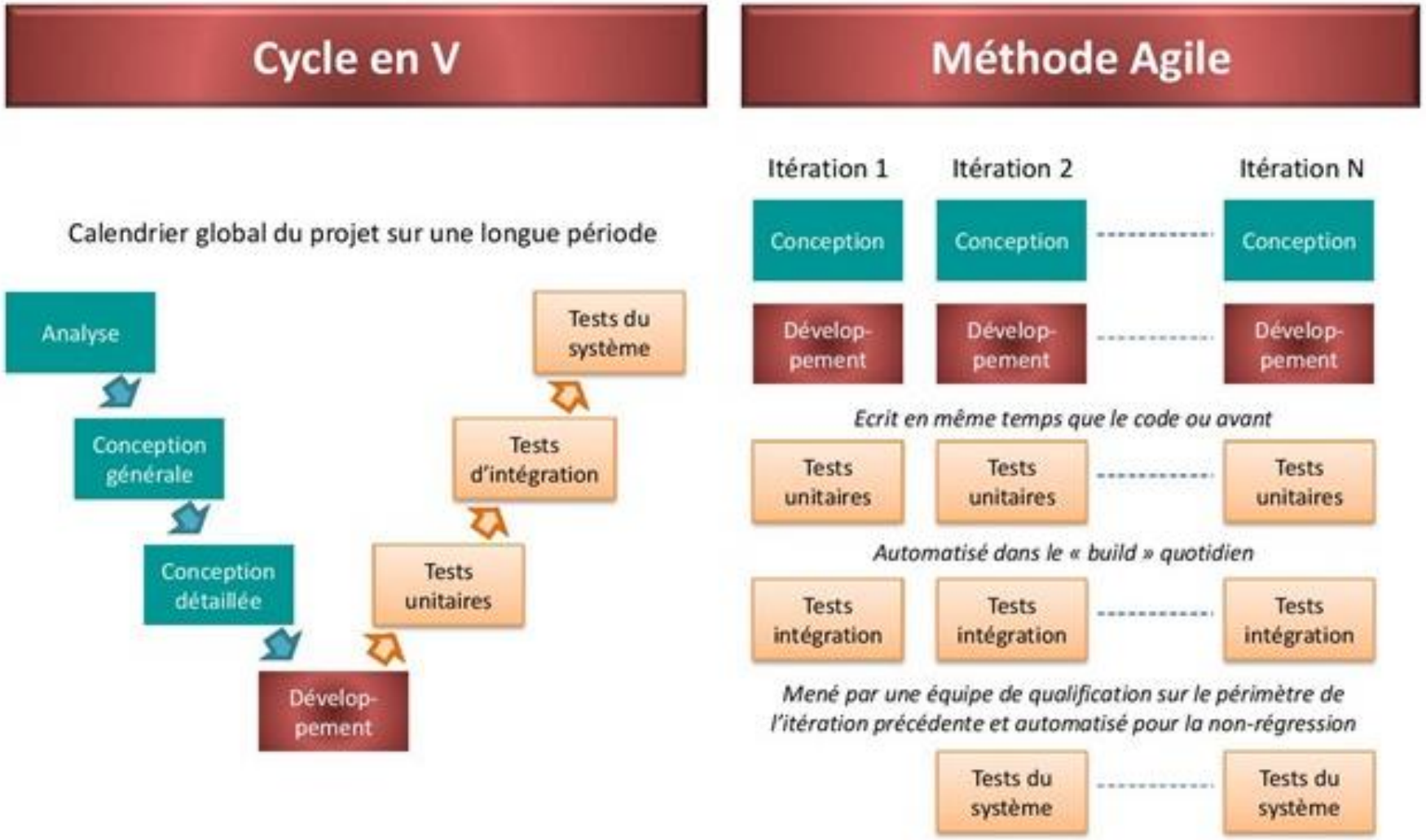
https://www.softfluent.fr/blog/societe/Les-meilleures-pratiques-du-test-logiciel

Cycle de vie d'un logiciel



https://chefdequipes.files.wordpress.com/2014/05/cycle-en-v-rc3b4les.png

Cycle de vie d'un logiciel



https://www.softfluent.fr/blog/societe/Les-meilleures-pratiques-du-test-logiciel

Type de programmation

- Programmation Procédurale (1/3)
  - Organisation d'un programme permettant d'appeler à tout moment une procédure (et ce même lors de récursion)
  - Une procédure ou fonction est une série d'actions ou d'étapes

Type de programmation

- Programmation Procédurale (2/3)
  - Avantages
    - Réutilisation du code
    - Maintenabilité facilité
    - Lecture du code plus aisée
    - Début de la modularité
    - Définition d'un scope (d'une portée des variables)
  - Inconvénients
    - Les données sont exposées à tout le programme (pb. sécurité)
    - Peu de cloisonnement des responsabilités
    - La généralisation est limitée → Réutilisation réduite
    - Pas aussi efficace que des langages de bas niveau (assembleur)
    - Difficulté pour représenter des problèmes concrets (objets de la vie réelle)
    - Ecriture de programmes difficile pour les projets conséquents
  - Langages procéduraux:
    - C, COBOL, etc…

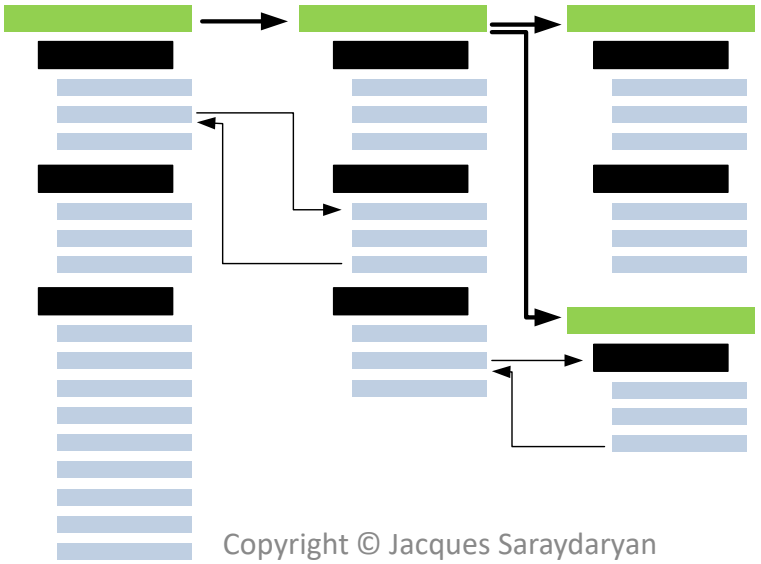
Type de programmation

- Programmation Procédurale (3/3)

```c
#include <stdio.h>

int addNumbers(int a, int b);          // function prototype

int main()
{
    int n1,n2,sum;
    printf("Enters two numbers: ");
    scanf("%d %d",&n1,&n2);
    sum = addNumbers(n1, n2);         // function call
    printf("sum = %d",sum);

    return 0;
}

int addNumbers(int a,int b)          // function definition
{
    int result;
    result = a+b;
    return result;                   // return statement
}
```

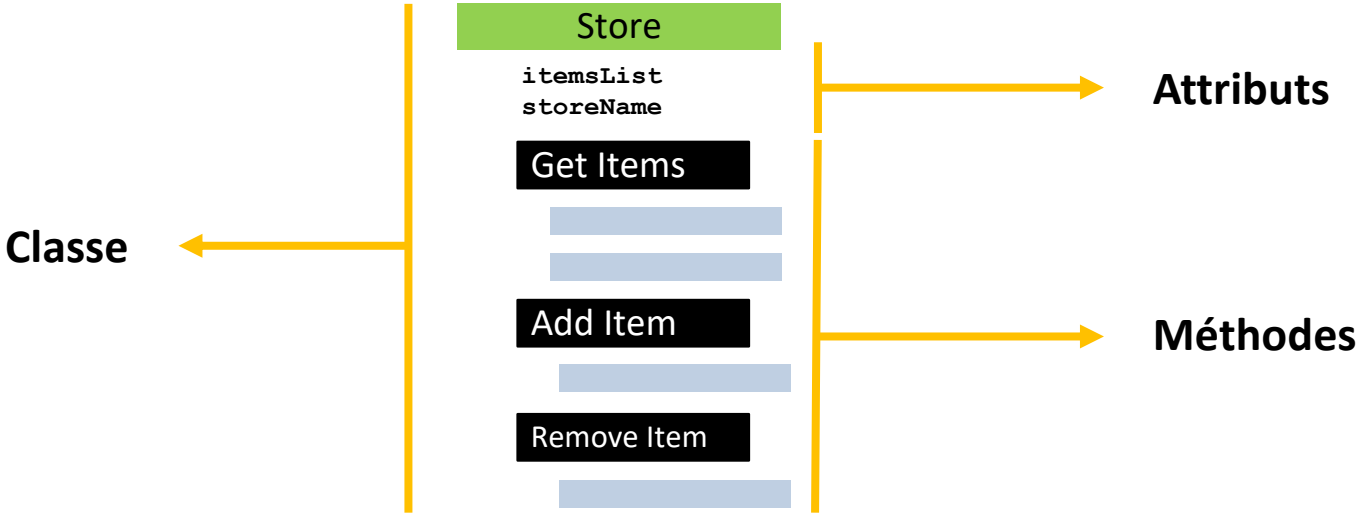
Que permet de faire POO ?

- Programmation Orientée Objet (POO) (1/7)
  - Objectifs:
    - Concevoir et maintenir de gros projets, réutiliser des éléments
    - Proche des éléments réels: collection d'objets qui en travaillant conjointement permettent de résoudre un problème



Que permet de faire POO ?

- Programmation Orientée Objet (POO) (2/7)
  - Qu'est ce qu'un objet ?
    - **Object** (classe)
      - **Attributs** : variables propres à l'objet (dont il est en charge)
      - **Méthodes**: fonctions qui définissent le comportement de l'objet



Que permet de faire POO ?

- Programmation Orientée Objet (POO) (3/7)
  - **Le polymorphisme**

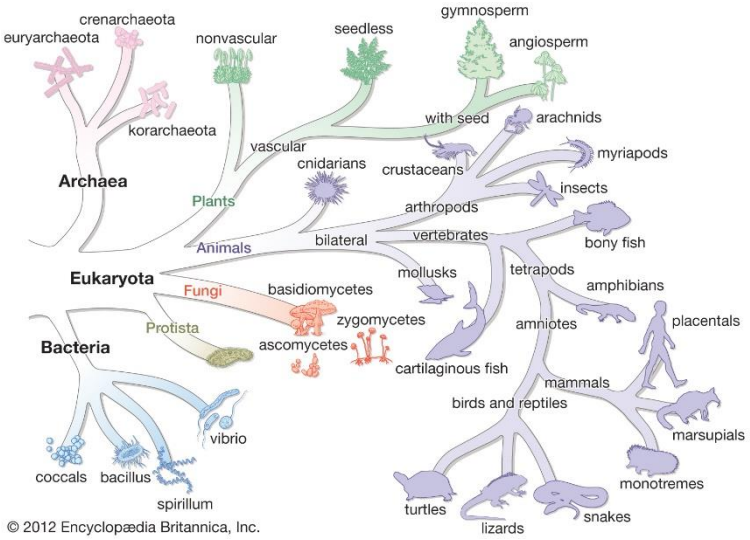
    Référence à un principe biologique indiquant qu'un organisme ou une espèce peut prendre différentes formes.
  - **Le polymorphisme et les Objets**

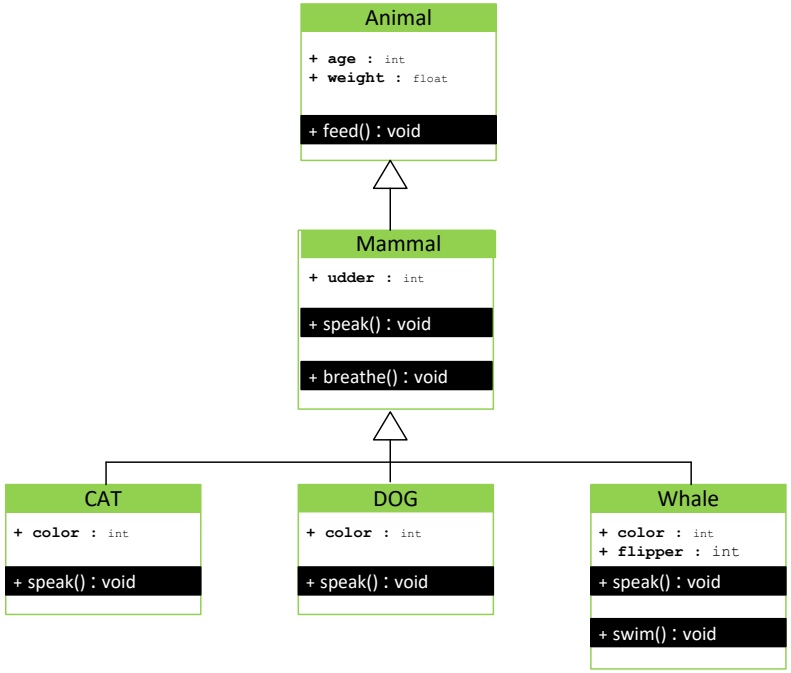
    Des **objets** peuvent **hériter** d'un certain nombre de **comportements (méthodes) et de contenus (attributs)** d'objets parents tout en pouvant spécialiser leur propre comportement /contenu

➡ **Principe de généralisation et de spécialisation**

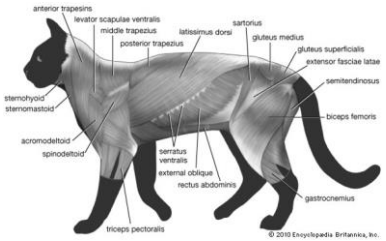
Que permet de faire POO ?

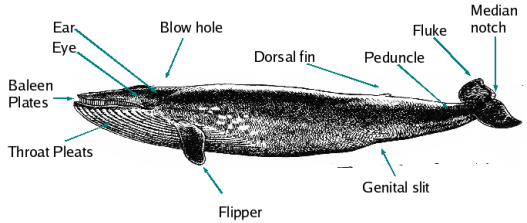
- Programmation Orientée Objet (POO) (4/7)
  - **Le polymorphisme**

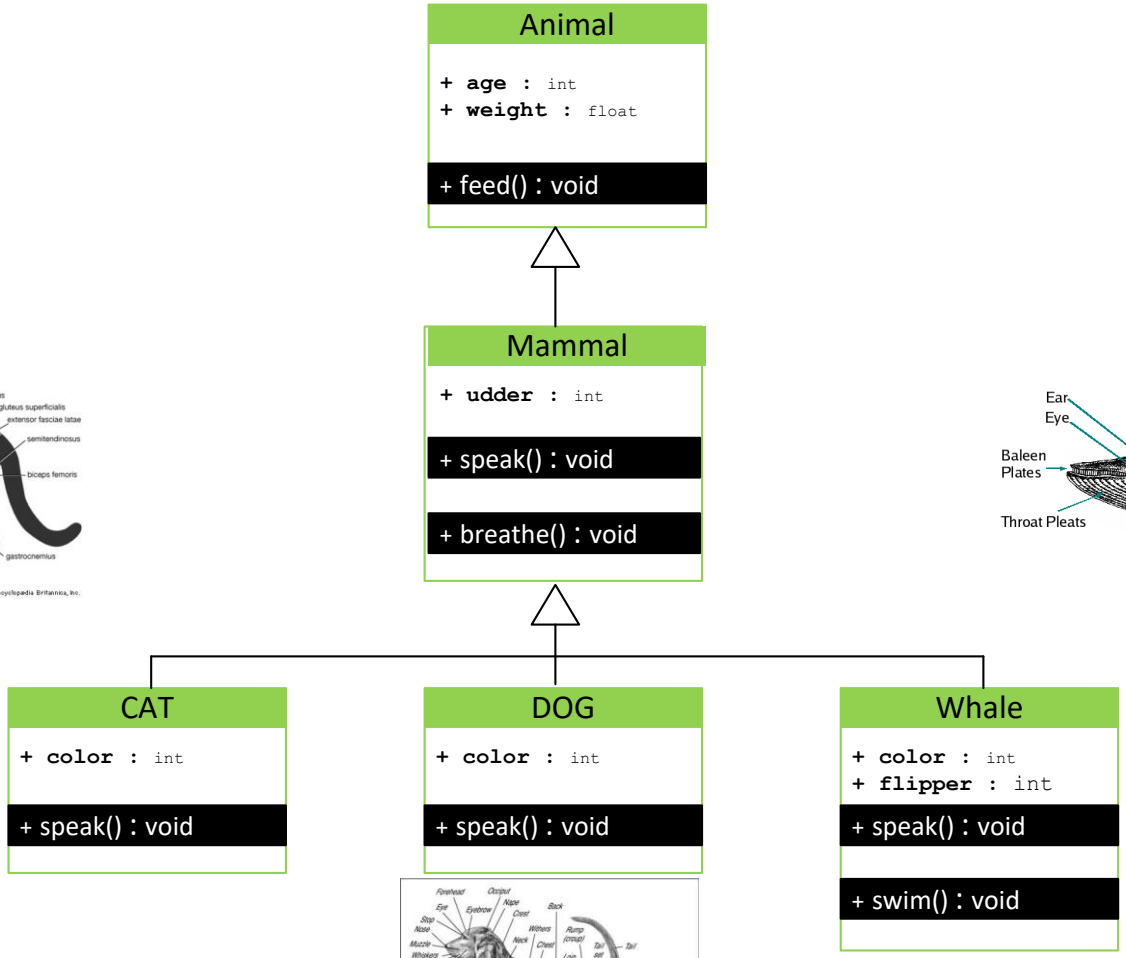


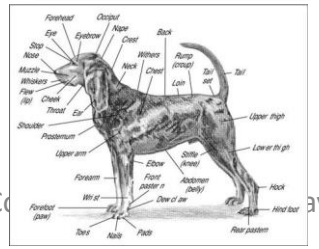


Que permet de faire POO ?





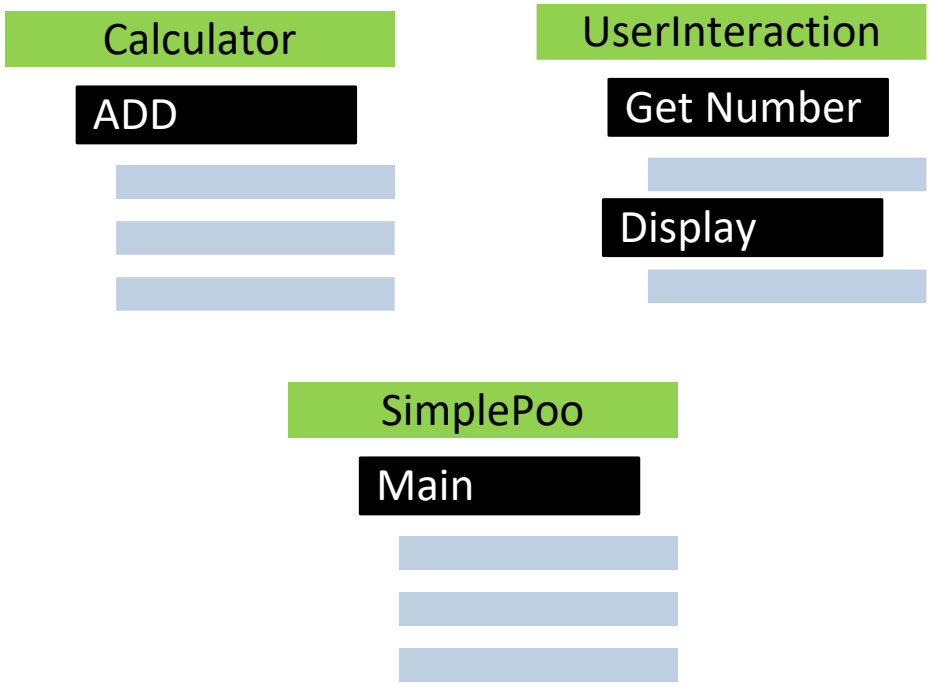




Que permet de faire POO ?

- Programmation Orientée Objet (POO) (5/7)
  - Procédure:
    - Découper le problème en objets en différentiant les responsabilités

{ Réaliser une **opération d'ajout** de deux nombres **provenant** d'un **utilisateur** et les **afficher** }



Que permet de faire POO ?

- Programmation Orientée Objet (POO) (6/7)
  - Avantages
    - Modularité
    - Abstraction, généralisation -> réutilisabilité, productivité
    - Sureté
    - Encapsulation
    - Réutilisation de code
  - Inconvénients
    - Qui est responsable de fonctionnalités transverses (sécurité/intégrité) ?
    - Peut engendrer une complexité de lecture (liée à de multiples objets)
  - Langage Orienté Objet:
    - C++, JAVA, C#, etc…

Que permet de faire POO ?

- Programmation Orientée Objet (POO) (7/7)

```
package com.course.examples.simplePoo;
import java.util.Scanner;

public class SimplePoo {
public static void main(String[] args) {
   int[] values;
   int result;

   Calculator cal = new Calculator();
   UserInteraction ui = new UserInteraction();

   values = ui.ask2Number();
   result = cal.addNumber(values[0], values[1]);
   ui.displayResult(result);
 }
}
```

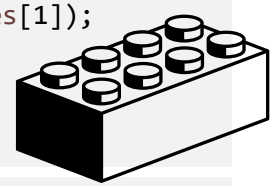
```
class Calculator {
     public int addNumber(int a, int b) {
        return a + b;
     }
}
```

```
class UserInteraction {
     public int[] ask2Number() {
           int[] result = new int[20];
           int a = 0, b = 0;
           Scanner in = new Scanner(System.in);
           System.out.println("Enter an integer: A");
           a = in.nextInt();
           System.out.println("You entered integer A: " + a
           System.out.println("Enter an integer: B");
           b = in.nextInt();
           System.out.println("You entered integer B: " + b
           result[0] = a;
           result[1] = b;
           in.close();
           return result;
     }

     public void displayResult(int result) {
           System.out.println("The result of the operation
                                                + result);
     }
}
```

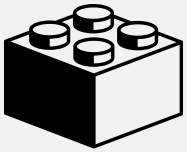
Procédural vs POO

```c
#include <stdio.h>

int addNumbers(int a, int b);

int main()
{
    int n1,n2,sum;
    printf("Enters two numbers: ");
    scanf("%d %d",&n1,&n2);
    sum = addNumbers(n1, n2);
    printf("sum = %d",sum);

    return 0;
}

int addNumbers(int a,int b)
{
    int result;
    result = a+b;
    return result;
}
```

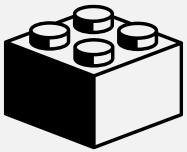
```
…
public class SimplePoo {
public static void main(String[] args) {
   int[] values;
   int result;
   Calculator cal = new Calculator();
   UserInteraction ui = new UserInteraction();
   values = ui.ask2Number();
   result = cal.addNumber(values[0], values[1]);
   ui.displayResult(result);
 }
}
```



```
class Calculator {
     public int addNumber(int a, int b) {
        …
     }
}
```



```
class UserInteraction {
      public int[] ask2Number() {
            …
     }
     public void displayResult(int result) {
            …
     }
}
```

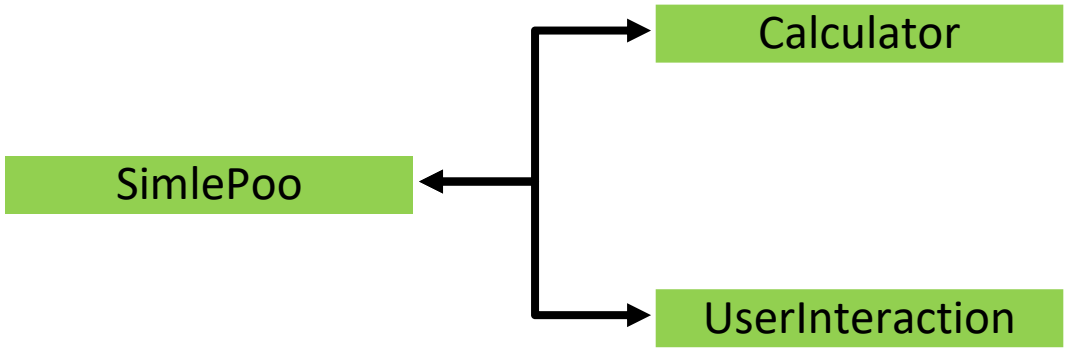


La conception POO

- **Introduction aux diagrammes de classes**
  - Comment représenter un programme en POO ?

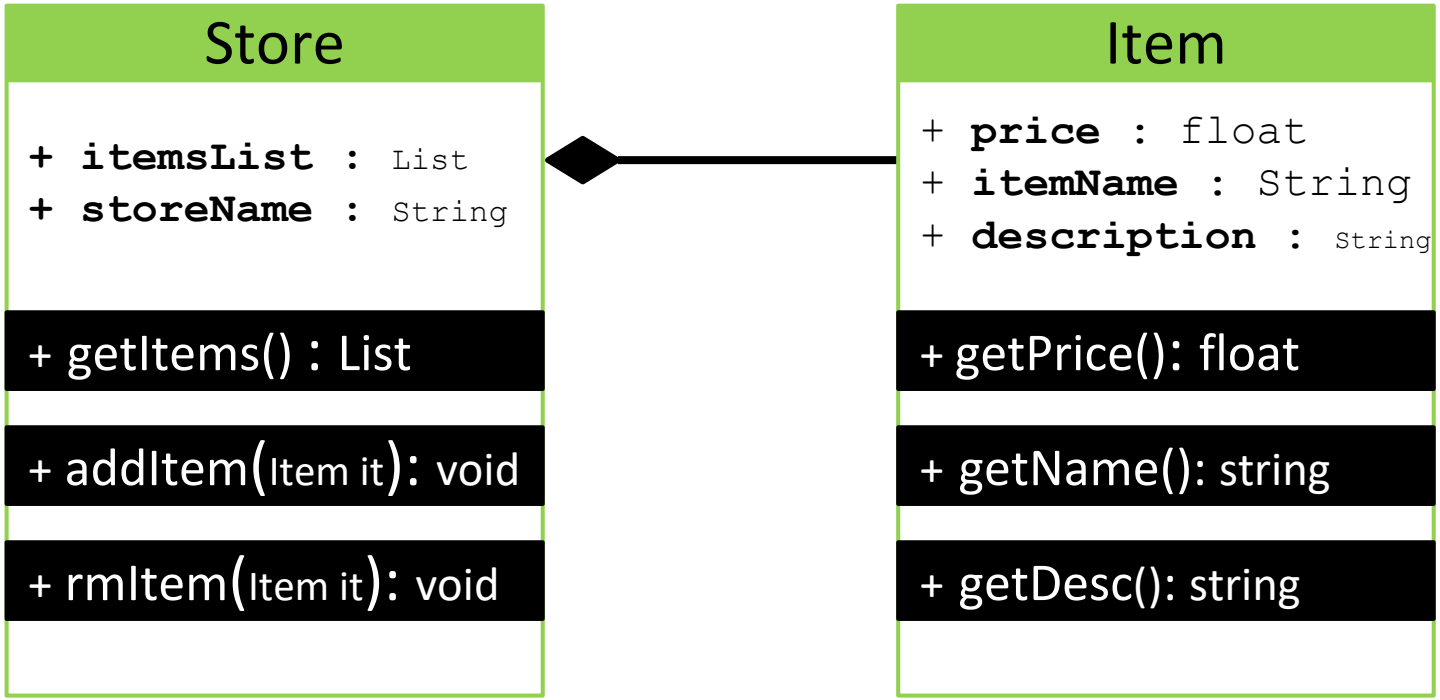
→ **Concevoir** un programme au travers de documents (**spécifications**) expliquant son fonctionnement

→ **Représenter** les **classes**, les **attributs**, les **méthodes** et les **liens** entre ces concepts (e.g formalisme UML).



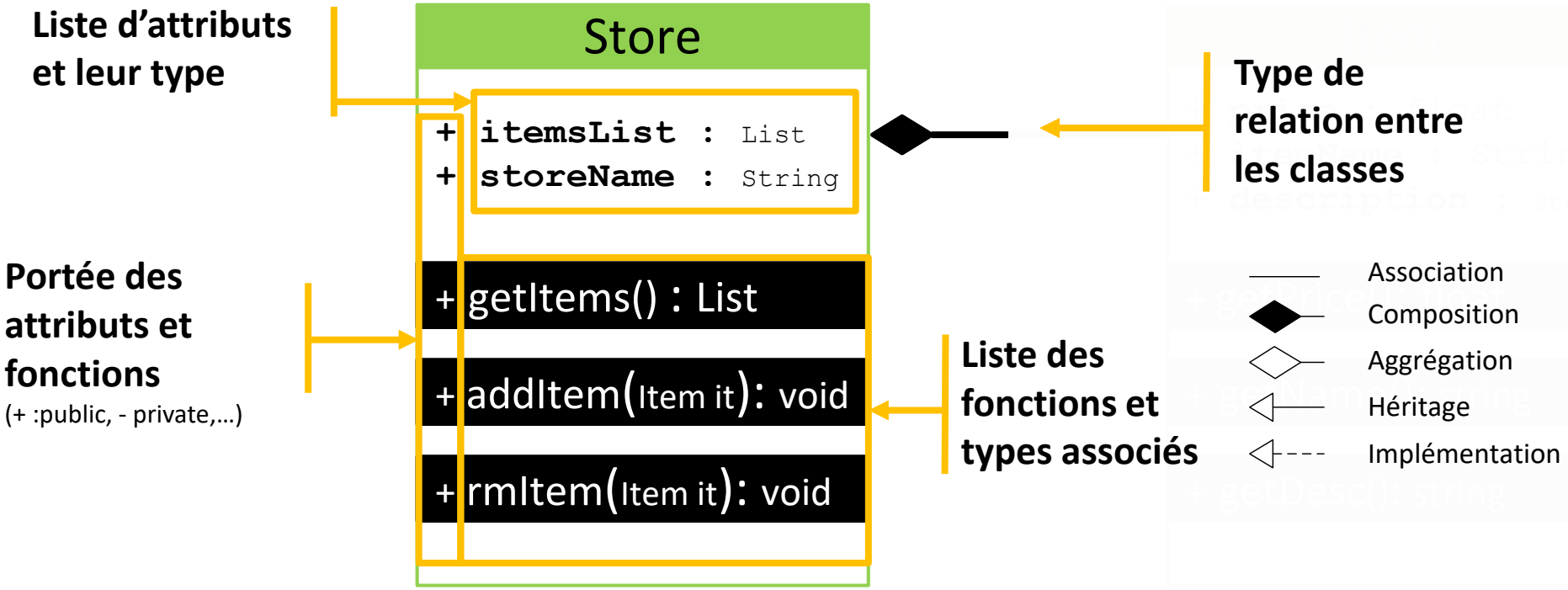
La conception POO

- **Formalisme du diagramme de classes**
  - Formalisme de description permettant de visualiser les programmes POO



La conception POO

- **Formalisme du diagramme de classes**
  - Formalisme de description permettant de visualiser les programmes POO

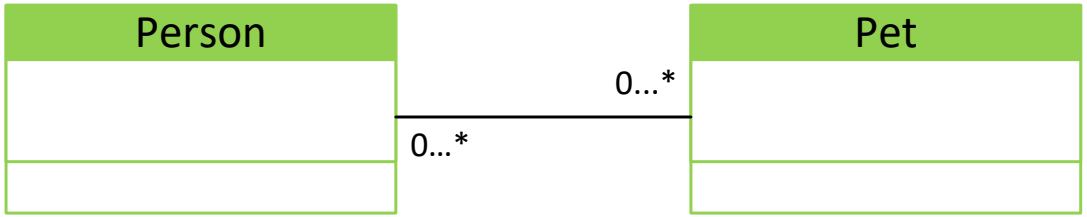


La conception POO

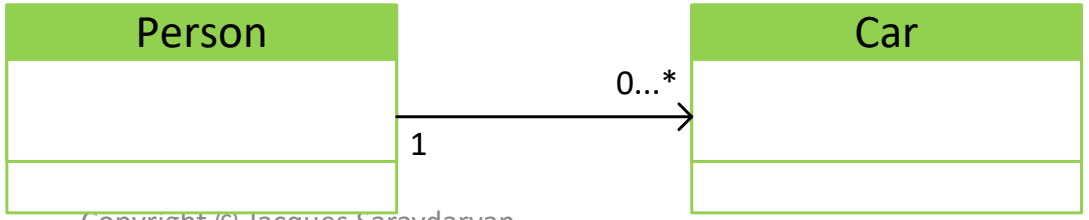
- **Formalisme du diagramme de classes**
  - **Les associations**

    Représente un lien entre deux classes, dans lequel **les classes n'influent pas sur le cycle de vie l'une de l'autre** (pas dépendance entre les classes). Les classes peuvent communiquer entre-elles

  - **Bi-directionnel:** Les deux classes sont au courants l'une de l'autre et de leur relation.



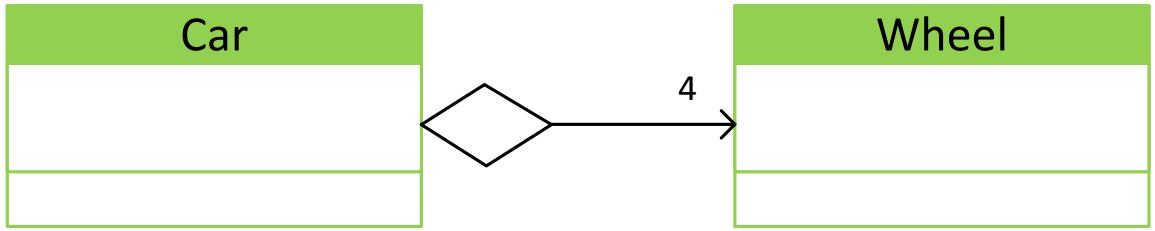
  - **Uni-directionnel:** Une seule classe connait la relation et peut utiliser cette relation



La conception POO

- **Formalisme du diagramme de classes**
  - **L'Aggrégation**

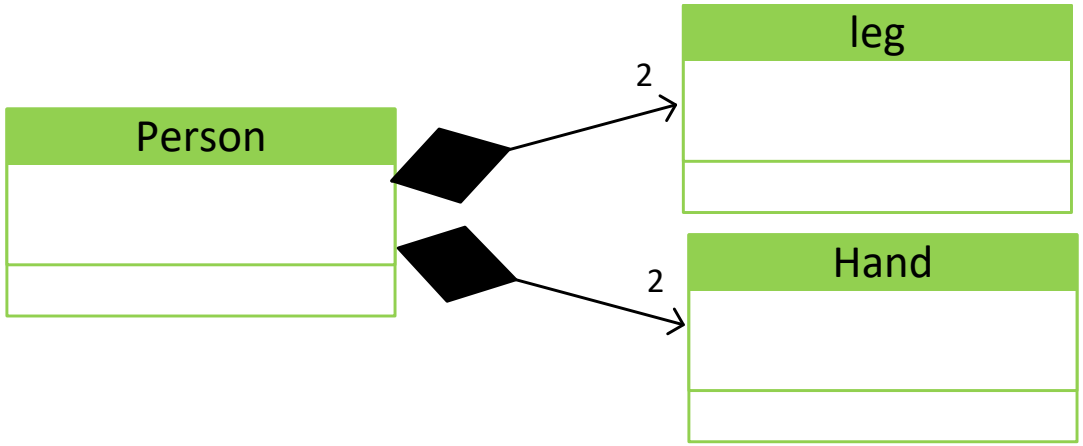
    Type spéciale d'association représentation une association de type ensemble / élément. **Le cycle de vie de " l'élément " est indépendant de " l'ensemble ".**



La conception POO

- **Formalisme du diagramme de classes**
  - **La Composition**

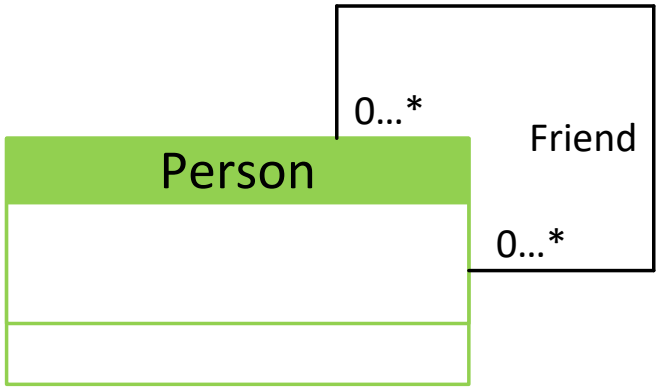
    Type spécial d'association représentant une association de type conteneur-contenu (aggrégation forte). Le contenu n'existe pas sans le conteneur. (le cycle de vie du contenu de son conteneur)

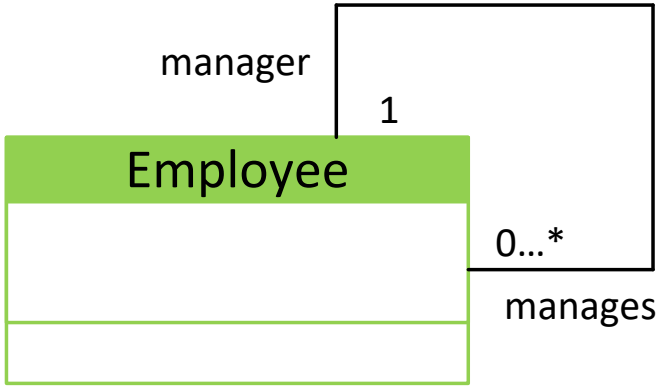


La conception POO

- **Formalisme du diagramme de classes**
  - **Association réflexive**

Cas spécial d'association permettant de lier une classe à elle même





La conception POO

- **Formalisme du diagramme de classes**
  - La cardinalité (formalisme UML)

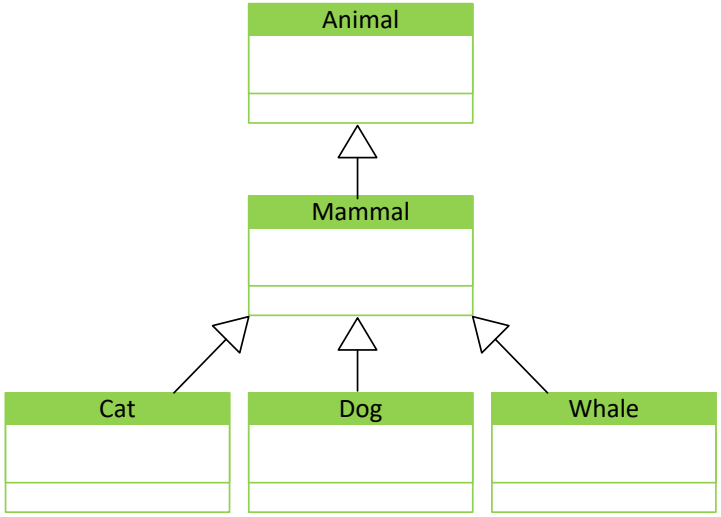
Indicator	Meaning
0..1	Zero or one
1	One only
0..*	Zero or more
*	Zero or more
1..*	One or more
3	Three only
0..5	Zero to Five
5..15	Five to Fifteen

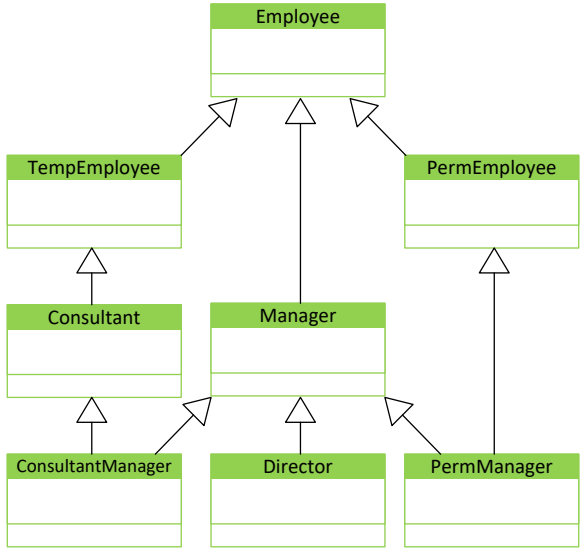
https://www.ibm.com/developerworks/rational/library/content/RationalEdge/sep04/bell/index.html

La conception POO

- **Formalisme du diagramme de classes**
  - L'Héritage

    Héritage est l'un des principes fondamentaux du POO, il **permet à une Classe d'hériter des propriétés (attributs) et des comportements (méthodes) d'une classe Parente.** Cette classe pourra également posséder ces propres propriétés et ces propres comportements. Elle pourra également redéfinir des comportements hérités.

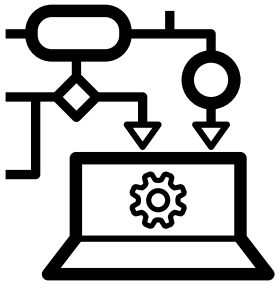




Héritage Multiple pas supporté par tous les langages

Exercice de conception

Réaliser la conception logicielle en POO représentant un garage automobile, possédant des pièces détachées des voitures à réparer et des garagistes affectés à des voitures



Le langage JAVA

Quel langage pour la POO ?

- **Liste des différents langages**

  - **Langage compilé (C, C++)**

  Le code est transformé en code machine directement interprétable par l'ordinateur

  - **Langage interprété (Python, JavaScript)**

  Le code est lu par un autre programme qui s'occupe de la traduction en code machine (traduction à la volée)

  - **Langage pré-compilé (JAVA)**

  Le code est compilé dans un code spéciale et est exécuté dans un environnement qui fait le lien avec le code machine

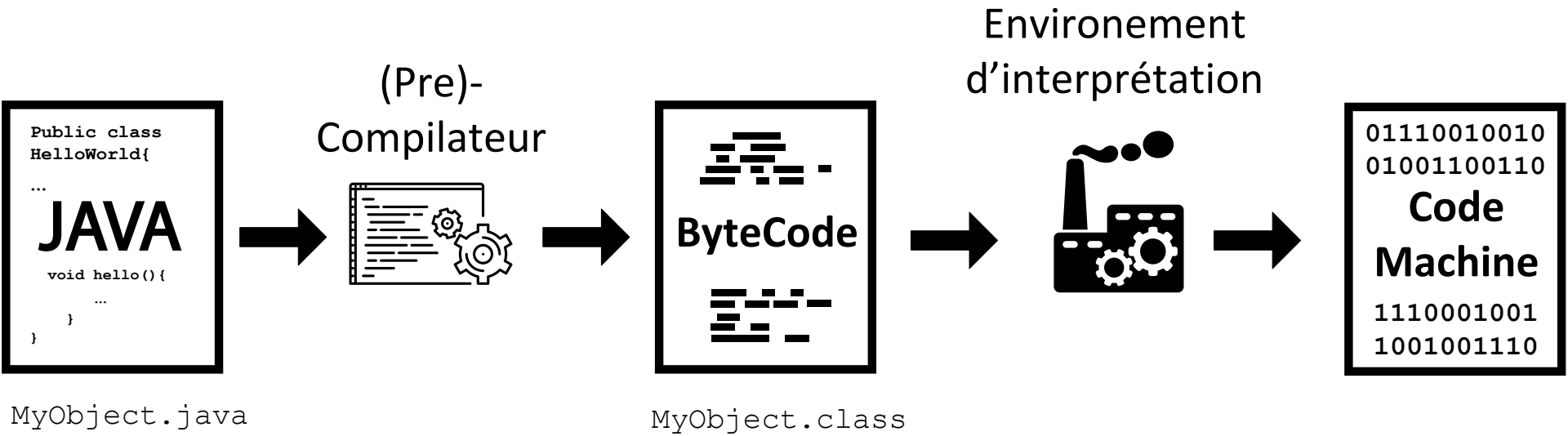
Quel langage pour la POO ?

- **Liste des différents langages**

Type de langage	Avantages	Inconvénients
Compilé (C,C++)	• Rapidité d'exécution • Taille de code	• Complexité de programmation • Faible portabilité
Interprété (Python, Javascript)	• Rapidité de développement • Changement à chaud • Portabilité	• Lenteur d'exécution • Nécessite l'interpréteur
Pré-compilé (JAVA)	• Rapidité de développement (moins que interprété) • Rapidité d'exécution (moins que compilé) • Portabilité	• Lourdeur de l'environnement

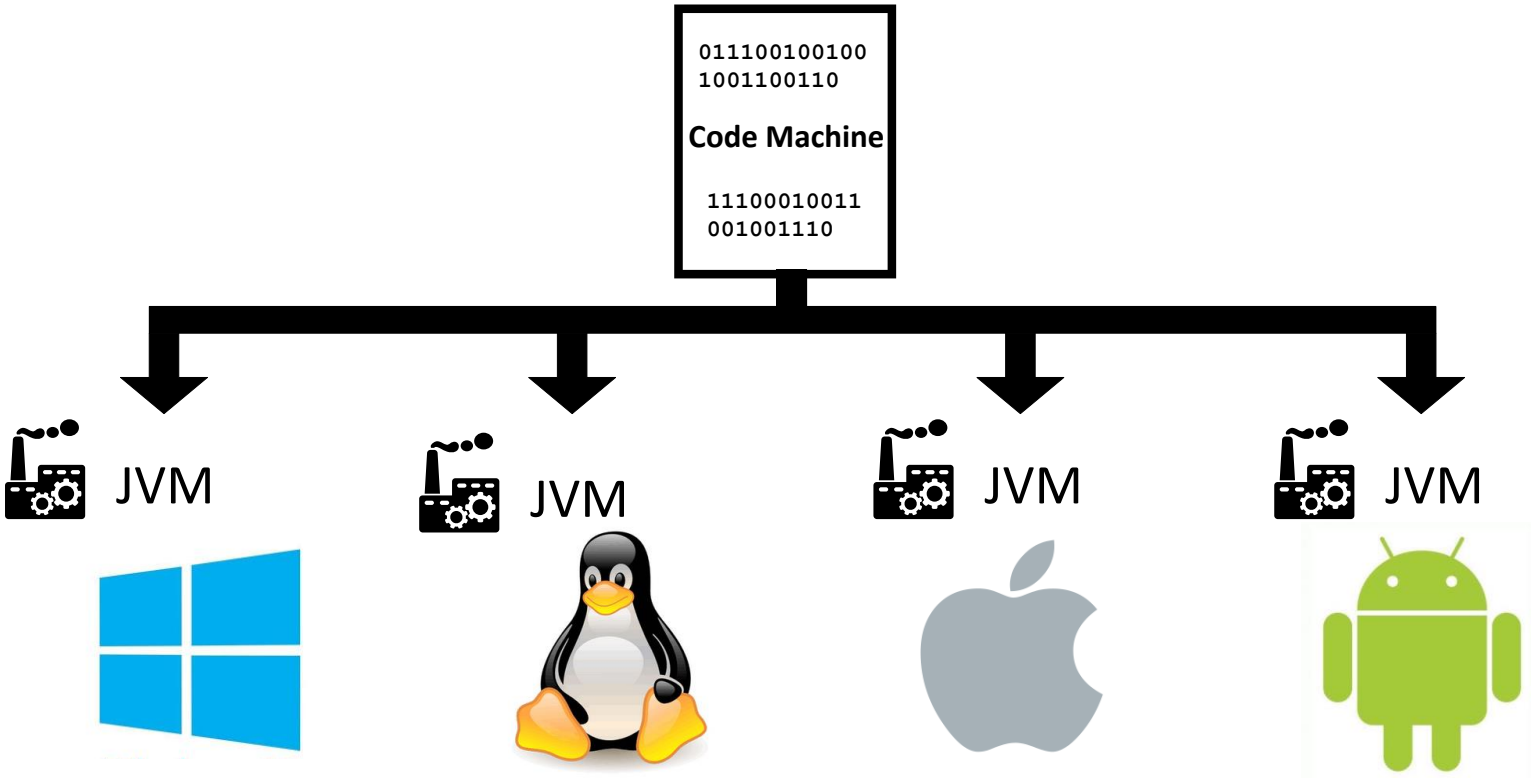
Pourquoi JAVA ?

- **Fonctionnement**



Pourquoi JAVA ?

- Fonctionnement



Pourquoi JAVA ?

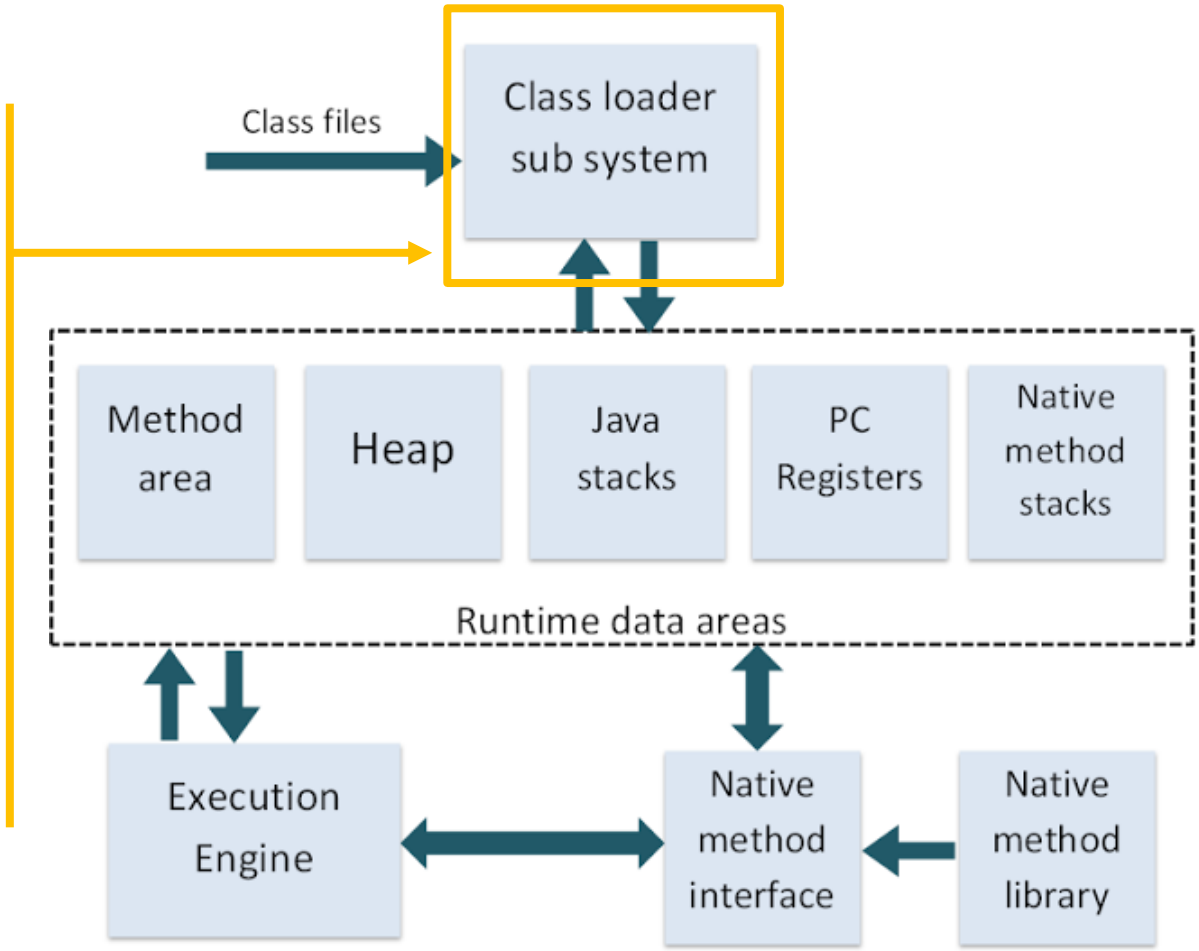
- **Propriété de JAVA**
  - Langage POO
  - De nombreux outils
  - Optimisation des performances par la JVM
  - Gestion de la mémoire automatisée (limitation et gestion des fuites mémoires par la JVM)
  - Généricité (on écrit une fois on exécute partout*)
  - Sécurité
  - Multitâches

Comment fonctionne JAVA ?

❑ JAVA et la JVM

En charge de 3 étapes:

- **Loading :** Charge le fichier class en mémoire
- **Linking:** Vérifie les instructions du Byte-Code
- **Initialization:** alloue la mémoire nécessaire au programme

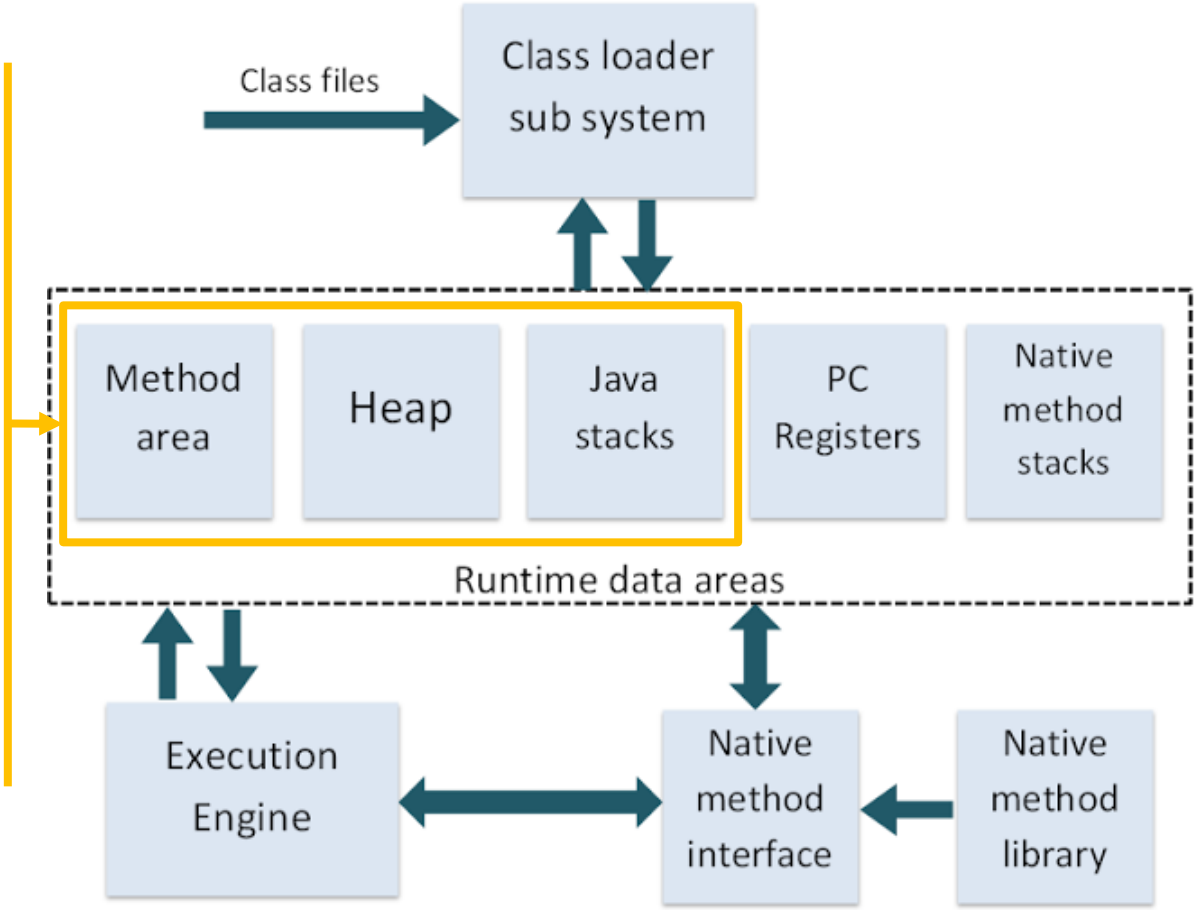


https://www.quora.com/What-is-the-architecture-of-JVM-and-responsibility-of-each-component-in-JVM-Java-virtual-machine

Comment fonctionne JAVA ?

- **JAVA et la JVM**

- **Method area :** Stocke le code des classes et des méthodes
- **Heap:** Les objets sont créés dans cette zone
- **Java Stacks:** zone d'exécution des méthodes



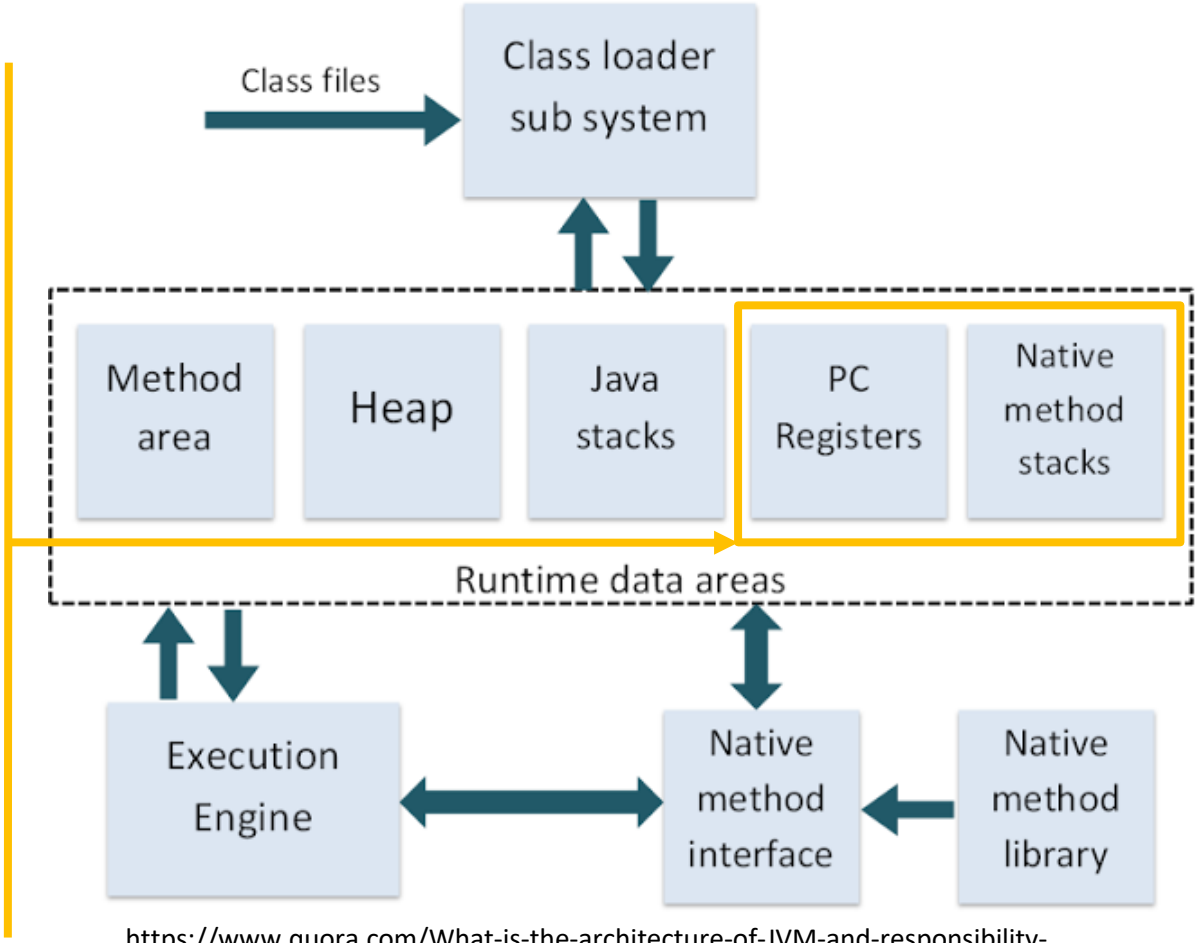
https://www.quora.com/What-is-the-architecture-of-JVM-and-responsibility-of-each-component-in-JVM-Java-virtual-machine

Comment fonctionne JAVA ?

❑ JAVA et la JVM

- **Program Counter Registers**: Stocke les adresses mémoires des instructions pour leur exécution sur les micro-processeurs

- **Native method stacks**: Zone d'exécution des méthodes natives (e.g programme C). Une méthode native est une méthode écrite dans un autre langage

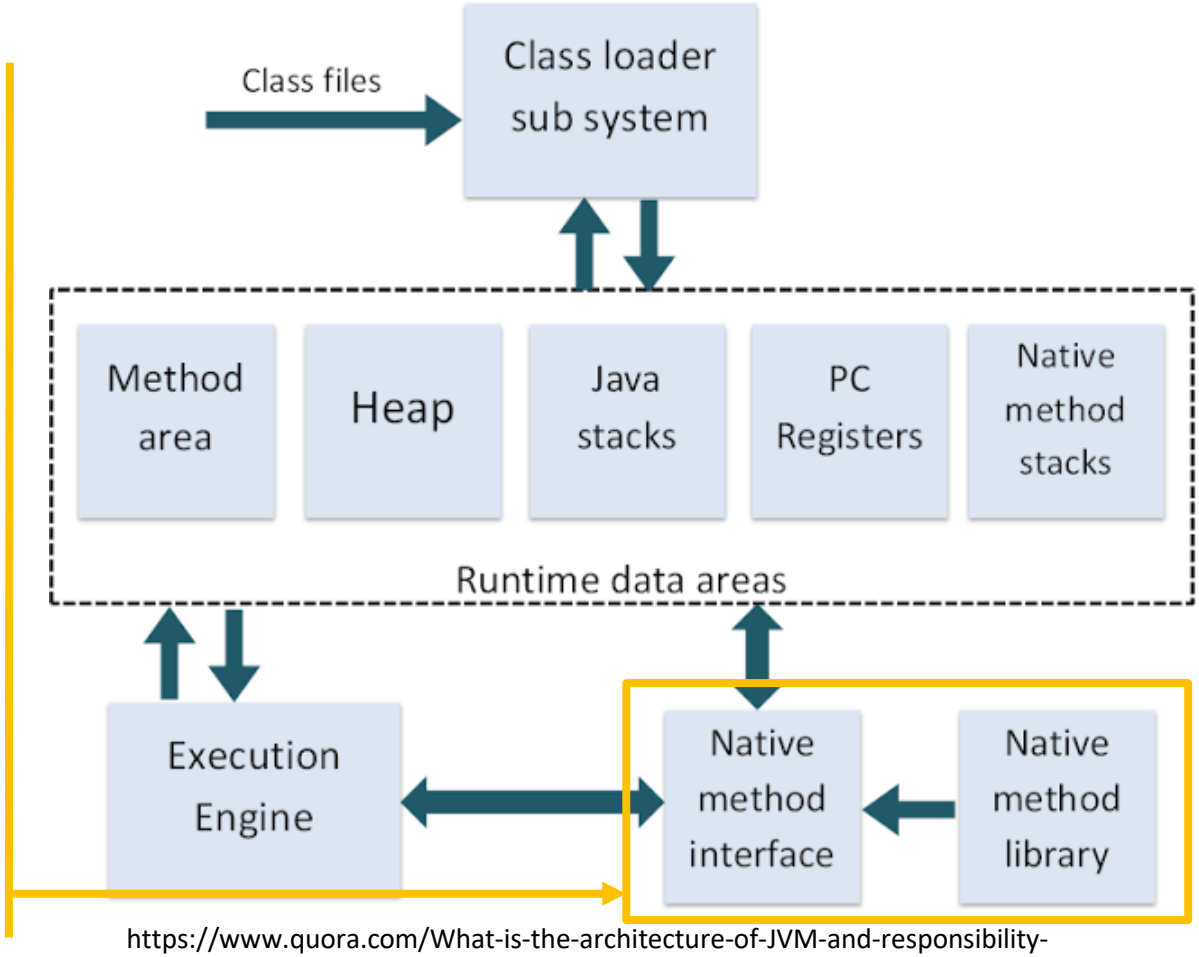


https://www.quora.com/What-is-the-architecture-of-JVM-and-responsibility-of-each-component-in-JVM-Java-virtual-machine

Comment fonctionne JAVA ?

❑ JAVA et la JVM

- **Native method interface**: Connecte les méthodes des librairies natives à la JVM. Autorise la JVM à appeler ces librairies C/C++ et d'être appelée par des librairies C/C++.
- **Native method library**: Récupère les libraires (C,C++) natives nécessaire à l' Execution Engine



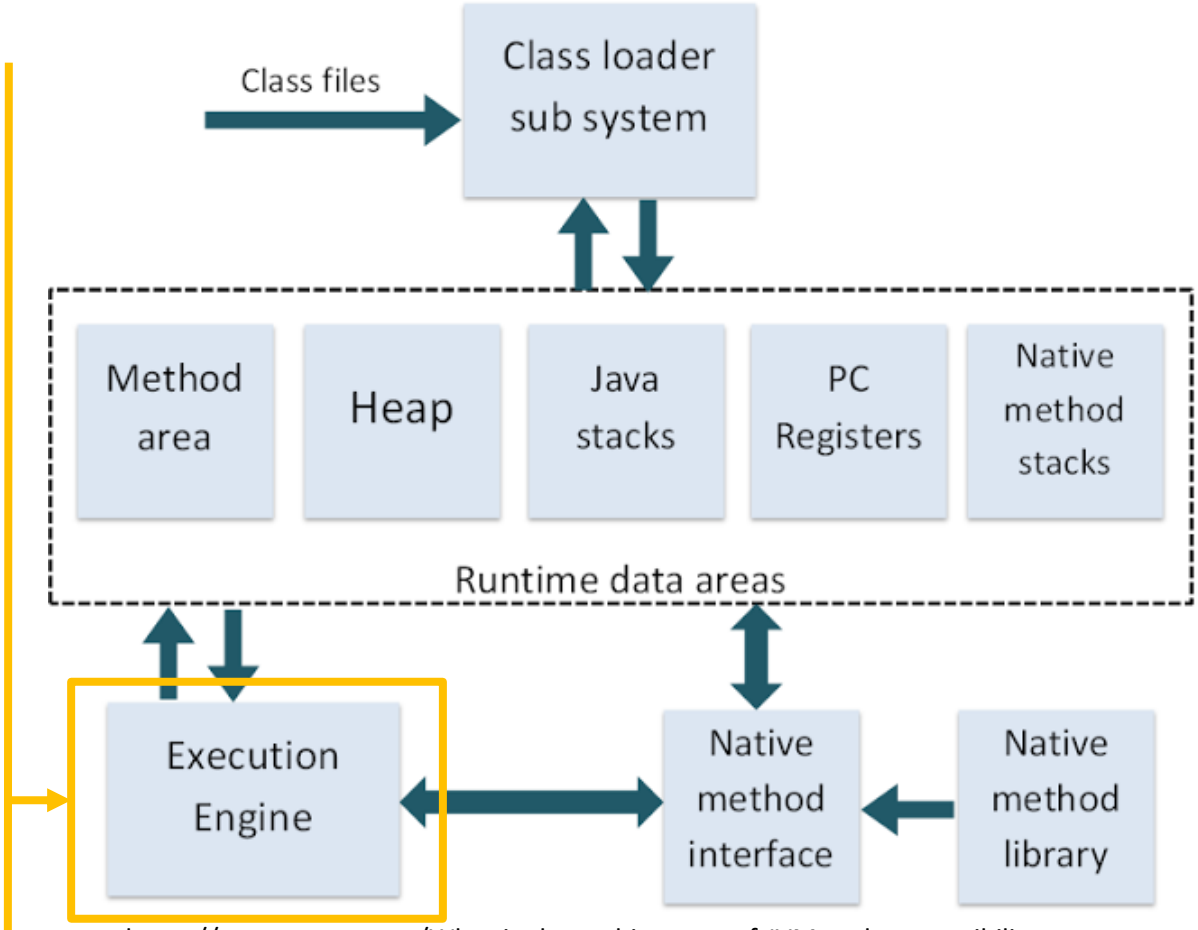
https://www.quora.com/What-is-the-architecture-of-JVM-and-responsibility-of-each-component-in-JVM-Java-virtual-machine

Comment fonctionne JAVA ?

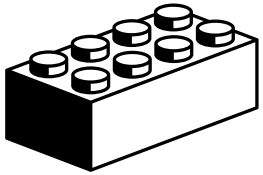
- **JAVA et la JVM**

Execution engine:

- **Convertit le Byte-Code en code machine** (Interpréteur et Juste In Time compiler).
- **Optimisation de l'exécution** en choisissant le code à interpréter et le code compiler en code machine (**JIT Compiler**)
- **Optimisation de la gestion de la mémoire** en détruisant les objets non référencés (**Garbage Collector**)



https://www.quora.com/What-is-the-architecture-of-JVM-and-responsibility-of-each-component-in-JVM-Java-virtual-machine



JAVA
les basics

Premier pas en java

❑ Exemple de code

Appartenance à un groupe de la classe, import des objets extérieurs à la classe

FirstClass.java

```
package com.course.examples.simplePoo;
import com.course.examples.simplePoo.A;

public class FirstClass {
   String myName;
   A myObjA;

   /**
    * Documentation Comment
    * @param name of the current Classe
    */
   public FirstClass(String name) {
      this.myName=name;
      myObjA=new A();
   }

   public String sayHello(){
      return "Hello I am "+myName;
   }

   public static void main(String[] args) {
      //Simple line comment
     int a;
     int b=5;
     int result;
     FirstClass fClass=new FirstClass("John");
     /*
      * Multiline
      * Comment
      */
      a=4;
      result=a+b;
      System.out.println("__ EXAMPLE __");
      System.out.println("Result of a:"+a+","+ " b:"+b+" ="+result);
      System.out.println(fClass.sayHello());
   }
}
```

L'objet FirstClass

Déclarations des attributs

Méthode spéciale pour la création de l'objet

Méthode de la classe

Méthode spéciale, point d'entrée de l'application

Premier pas en java

- **Exemple de code**

FirstClass.java

```
public class FirstClass {

   public static void main(String[] args) {
     int a;
     int b=5;
     int result;

     a=4;
     result=a+b;

     System.out.println("__ EXAMPLE __");
     System.out.println("Result of a:"+a+","
                    + " b:"+b+" ="+result);
   }
}
```

Résultat

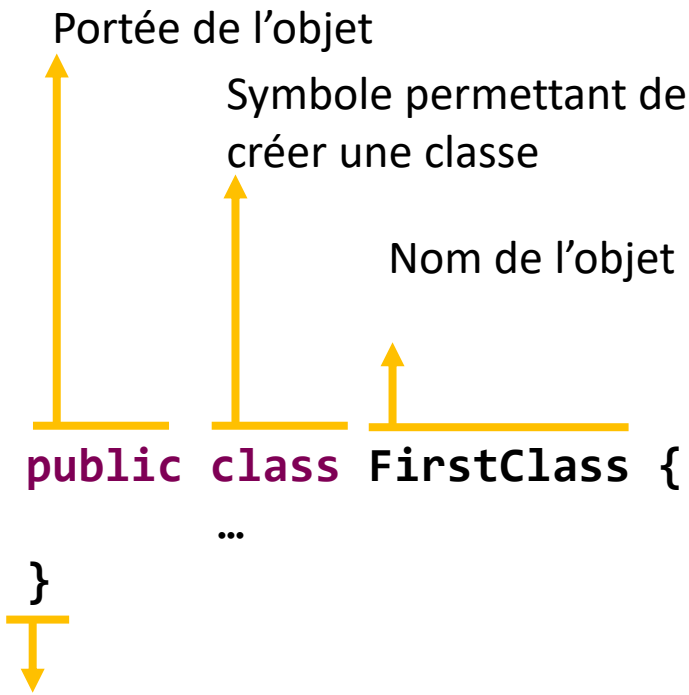
```
Result of a:4, b:5 =9
```

Premier pas en java

- **Exemple de code**

FirstClass.java

```
public class FirstClass {

    public static void main(String[] args) {
      int a;
      int b=5;
      int result;

      a=4;
      result=a+b;

      System.out.println("__ EXAMPLE __");
      System.out.println("Result of a:"+a+","
                    + " b:"+b+" ="+result);
    }
}
```



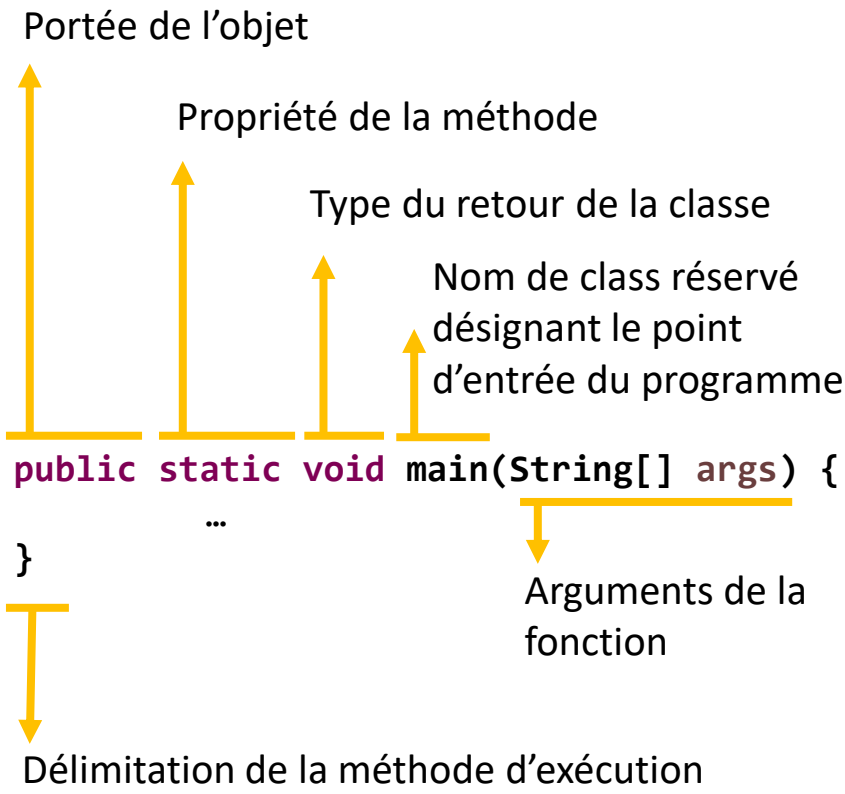
→ Déclaration d'un Objet **FirstClass**

Premier pas en java

- **Exemple de code**

FirstClass.java

```
public class FirstClass {

   public static void main(String[] args) {
      int a;
      int b=5;
      int result;

      a=4;
      result=a+b;

      System.out.println("__ EXAMPLE __");
      System.out.println("Result of a:"+a+","
                        + " b:"+b+" ="+result);
   }
}
```



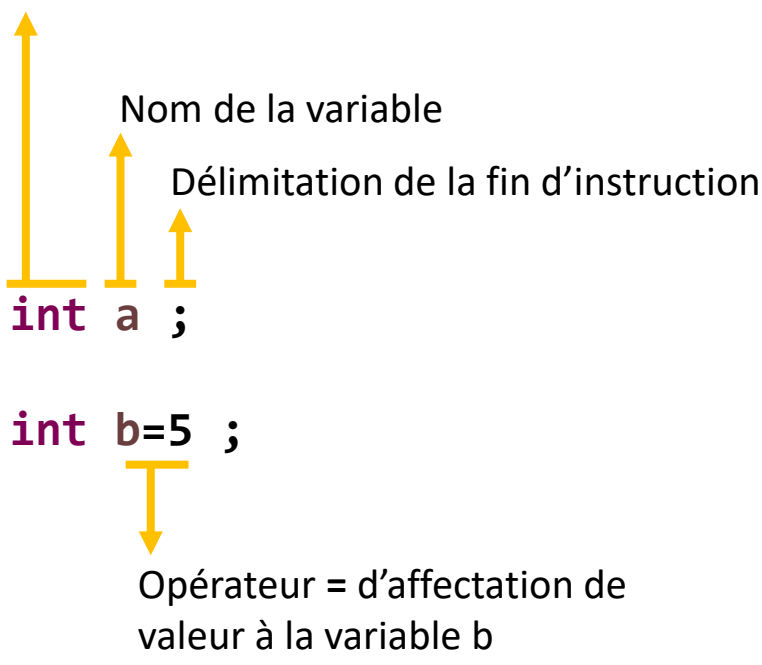
→ Méthode d'exécution de la Classe

Premier pas en java

- **Exemple de code**

FirstClass.java

```
public class FirstClass {

   public static void main(String[] args) {
     int a;
     int b=5;
     int result;

     a=4;
     result=a+b;

     System.out.println("__ EXAMPLE __");
     System.out.println("Result of a:"+a+","
                    + " b:"+b+" ="+result);
   }
}
```



→ Instruction de création et affectation de variables

Premier pas en java

- **Exemple de code**

FirstClass.java

```
public class FirstClass {

   public static void main(String[] args) {
     int a;
     int b=5;
     int result;

     a=4;
     result=a+b;

     System.out.println("__ EXAMPLE __");
     System.out.println("Result of a:"+a+","
                    + " b:"+b+" ="+result);
   }
}
```

Affectation de valeur à une variable

```
a=4 ;

result= a + b ;
```

Affectation de la valeur des variables **a** et **b** à **result**

→ Instruction affectation de valeurs à des variables de type primitif

Premier pas en java

- **Exemple de code**

FirstClass.java

```
public class FirstClass {

   public static void main(String[] args) {
     int a;
     int b=5;
     int result;

     a=4;
     result=a+b;

     System.out.println("__ EXAMPLE __");
     System.out.println("Result of a:"+a+","
                    + " b:"+b+" ="+result);
   }
}
```

Nom de la fonction native de JAVA

```
System.out.println("__ EXAMPLE __");
```

Argument de la fonction (chaine de caractères)

→ Appel d'une fonction native de JAVA

Premier pas en java

- **Exemple de code**

FirstClass.java

```
public class FirstClass {

   public static void main(String[] args) {
     int a;
     int b=5;
     int result;

     a=4;
     result=a+b;

     System.out.println("__ EXAMPLE __");
     System.out.println("Result of a:"+a+","
                      + " b:"+b+" ="+result);
   }
}
```

Nom de la fonction native de JAVA

```
System.out.println("Result of a:"+a+","
                 + " b:"+b+" ="+result);
```

Concaténation d'une chaine de caractère avec la valeur d'une variable avec l'opérateur **+**

→ Concaténation de chaines de caractères

Premier pas en java

- **Exécution de l'exemple**

Pré-compilation du fichier FirstClass. Java en FirstClass.class

```
javac FirstClass.java
```

Exécution du programme

```
java FirstClass

__ EXAMPLE __
Result of a:4, b:5 =9
```

Premier pas en java

- **Les commentaires**

FirstComment.java

```
public class FirstComment {
   private int value;
   public FirstComment() {
      // One line Comment
      value=5;
      /*
       * Multilines Comment
       */
   }
   /**
    * Java Doc Comment
    * @param name
    * @param v
    * @return
    */
   public String sayHello(String name, int v){
      return name+v;
  }
}
```

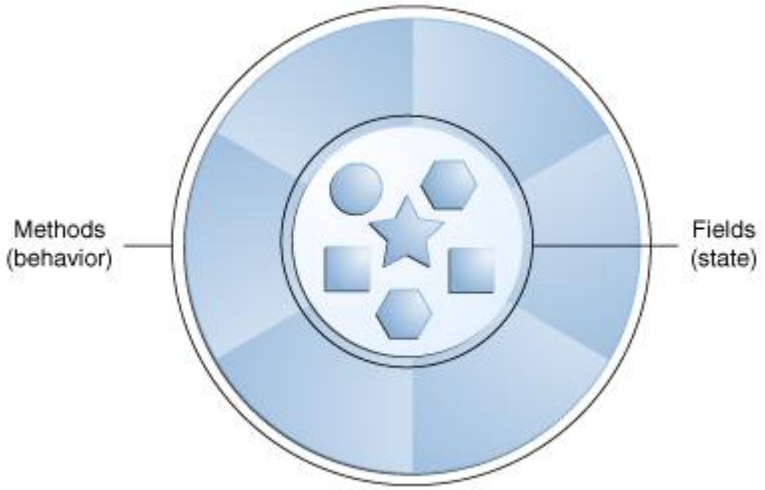
```
// One line Comment

/*
* Multilines Comment
*/


/**
    * Java Doc Comment
    * @param name
    * @param v
    * @return
    */
```

Exercice de prise en main

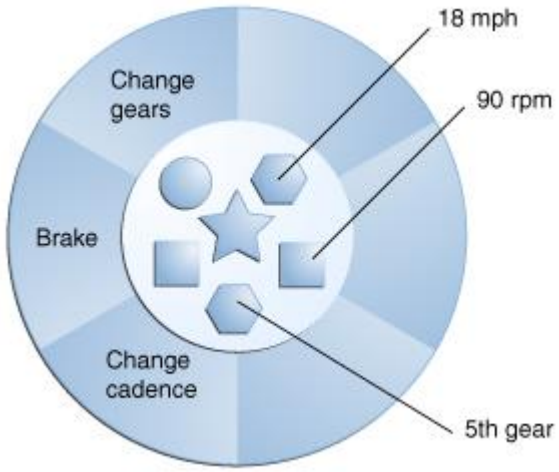
Créer un programme java permettant de définir 4 variables d'une valeur fixée à l'initialisation d'effectuer les opérations d'ajout, de multiplication. Afficher le résultat à chaque fois

Premier pas en java

- Les Classes



Objet



Vélo modélisé en Objet

https://docs.oracle.com/javase/tutorial/java/concepts/object.html

Premier pas en java

❑ Les Classes

```
class Bicycle {
    int cadence = 0;
    int speed = 0;
    int gear = 1;
    void changeCadence(int newValue) {
         cadence = newValue;
    }
    void changeGear(int newValue) {
         gear = newValue;
    }
    void speedUp(int increment) {
         speed = speed + increment;
    }
    void applyBrakes(int decrement) {
         speed = speed - decrement;
    }
    void printStates() {
         System.out.println("cadence:" +
             cadence + " speed:" +
             speed + " gear:" + gear);
    }
}
```

Déclaration de la classe (Objet)

Etats, attributs de la classe

Méthodes (et paramètres associés) définissant le comportement de l'objet,

https://docs.oracle.com/javase/tutorial/java/concepts/class.html

Premier pas en java

❑ Les Classes

```
class BicycleDemo {

public static void main(String[] args){

        Bicycle bike1 = new Bicycle();
        Bicycle bike2 = new Bicycle();

        bike1.changeCadence(50);
        bike1.speedUp(10);
        bike1.changeGear(2);
        bike1.printStates();

        bike2.changeCadence(50);
        bike2.speedUp(10);
        bike2.changeGear(2);
        bike2.changeCadence(40);
        bike2.speedUp(10);
        bike2.changeGear(3);
        bike2.printStates();
    }
}
```

Déclaration de la classe (Objet)

Déclaration de la méthode d'exécution (point d'entrée du programme)

Création de deux objets Bicycles

Appel des méthodes de l'objet 1

Appel des méthodes de l'objet 2

https://docs.oracle.com/javase/tutorial/java/concepts/class.html

Premier pas en java

- **Exécution de l'exemple**

Pré-compilation du fichier Bicycle et BicycleClassDemo

```
javac Bicycle.java BicycleDemo.java
```

Exécution du programme

```
java BicycleDemo
cadence:50 speed:10 gear:2
cadence:40 speed:20 gear:3
```

Attention :

```
java Bicycle
Erreur : la méthode principale est introuvable dans la classe
Bicycle, définissez la méthode principale comme suit :
   public static void main(String[] args)
```

Premier pas en java

- **Types et Operateurs** (1/2)

```
int a ;
float b;
```

Types primitifs

Type	Size	Min Value	Max Value	Default Value
byte	8-bits	-128	127	0
short	16-bits	-32,768	32,767	0
int	32-bits	-2^{31}	$2^{31}-1$	0
long	64-bits	-2^{63}	$2^{63}-1$	0L
float	32-bits	32-bit IEEE 754 floating point		0,0f
double	64-bits	64-bit IEEE 754 floating point		0,0d
char	16-bits	16-bit Unicode character fom '\u0000' to '\uffff'		'\u0000'
boolean	1-bit	true or false		False

Premier pas en java

```
result=a+b;
result++;
```

❑ Types et Opérateurs (2/2)

Opérateurs

Simple Assignment Operator	
=	Simple assignment operator

Arithmetic Operators	
+	Additive operator (also used for String concatenation)
-	Subtraction operator
*	Multiplication operator
/	Division operator
%	Remainder operator

Equality and Relational Operators	
==	Equal to
!=	Not equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to

Unary Operators	
+	Unary plus operator; indicates positive value (numbers are positive without this, however)
-	Unary minus operator; negates an expression
++	Increment operator; increments a value by 1
--	Decrement operator; decrements a value by 1
!	Logical complement operator; inverts the value of a boolean

Conditional Operators	
&&	Conditional-AND
||	Conditional-OR
?:	Ternary (shorthand for if-then-else statement)

Bitwise and Bit Shift Operators	
~	Unary bitwise complement
<<	Signed left shift
>>	Signed right shift
>>>	Unsigned right shift
&	Bitwise AND
^	Bitwise exclusive OR
|	Bitwise inclusive OR

Type Comparison Operator	
instanceof	Compares an object to a specified type

Premier pas en java

```
int a ;
float b;
```

- **Variables**
  - **Type de variables**
    - **Variable locale :** variables temporaires d'une portée limitée
    - **Paramètres :** variables passées à une méthode ou fonction
    - **Variable d'instance :** variable représentant l'état
    - **Variable de classe (static)**

Paramètres:
- Type primitif par valeur (copie)
- Objet (référence type) passé par valeur (la référence à l'objet est passé en valeur)
  MAIS:
  - Les valeurs de l'objets peuvent être modifiés dans la méthodes
  - Les objets modifiés dans la méthodes sont persistants
  - Si la référence à l'objet est modifié (e.g obj=null), l'objet retrouve sa référence au retour de la méthode

https://docs.oracle.com/javase/tutorial/java/javaOO/arguments.html

Premier pas en java

❑ Variables

```
class A {
   int a;
   int b;

   static String msg= "MY ONE INSTANCE VARIABLE";

    void setAB(int a_tmp, int b_tmp){
     System.out.println("A_tmp"+a_tmp+",B_tmp"+b_tmp);
     a=a_tmp;
     b=b_tmp;
    }

    int sumAB(){
     int result;
     result = a + b;
     System.out.println("A + B =" + result);
     return result;
    }
}
```

Variable d'instance

Variable de classe

Paramètres

Variable Locale

Premier pas en java

❑ Variables

```
public class ADemo {
   public static void main(String[] args) {

    A a_obj=new A();
    A a1_obj=new A();

    a_obj.setAB(10, 7);
    a_obj.sumAB();

    System.out.println("a_obj.a:"+a_obj.a);
    System.out.println("a_obj.b:"+a_obj.b);
    System.out.println("a_obj.msg:"+a_obj.msg);

    System.out.println("a1_obj.a:"+a1_obj.a);
    System.out.println("a1_obj.b:"+a1_obj.b);
    System.out.println("a1_obj.msg:"+a_obj.msg);
   }
}
```

→ Pas une bonne pratique d'appel d'attributs

Résultat

```
A_tmp10,B_tmp7
A + B =17
a_obj.a:10
a_obj.b:7
a_obj.msg:MY ONE INSTANCE VARIABLE
a1_obj.a:0
a1_obj.b:0
a1_obj.msg:MY ONE INSTANCE VARIABLE
```

Premier pas en java

❑ Instruction de base : les boucles

```
class BasicInstruction {
      public static void main(String[] args) {

      float result=0;

      for(int i=0; i <10; i++){
            result=result+1;
            System.out.println("FOR ++: Result:"+result);
      }

      for(int i=10; i> 5; i--){
             result=result-1;
             System.out.println("FOR --: Result:"+result);
      }

      int i=0;
      result=0;

      while(i<5){
             result=result+1;
             System.out.println("WHILE: Result:"+result);
             i++;
      }
      }
}
```

Résultat

```
FOR ++: Result:1.0
FOR ++: Result:2.0
FOR ++: Result:3.0
FOR ++: Result:4.0
FOR ++: Result:5.0
FOR ++: Result:6.0
FOR ++: Result:7.0
FOR ++: Result:8.0
FOR ++: Result:9.0
FOR ++: Result:10.0

FOR --: Result:9.0
FOR --: Result:8.0
FOR --: Result:7.0
FOR --: Result:6.0
FOR --: Result:5.0
```

```
result=0;
i=0;
do{
  result=result+1;
  System.out.println("DOWHILE:
                              Result:"+result);
  i++;
}while(i<5);
```

Premier pas en java

❑ Instruction de base : les conditions (1/2)

```
import java.util.Random;
public class BasicInstruction2 {
   public static void main(String[] args) {
     Random rand=new Random();
     String myKey="";
     int a =rand.nextInt(50);

     if (a<20){
         myKey="LOW";
     }

     if((a>=20) && (a<40)){
        myKey="MEDIUM";
     }else if (a>40){
         myKey="HIGH";
     }

     if(myKey.equals("LOW")){
        System.out.println("The current
                            value is low");
     }else{
        System.out.println("The current
                            value is: "+a);
   }
}
```

If (*<condition>***){**
<instruction>
}

If (*<condition1> <operator> <condition2>***){**
<instruction>
}

<operator> :
AND -> &&
OR -> ||

Premier pas en java

❑ Instruction de base : les conditions (2/2)

```
…
switch(myKey){
    case "LOW":
       System.out.println("LOW: "+a);
    break;

    case "MEDIUM":
       System.out.println("MEDIUM: "+a);
    break;

    case "HIGH":
       System.out.println("HIGH: "+a);
    break;

    default:
       System.out.println("not processed: "+a);
  }
}
```

Switch (*<variable>***) {**
    case <value>:
        <instruction>
    break;

    case <value>:
        <instruction>
    break;

    …

    default:
        <instruction>

}

Exercice de prise en main

Prise en main d'Eclipse

Exercice de prise en main

Réaliser une calculette permettant:

- Effectuer des additions, multiplications divisions sur 2 nombres
- D'indiquer si un nombre est pair ou non
- D'afficher l'historique des commandes réalisées

Premier pas en java

- **Type prédéfini simple : les tableaux** (1/3)
  - Reference data type
  - Containeur de variables (type primitif, ou autres objets) de taille fixe
  - e.g Déclaration d'un tableau:

Type des variables contenues dans le tableau

```
int     myTab[]      = {1,2,3,4,5};
char    MyTabChar[]  = {'a','b','c','d'};
```

Nom de la variable tableau. [] symbole indiquant que la variable courante est un tableau

Affectation des variables aux tableaux

```
int[]   myTab        = new int[5] ;
//or
char    MyTabChar[]  = new char[4] ;
```

Déclaration de tableaux de taille fixe (pas de valeur initiale)

Premier pas en java

- **Type prédéfini simple : les tableaux** (2/3)
  - Reference data type
  - Containeur de variables (type primitif, ou autres objets) de taille fixe
  - e.g Déclaration d'un tableau:

```
System.out.println( myTabChar[2] );
```

Récupération de la 2nd variable contenue dans le tableau

Récupération de la taille du tableau
(nbre de variables dans le tableau)

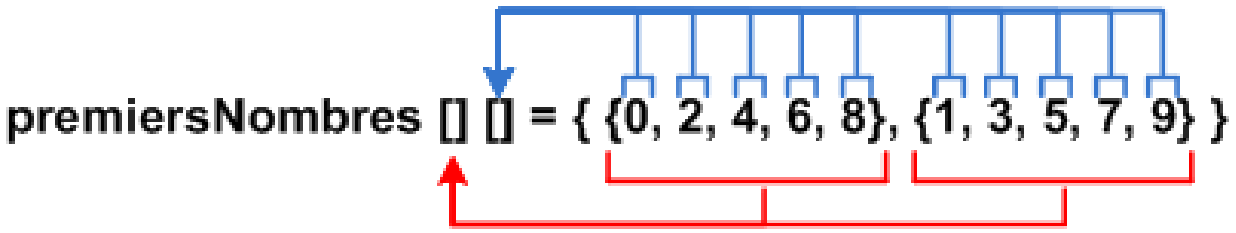
```
for(int i=0; i < myTabChar2.length;i++){
        myTabChar2[i]="A"+i;
}
```

Affectation des variables du tableau

Premier pas en java

- **Type prédéfini simple : les tableaux** (3/3)
  - Containeur de variables (type primitif, ou autres objets) de taille fixe
  - E.G Tableau multi-dimensions

```
int premiersNombres[][] = { {0,2,4,6,8},{1,3,5,7,9} };
int premiersNombres[][] = new int[2][5];
```



Nous changeons de colonne par le biais de la première paire de crochets

Nous choisissons le terme d'un tableau grâce à la deuxième paire de crochets

https://openclassrooms.com/fr/courses/26832-apprenez-a-programmer-en-java/20998-les-tableaux

Exercice de prise en main

Réaliser un programme permettant de trouver la valeur la plus grande et plus petite d'un tableau de float
(remplir votre tableau de valeurs aléatoires)

Exercice de prise en main

Réaliser un programme permettant de trier un tableau de float par ordre croissant et ordre décroissant

Premier pas en java

- Fonction d'entrée-sortie: lecture des informations du claviers

```
import java.util.Scanner;

class KeyboardInputReading {
  public static void main(String[] args) {
    double tabInput[]= new double[3];
    Scanner input = new Scanner(System.in);

     System.out.print("Enter three numbers: ");
     tabInput[0] = input.nextDouble();
     tabInput[1] = input.nextDouble();
     tabInput[2] = input.nextDouble();

     for(int i=0;i<tabInput.length;i++){
     System.out.println("["+i+"]:\t "+tabInput[i]);
     }
     input.close();
 }
}
```

Déclaration d'une lecture sur l'entrée standard du système

Attente bloquante sur l'entrée de l'utilisateur
Import directement dans une variable (plusieurs types d'import existent)

Fermeture du flux de lecture

Premier pas en java

- **Fonction d'entrée-sortie: lecture des informations d'un fichier**

Récupère des informations sur le chemin passé en paramètre (liste des fichiers,…)

```
String contents = new String(Files.readAllBytes(Paths.get(path_file)));
```

Lit tous les bytes d'un fichier
(attention lit un seul fichier)

Converti les bytes en String

Premier pas en java

- Fonction d'entrée-sortie: lecture des informations d'un fichier

```
public class FileReader {
   public static void main(String[] args) {
      try {
         String path="./src/com/course/examples/simplePoo/";
         String path_file=path+"myFile.txt";
         // Reading file into String in one line in JDK 7
         String contents = new String(Files.readAllBytes(Paths.get(path_file)));
         System.out.println("Contents (Java 7) : " + contents);

         // Reading file into String using proper character encoding
          String fileString =
                  new String(Files.readAllBytes(Paths.get(path_file)),
                             StandardCharsets.UTF_8);
     System.out.println("Contents (Java 7 with character encoding ) : " + fileString);

         // It's even easier in Java 8, lambda expression
         Files.lines(Paths.get(path_file), StandardCharsets.UTF_8).forEach(
             System.out::println
         );
} catch (IOException e) {
  // in case of error
 e.printStackTrace();
}}}
```

Premier pas en java

- **Fonction d'entrée-sortie: Ecriture d'un fichier**

Création d'un flux d'écriture bufferisé

Création d'un flux d'écriture (Character Stream)

```
PrintWriter outputStream = new PrintWriter(new FileWriter(path + "myFile2.txt"));
outputStream.println("This is a first line to write");
…
outputStream.close();
```

Fermeture du flux d'écriture

Ecriture d'une chaine de caractères suivie d'un saut de ligne dans le fichier

Premier pas en java

- Fonction d'entrée-sortie: Ecriture d'un fichier

```
public class FileWriteSample {

   public static void main(String[] args) {
      String path = "./src/com/course/examples/simplePoo/";

      PrintWriter outputStream = null;
      try {
         outputStream = new PrintWriter(new FileWriter(path + "myFile2.txt"));
         outputStream.println("This is a first line to write");
         outputStream.println("And a second...");
      } catch (IOException e) {
           System.err.println("Error occurs...");
      } finally {
         if (outputStream != null) {
             outputStream.close();
         }
      }
   }
}
```

Bonnes pratiques et conventions

- **Convention:**
  - https://www.jmdoudoux.fr/java/dej/chap-normes-dev.htm

- **Multiples références** quelques-unes des principales
  - http://gee.cs.oswego.edu/dl/html/javaCodingStd.html
  - http://www.javapractices.com/home/HomeAction.do

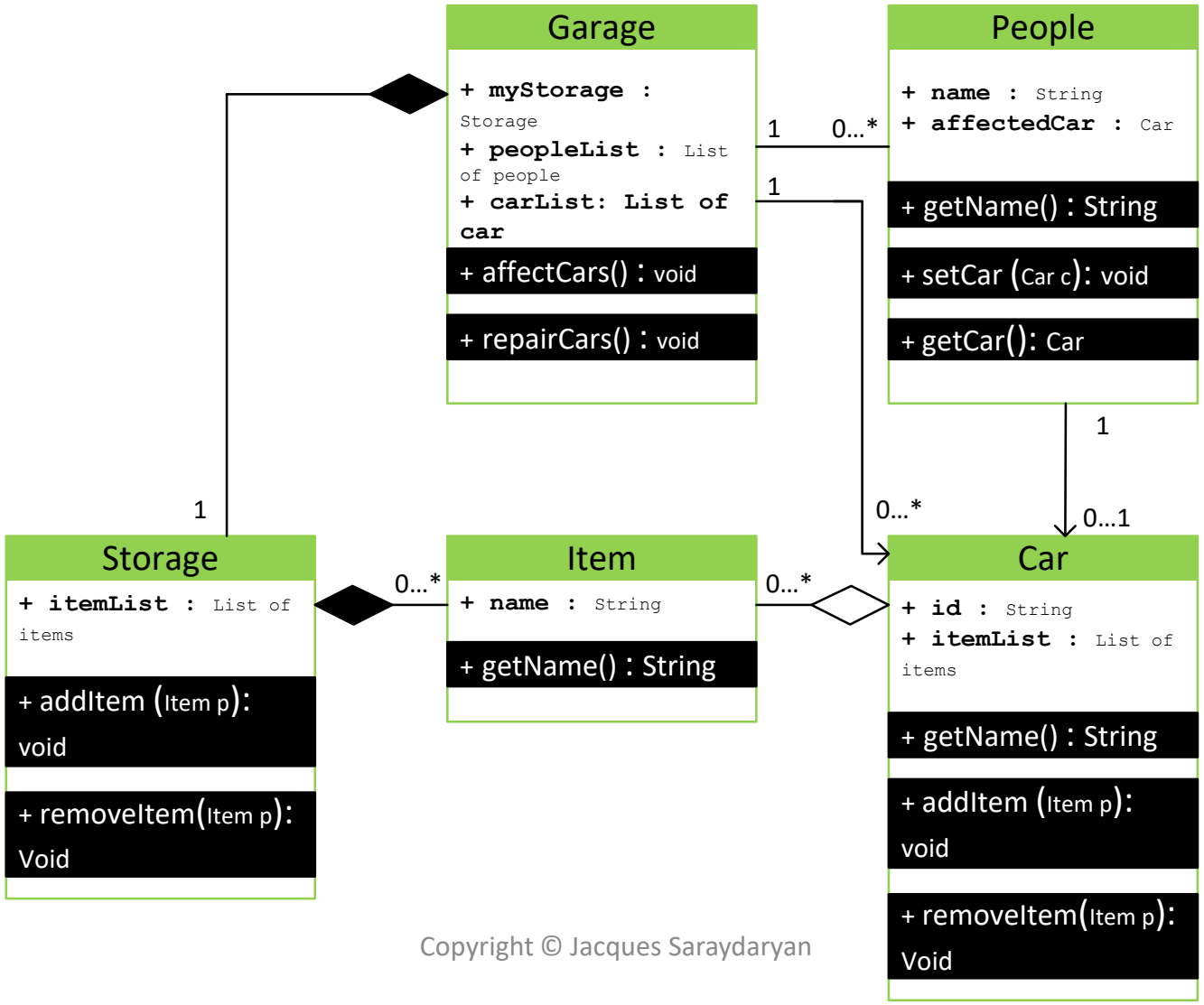
Exercice de prise en main

Réaliser le programme de l'exercice de conception précédent:

Réaliser la conception logiciel en POO représentant un garage automobile, possédant des pièces détachées des voitures à réparer et des garagistes affectés à des voitures pouvant réparer la voiture

Chaque item a également comme propriété un boolean isBroken

Exercice de prise en main

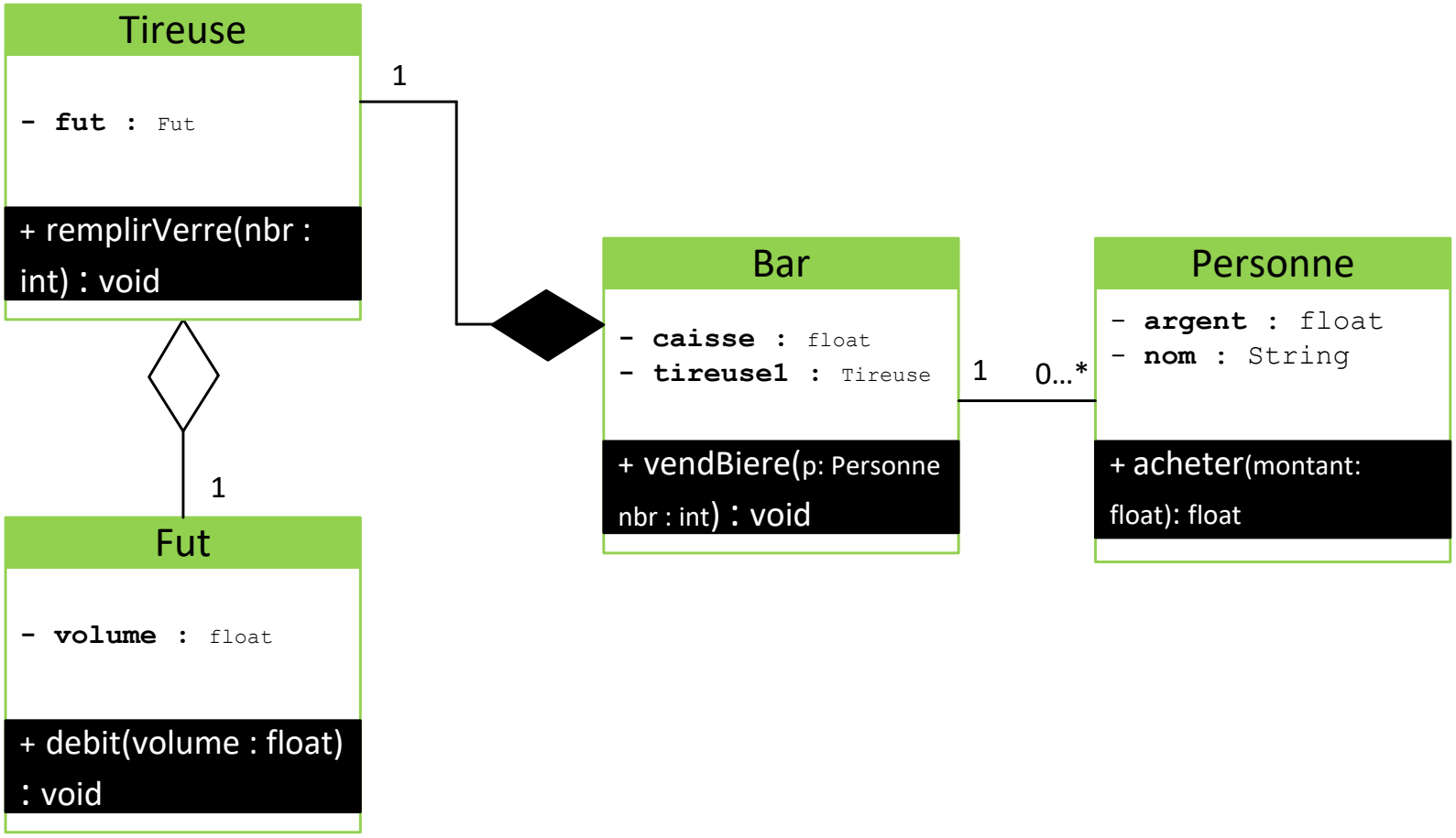


Exercice de prise en main

Réaliser le programme de l'exercice de conception précédent:

Réaliser la conception logiciel en POO représentant un bar possédant une tireuse (qui possède un fut). Des personnes viennent acheter des bières au bar

Exercice de prise en main



Questions ?

References

References

- **Web references**
  - Classes package
    - https://docs.oracle.com/javase/tutorial/java/javaOO/classdecl.html
  - Héritage / interface / classe abstraite
    - https://docs.oracle.com/javase/tutorial/java/concepts/inheritance.html
    - https://docs.oracle.com/javase/tutorial/java/IandI/index.html
    - https://programming.guide/java/clone-and-cloneable.html
    - https://dzone.com/articles/java-interface-vs-abstract-class
    - https://www.javaworld.com/article/2077421/learn-java/abstract-classes-vs-interfaces.html
  - Uml/Diagramme de classe
    - http://users.teilar.gr/~gkakaron/oose/04.pdf
    - https://www.ibm.com/developerworks/rational/library/content/RationalEdge/sep04/bell/index.html
  - Collection
    - https://www.mainjava.com/java/core-java/complete-collection-framework-in-java-with-programming-example/
    - http://tutorials.jenkov.com/java-collections/index.html
    - http://www.javapractices.com/topic/TopicAction.do?Id=65
  - Générique
    - https://docs.oracle.com/javase/tutorial/java/generics/types.html
  - Exception
    - https://www.jmdoudoux.fr/java/dej/chap-exceptions.htm
  - Concurrence
    - https://docs.oracle.com/javase/tutorial/essential/concurrency/
  - I/O
    - https://docs.oracle.com/javase/tutorial/essential/io/charstreams.html
  - Annotation
    - https://docs.oracle.com/javase/tutorial/java/annotations/index.html
    - https://www.jmdoudoux.fr/java/dej/chap-annotations.htm

References

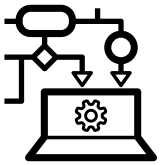
- **Web references**
  - Maven
    - https://maven.apache.org/guides/getting-started/
    - https://java.developpez.com/tutoriels/java/maven-book/
    - http://igm.univ-mlv.fr/~dr/XPOSE2010/Apache_Maven/introduction.html
    - https://mermet.users.greyc.fr/Enseignement/CoursPDF/maven.pdfTests
  - Tests
    - http://www.test-recette.fr/tests-techniques/
  - JVM
    - https://www.geeksforgeeks.org/jvm-works-jvm-architecture/ ->TB
    - https://javatutorial.net/jvm-explained
    - https://www.cubrid.org/blog/understanding-jvm-internals/

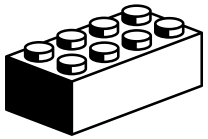
- **Livre / autres ressources**
  - Programmation Orientée Objet en Java, F. Perrin, CPE Lyon
  - Kathy SIERRA et Bert BATES pour leur ouvrage Java -Tête la Première (O'Reilly edition)

References

creative commons

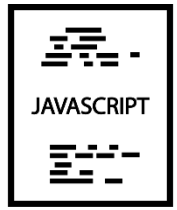

Created by Demograph
from the Noun Project


Created by H Alberto Gongora
from the Noun Project


Created by jon trillana


Created by lastspark
from Noun Project


Created by Aha-Soft
from Noun Project



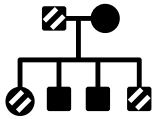
Created by AlfredoCreates.com
from Noun Project


Created by Trần Quang Hiển
from the Noun Project

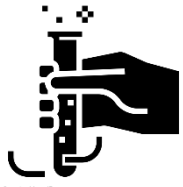

Created by Aldric Rodriguez
from the Noun Project


Created by parkjisun
from the Noun Project

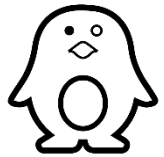

Created by priyanka
from the Noun Project


Created by dDara


Created by Opher Aloni
from the Noun Project


Created by dDara
from Noun Project


Created by Krisada
from Noun Project


Created by Delwar Hossain
from Noun Project


Created by Mohammed Tarequl Alam
from Noun Project


Created by Delwar Hossain
from Noun Project

CPE LYON
ÉCOLE SUPÉRIEURE
DE CHIMIE PHYSIQUE ÉLECTRONIQUE
DE LYON

Jacques Saraydaryan

Jacques.saraydaryan@cpe.fr