



# Programmation orientée objet avancée

## *Introduction*

# Programmation orientée objet avancée

**Code du cours sur Moodle**

**JavaB219**

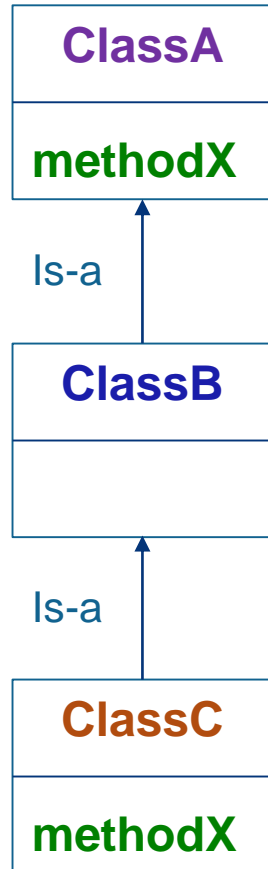
# Programmation orientée objet avancée

## 1. Prérequis

# Prérequis

- Encapsulation – Information Hiding
  - Variables d'instance privées
  - Getters/setters (éventuels) publiques
- Liens entre classes
  - Relations 1 à N
    - Une variable d'instance de type référence dans une classe
    - Un tableau d'objets dans l'autre classe
  - Relations N à N
- Héritage
- Polymorphisme
- Variable et méthode de classe (static)
- Classe abstraite et interface

# Polymorphisme



**ClassA** a = ... ; // initialisation de a  
**a.methodX();** ⇒ OK à la compilation!

*Quelle méthode sera appelée?*

⇒ *Dépend de l'initialisation de l'objet a*

## Exemples

a = new **ClassA();**

⇒ **methodX** de la **ClassA** qui sera exécutée

a = new **ClassB();**

⇒ **methodX** de la **ClassA** qui sera exécutée

a = new **ClassC();**

⇒ **methodX** de la **ClassC** qui sera exécutée

# Interface

```
public interface InterfaceX
```

```
{ void methodX();
```

```
...
```

```
// déclarations de méthodes (sans implémentation)
```

```
}
```

```
public class ClassA implements InterfaceX
```

```
{ void methodX() {... // implémentation }
```

```
...
```

```
// redéfinir TOUTES les méthodes de InterfaceX
```

```
}
```

## Utilisation

**On peut déclarer une variable de type interface !!!**

Exemple : **InterfaceX** variable = ... ;

variable.methodX();

**A condition que *variable* soit instanciée par un objet d'une classe qui implémente InterfaceX, par exemple: *variable* = new ClassA();**

# Programmation orientée objet avancée

**1. Prérequis**

**2. Contenu du cours**

# Contenu du cours

- Gestion des exceptions
- Collections génériques d'objets
- Processus parallèles (threads)
- Gestion des événements
- **Architecture des applications**
  - Découpe en couches
- **Accès (en lecture/écriture) à une base de données**
- **Design Patterns**
- Validations des formulaires
- Tests unitaires
- Streams



# Contenu du cours

- 20h de théorie
- 40h de labo
  - Sur IntelliJ

# Programmation orientée objet avancée

1. Prérequis
2. Contenu du cours
3. Evaluation

- Examen
  - Programme Java
    - Par 2 étudiants
  - Lien FACULTATIF avec le sujet du cours d'analyse
  - Attention particulière à
    - Découpe en couches
    - Accès à une BD relationnelle
  - + examen oral
    - Sur n'importe quelle ligne de code

# Evaluation

- TJ
  - Interro sur Design Patterns
    - Lors du 1<sup>er</sup> cours après Pâques
  - Résultats possibles
    - Insuffisant : pénalité de 2 points/20 sur la note du travail
    - Suffisant : la note du travail n'est pas modifiée
    - Très bien : bonus de 2 points/20 sur la note du travail

# Programmation orientée objet avancée

1. Prérequis
2. Contenu du cours
3. Evaluation
4. Type de passage des arguments

```
public class Personne {
    private String nom;
    public Personne (String nom) {
        this.nom = nom; }
    public void setNom (String nom) ...
    public String getNom( ) ... }
}
```

Passage par copie

```
public class Bidon {
    void modifierInt (int a) {
        a ++; }
    void modifierPersonne (Personne p) {
        p.setNom ("Jules"); }
}
```

Passage par référence

```
public class Principal {
    public static void main(String[] args) {
        Bidon bidon = new Bidon( );

        int x = 20;
        bidon.modifierInt(x);
        System.out.println("x = "+x);

        Personne pierre = new Personne("Pierre");
        bidon.modifierPersonne(pierre);
        System.out.println(pierre.getNom( ));
    }
}
```

(1)

(2)

Affichages

(1) ?

(2) ?

(1) = 20

(2) = Jules

# Programmation orientée objet avancée

1. Type de passage des arguments
2. Nombre d'arguments variable

# Nombre d'arguments variable

## Nombre variable d'arguments dans une méthode

### Via utilisation d'ellipsis ...

#### Conditions

- Un seul argument de type ellipsis
- Obligatoirement le dernier de tous les arguments
- Arguments en nombre variable: de type primitif ou référence
- Syntaxe: ***typeArgument...*** ***nomArgument***

Dans le code de la méthode:

Accès aux différents arguments via un tableau dont le nom est ***nomArgument***



# Nombre d'arguments variable

## Exemple de méthode avec un nombre variable d'arguments

Arguments: 0, 1 ou plusieurs objets de type Book

```
public class Library {  
    public int countPages (Book... books)  
    { int pagesTotal = 0;  
      for (int i = 0; i < books.length; i++)  
          { pagesTotal += books[i].getPagesCount( ); }  
      return pagesTotal;  
    }  
}
```

# Nombre d'arguments variable

## Exemple d'appel de méthode avec un nombre variable d'arguments

```
Book book1, book2, book3, book4;
```

```
Library library = new Library ();
```

```
int pagesTotal;
```

```
...
```

```
// Exemples d'appel de la méthode
```

```
pagesTotal = library.countPages( );
```

```
pagesTotal = library.countPages( book1 );
```

```
pagesTotal = library.countPages( book1, book2, book3, book4 );
```

# Nombre d'arguments variable

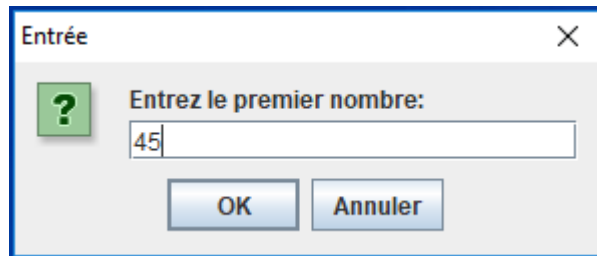
## Autre exemple

```
public static void main(String... args)
```

# Programmation orientée objet avancée

1. Type de passage des arguments
2. Nombre d'arguments variable
3. JOptionPane

# JOptionPane



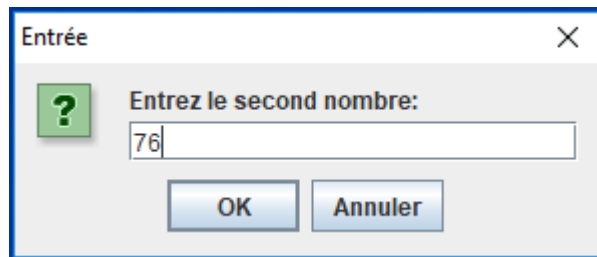
Entrée

?

Entrez le premier nombre:

45

OK Annuler



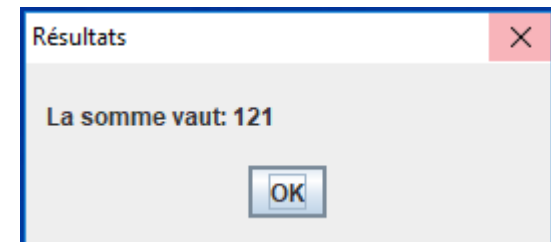
Entrée

?

Entrez le second nombre:

76

OK Annuler



Résultats

La somme vaut: 121

OK

```
import javax.swing.*;
```

```
public class Principal {
```

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

```
        int nombre1; int nombre2; int somme;
```

```
        String premier = JOptionPane.showInputDialog ("Entrez le premier nombre:");
```

```
        String second = JOptionPane.showInputDialog ("Entrez le second nombre: ");
```

```
        nombre1 = Integer.parseInt(premier); —————> Transforme String en entier
```

```
        nombre2 = Integer.parseInt(second);
```

```
        somme = nombre1 + nombre2;
```

```
        JOptionPane.showMessageDialog (null, "La somme vaut: " + somme,
                                         "Résultats", JOptionPane.PLAIN_MESSAGE);
```

*Contenu de la boîte de dialogue*

*icône*

*Titre de la boîte de dialogue*

```
        System.exit(0);
```

```
    }
```

```
}
```

# JOptionPane

JOptionPane.QUESTION\_MESSAGE



JOptionPane.ERROR\_MESSAGE



JOptionPane.INFORMATION\_MESSAGE



JOptionPane.WARNING\_MESSAGE

