**Les variables, This, la surcharge**

**Les variables**

**Une variable est un emplacement en mémoire dans lequel on stocke une valeur et désignée dans le programme par un nom.**

**Il existe 3 types de variables**

|  |  |  |
| --- | --- | --- |
| **Attribut** | **Paramètre** | **Variable locale** |
| Variable qui se trouve dans l’état d’une classe | Variable transmise à un constructeur ou une méthode | Variable déclarée à l’intérieur d’un constructeur ou d’une méthode |
| Cette variable porte sur toute la classe (le constructeur et les méthodes de la classe) | Cette variable porte dans le constructeur ou la méthode recevant le paramètre | A partir de sa déclaration jusqu’à } qui suit |
| La durée de vie d’une variable d’un attribut est jusqu’à la mort de l’objet | La durée de vie d’une variable d’un paramètre est jusqu’à la fin de l’exécution du constructeur ou de la méthode | La durée de vie de cette variable est dans tout le bloque dans laquelle elle est déclaré |

**Les variables primitives prennent une valeur**

**Les variables de référence prennent la référence d’un objet (d’une instance) si l’instance est changée la valeur de la variable change même s’il n’est pas mentionné.**

**Voir fin de documents**

**THIS**

* Désigne l’objet courant
* Dans le constructeur il fait référence à l’objet que l’on construit
* Dans une méthode fait référence a l’objet sur lequel la méthode a été appelée
* Son utilité distinguer les attributs des l’objet des paramètres des méthodes

**La surcharge**

**Signature = nom de la méthode + liste ordonnée des types des paramètres**

***Une signature est ce qui caractérise une méthode en java on peut donc pas avoir de méthodes avec le même nom et les mêmes paramètres cependant on peut avoir 2 méthodes ou plus avec le même nom si au moins un des paramètres diffères***

Lorsque 2 méthodes d’une classe ont le même nom, on parle de surcharge de méthodes

Il est aussi possible de faire une surcharge du constructeur, il suffit que les constructeur et des signatures différentes.

On peut utiliser this(....) qui écris comme ca permet de donné une donné tout en invoquant un autre constructeur de la classe par exemple :

**public class Cercle{**

**private double rayon;**

**public Cercle (double rayon){**

**this.rayon = rayon;**

**}**

**public Cercle(){**

**this(1.0);**

|  |
| --- |
| **public class TestCercle{ public static void main (String[] args){**  **Cercle cercle1 = new Cercle();**  **Cercle cercle2 = new Cercle(3.0);**  **System.out.println(cercle1.getRayon());**  **System.out.println(cercle2.getRayon()); }**  **}** |

Avant l’assignation i=j Assignation Après l’assignation i=j

|  |
| --- |
| **1** |

|  |
| --- |
| **2** |

|  |
| --- |
| **2** |

|  |
| --- |
| **2** |

i

**i=1;**

**j=2;**

**i=j;**



j

➢ Après l’assignation, la variable **i** contient la même valeur que la variable **j**

14

15

Avant l’assignation

c1=c2

Après l’assignation

c1=c2

c1

c2

rayon = 9.0

:

Cercle

rayon = 5.0

Cercle

:

c1

c2

rayon = 9.0

Cercle

:

rayon = 5.0

:

Cercle

Assignation



**c1 = new Cercle(5.0);**

**c2 = new Cercle(9.0);**

**c1 = c2;**