

# INITIATION JAVA

Omar Farouk KOUGBADA

Ingénieur Informatique et Consultant Formateur en IT

# PROGRAMME

1. PRÉSENTATION
2. LE LANGAGE JAVA
3. INTERFACES GRAPHIQUES
4. COMMUNICATION BASE DE DONNÉES
5. INTRODUCTION AUX THREADS D'EXÉCUTION
6. INTRODUCTION À LA PROGRAMMATION TCP-IP
7. JAVA RMI
8. CONSTRUCTION D'APPLICATIONS DISTRIBUÉES CORBA

# PRÉSENTATION

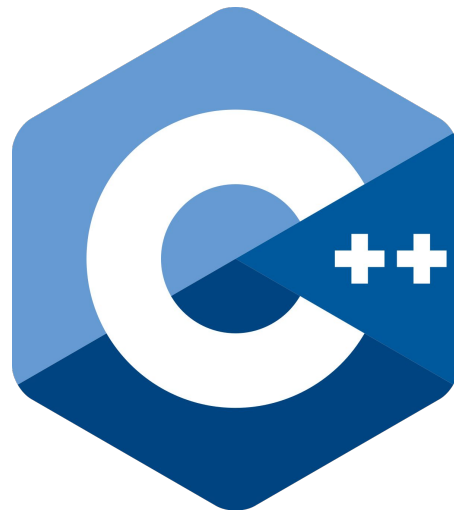
- Différents styles de programmation
- Avantages de la programmation objet
- Programmation modulaire
- L'abstraction et la spécialisation

# • Différents styles de programmation

- programmation orientée objet.
- Programmation fonctionnelle.
- Programmation impérative.
- Programmation récursive.
- Programmation dynamique.
- Programmation itérative.

# La programmation orientée objet (POO)

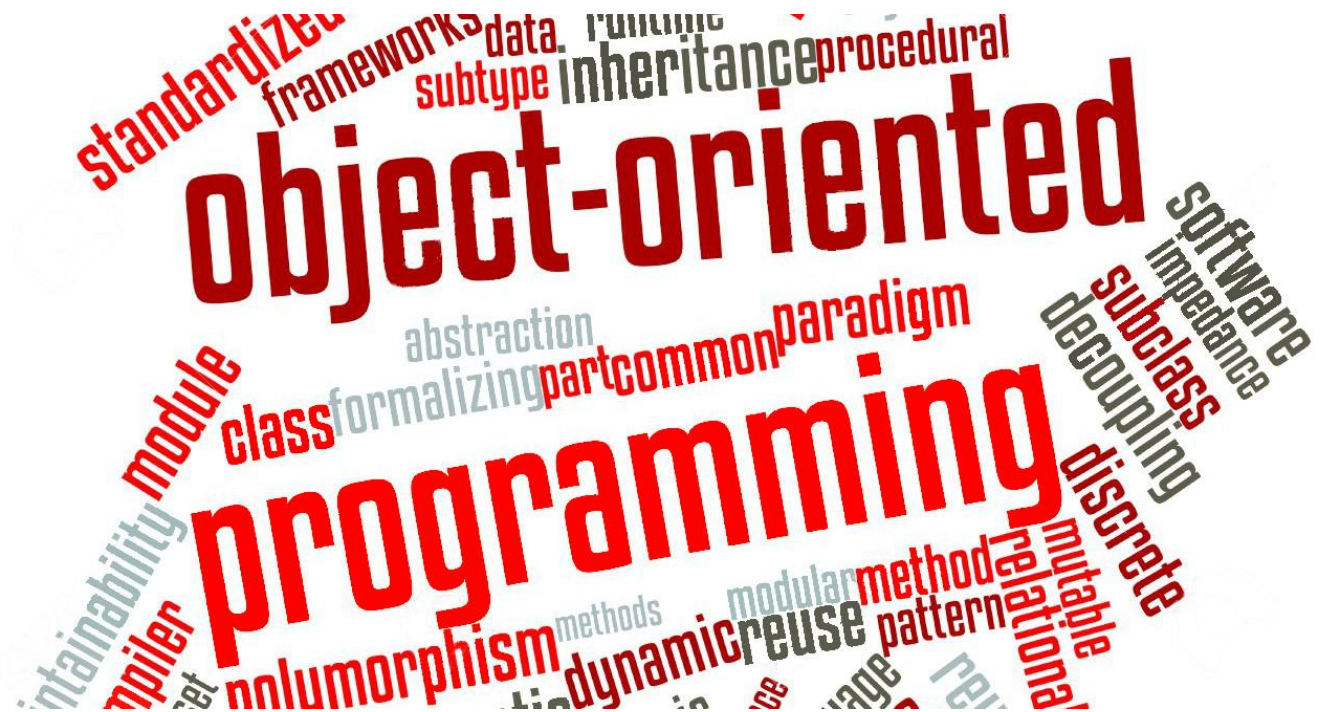
- POO est un paradigme informatique consistant à définir et à faire interagir des objets grâce à différentes technologies



Ruby

# PОО : concepts fondamentaux

- Objet
- Classe
- Héritage
- Encapsulation
- Polymorphisme

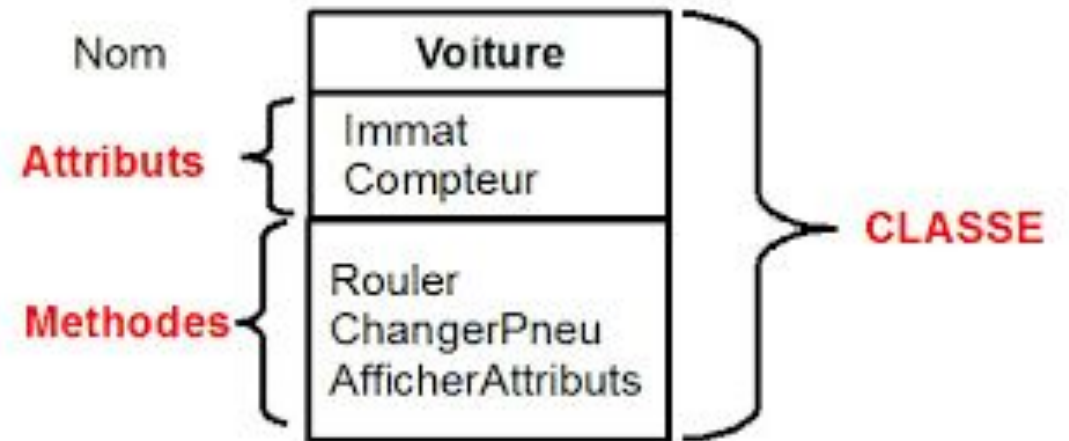


# POO : Objet

- Les objets sont omniprésents (humain, livre, voiture.) dans notre monde.
- En POO : **objet = identité + états + comportements**
- **L'identité** doit permettre d'identifier sans ambiguïté l'objet (adresse/référence ou nom).
- **Les états** sont stockés dans des propriétés (variables)
- **Les comportements** sont implémentés à l'aide de méthodes (procédures / fonctions)

# POO : classe

- Le monde réel regroupe des objets du même type.
- Il est pratique de concevoir une maquette d'un objet et de produire les objets à partir de cette maquette.
- En POO, une maquette se nomme une classe.
- Une classe est donc un modèle de la structure statique (variables d'instance) et du comportement dynamique (les méthodes) des objets associés à cette classe.





# POO:Héritage

- L'héritage, est l'un des mécanismes les plus puissants de la programmation orientée objet, permet de reprendre des membres d'une classe (appelée superclasse ou classe mère) dans une autre classe (appelée sous-classe, classe fille ou encore classe dérivée), qui en hérite.

```
public class Vehicule
{
    public int vitesse;
    public int nombre_de_places;
}

public class Automobile extends Vehicule
{
    public Automobile()
    {
        this.vitesse = 90;
        this.nombre_de_places = 5;
    }
}
```

# POO : Encapsulation

```
public class EncapTest {  
    private String name;  
    private String idNum;  
    private int age;  
  
    public int getAge() {  
        return age;  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    public String getIdNum() {  
        return idNum;  
    }  
  
    public void setAge( int newAge) {  
        age = newAge;  
    }  
  
    public void setName(String newName) {  
        name = newName;  
    }  
  
    public void setIdNum( String newId) {  
        idNum = newId;  
    }  
}
```

- Ce mécanisme permet donc de garantir l'intégrité des données contenues dans l'objet,
- Pour réaliser l'encapsulation en Java :
- Déclarez les variables d'une classe comme privées.
- Fournissez des méthodes setter et getter publiques pour modifier et afficher les valeurs des variables.

# POO: Polymorphisme

- Polymorphisme : signifie «plusieurs formes», et cela se produit lorsque nous avons de nombreuses classes liées les unes aux autres par héritage.
- L'héritage nous permet d'hériter des attributs et des méthodes d'une autre classe.
- Le polymorphisme utilise ces méthodes pour effectuer différentes tâches.
- Cela nous permet d'effectuer une seule action de différentes manières.
- Par exemple, pensez à une superclasse appelée Animal qui a une méthode appelée animalSound().
- Les sous-classes d'animaux peuvent être des chats, des chiens, des oiseaux et ils ont également leur propre implémentation d'un son animal (les miaulements de chat, les gazouillement des oiseaux ,etc.):

# Programmation fonctionnelle

- La programmation fonctionnelle est un paradigme de programmation de type déclaratif qui considère le calcul en tant qu'évaluation de fonctions mathématiques.
- Un langage fonctionnel est donc un langage de programmation dont la syntaxe et les caractéristiques encouragent la programmation fonctionnelle.
- Ce type de programmation permet de générer un code très précis

# Programmation impérative

- Ce paradigme décrit les opérations d'un programme comme des séquences d'instructions exécutées par l'ordinateur pour modifier l'état du programme.
- C'est le paradigme de programmation le plus ancien et il est retrouvé dans les jeux d'instructions des processeurs et dans les langages les plus utilisés aujourd'hui.



# Programmation réursive

```
Online Java Compiler - Online . x +
jdoodle.com/online-java-compiler/

5
6 public class fibonacci
7 {
8     public static int fib(int n)
9     {
10         //base case
11         if (n <= 1)
12             return n;
13         // recursive case;
14         else
15             return fib(n-1) + fib(n-2);
16     }
17
18     public static void main (String args[])
19     {
20         int n = 9; // n starts from 0
21         if (n < 0)
22             System.out.println("Fibonacci number is not defined");
23         else
24             System.out.println(fib(n));
25     }
26 }
```

# Programmation itérative

- La programmation itérative (ou impérative) repose sur une série d'instructions exécutées de façon séquentielle par l'ordinateur.
- Ces instructions viennent modifier étape par étape les valeurs des variables pour finalement aboutir au résultat souhaité.

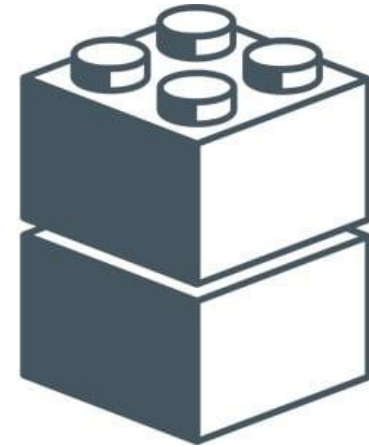
# Avantages de la programmation objet

- Modularité
- Abstraction
- Productivité
- Réutilisabilité
- Sécurité
- Encapsulation



# Avantages de la programmation objet

- Modularité : les objets forment des modules compacts regroupant des données et un ensemble d'opérations.
- Abstraction : Les entités objets de la POO sont proches de celles du monde réel. Les concepts utilisés sont donc proches des abstractions familières que nous exploitons.



# Avantages de la programmation objet

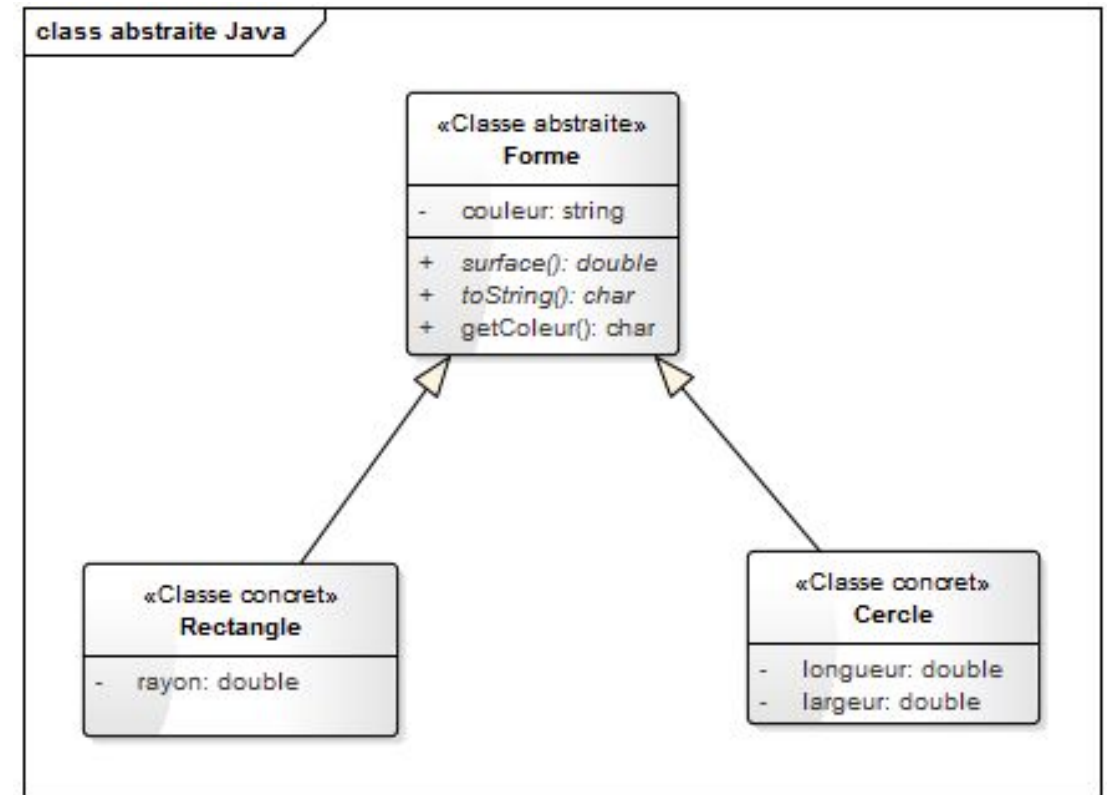
- Plus l'application est complexe et plus l'approche POO est intéressante en terme de productivité.
- Sûreté : L'encapsulation et le typage des classes offrent une certaine robustesse aux applications.

# La programmation modulaire

- En informatique, la programmation modulaire reprend l'idée de fabriquer un produit (le programme) à partir de composants (les modules).
- Réduction de la taille des programmes sources améliorant la lisibilité et réduisant les temps de compilation
- Structuration accrue de l'application par la conception de modules fonctionnellement indépendants.

# L'abstraction et la spécialisation

- L'abstraction est l'un des concepts clés dans les langages de programmation orientée objet (POO).
- Son objectif principal est de gérer la complexité en masquant les détails inutiles à l'utilisateur.
- Cela permet à l'utilisateur d'implémenter une logique plus complexe sans comprendre ni même penser à toute la complexité cachée.
- Les avantages de l'abstraction:
  - Réduire la complexité.
  - Évite la duplication de code et augmente la possibilité de réutilisation.
  - Aide à renforcer la sécurité d'une application ou d'un programme car seuls les détails importants sont fournis à l'utilisateur.

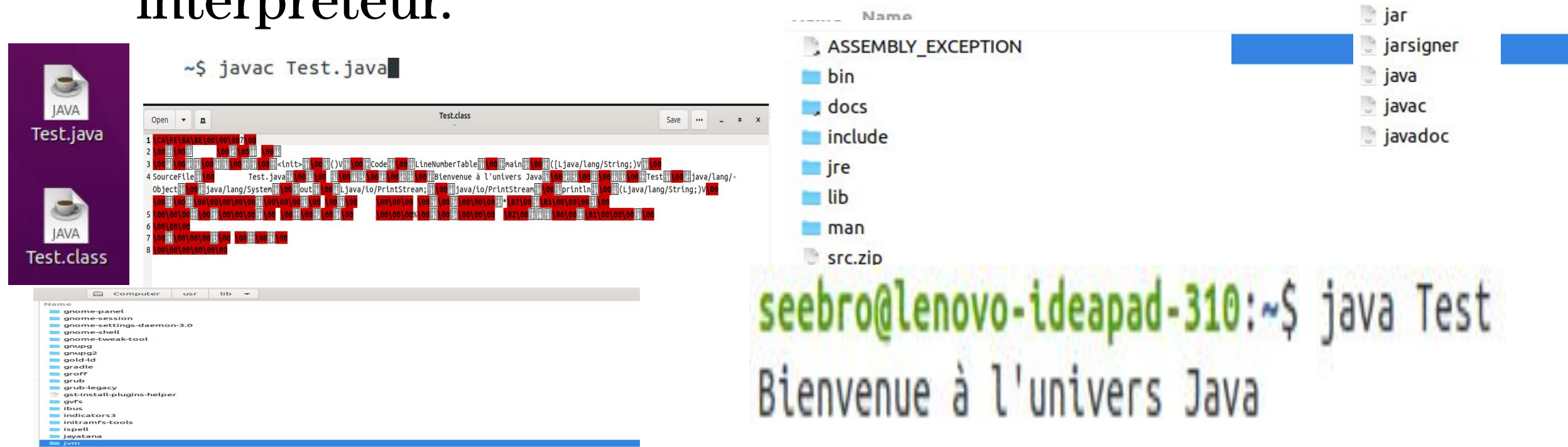


# LE LANGUAGE JAVA

- Compilation Btyecode, la machine virtuelle (JVM)
- Classes et interfaces
- Classes d'usage courant

# Compilation Btyecode, la machine virtuelle (JVM)

- Java est un langage interprété, ce qui signifie qu'un programme compilé n'est pas directement exécutable par le système d'exploitation mais il doit être interprété par un autre programme, qu'on appelle interpréteur.



The image shows a terminal window with the following content:

```
~$ javac Test.java
```

Below the terminal output, there is a file explorer window showing the contents of the current directory. The files listed are:

- Test.java
- Test.class
- bin
- docs
- include
- jre
- lib
- man
- src.zip
- jar
- jarsigner
- java
- javac
- javadoc

At the bottom of the terminal window, the command `java Test` is entered, and the output is:

```
Bienvenue à l'univers Java
```

# La machine virtuelle (JVM)

- Java Virtual Machine, ou JVM, charges, vérifie et exécute Java Bytecode. Il est connu comme l'interprète ou le noyau du langage de Java car il exécute un programme Java.
- JVM est spécifiquement responsable de la conversion de bytecode en code spécifique à la machine et est nécessaire pour JDK et JRE

# JRE

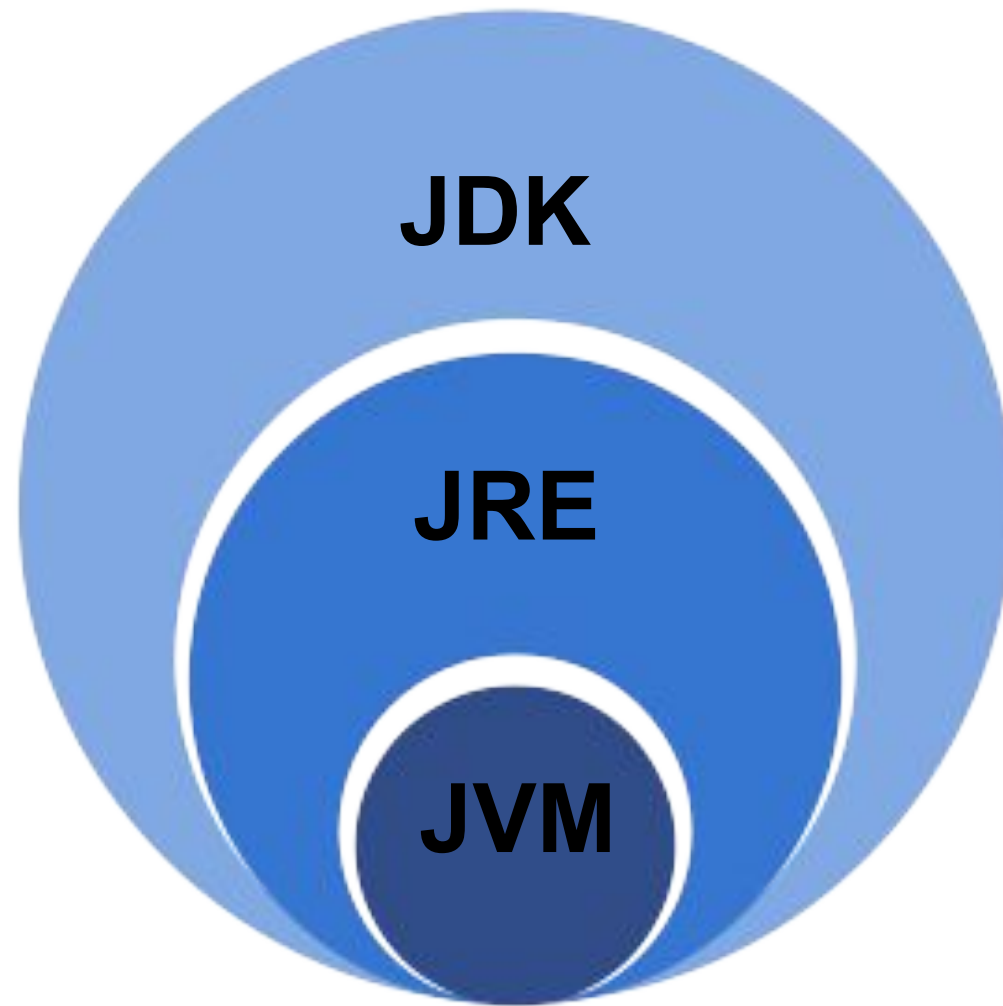
- Java Runtime Environment, ou JRE, est un ensemble d'outils logiciels responsables de l'exécution du programme Java ou de l'application sur votre système.
- Si un programmeur souhaite exécuter un programme Java à l'aide de la commande Java, ils doivent installer JRE.



# JDK

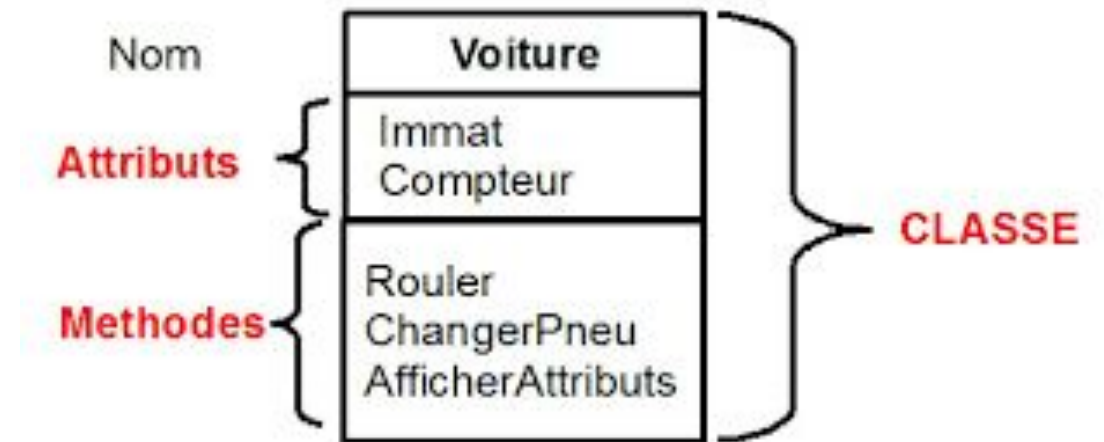
- Java Development Kit (JDK), est un kit de développement logiciel souvent appelé un superset de JRE.
- JDK, contient tous les outils nécessaires pour compiler, déboguer et exécuter un programme développé à l'aide de la plate-forme Java

# JVM vs JRE vs JDK



# Classes et interfaces

- Une interface décrit un ensemble de méthodes en fournissant uniquement leur signature.
- Une interface est un prototype de classe. Elle définit la signature des méthodes qui doivent être implémentées dans les classes construites à partir de ce prototype.
- Une classe est donc un modèle de la structure statique (variables d'instance) et du comportement dynamique (les méthodes) des objets associés à cette classe.



# Classes et interfaces

Compte.java

```
1 package abstraction;
2
3 public interface Compte {
4
5     void deposer(int montant);
6     int retirer (int montant);
7     int getBalance();
8 }
9
```

CompteBancaire.java

```
1 package abstraction;
2
3 public class CompteBancaire implements Compte{
4
5
6     private int balance;
7     @Override
8     public void deposer(int montant) {
9         this.balance += montant;
10    }
11
12    @Override
13    public int retirer(int montant) {
14
15        return this.balance -= montant;
16    }
17
18    @Override
19    public int getBalance() {
20        return this.balance;
21    }
22
23 }
24
```

# Classes d'usage courant

- La classe **java.util.Arrays** donne accès à des méthodes statiques permettant différentes opérations sur les tableaux en particulier les tris et les recherches d'éléments
- La classe **String** est une classe fondamentale du langage Java, puisque c'est elle qui permet de gérer les chaînes de caractères
- La classe **java.util.Date** : elle encapsule un point dans le temps
- La classe **java.util.Calendar** et **java.util.GregorianCalendar** : elle permet la manipulation d'une date
- La classe **java.util.TimeZone** et **java.util.SimpleTimeZone** : elle encapsule un fuseau horaire à partir du méridien de Greenwich (GMT) et les informations relatives aux décalages concernant les heures d'été et d'hiver
- La classe **java.text.DateFormat**, **java.text.SimpleDateFormat** : elle permet de convertir une date en chaîne de caractères et vice versa
- La classe **java.text.DateFormatSymbols** : elle permet de traduire les différents éléments d'une date (jour, mois, ...)

# INTERFACES GRAPHIQUES

- Présentation de Swing
- Les composants graphiques Swing

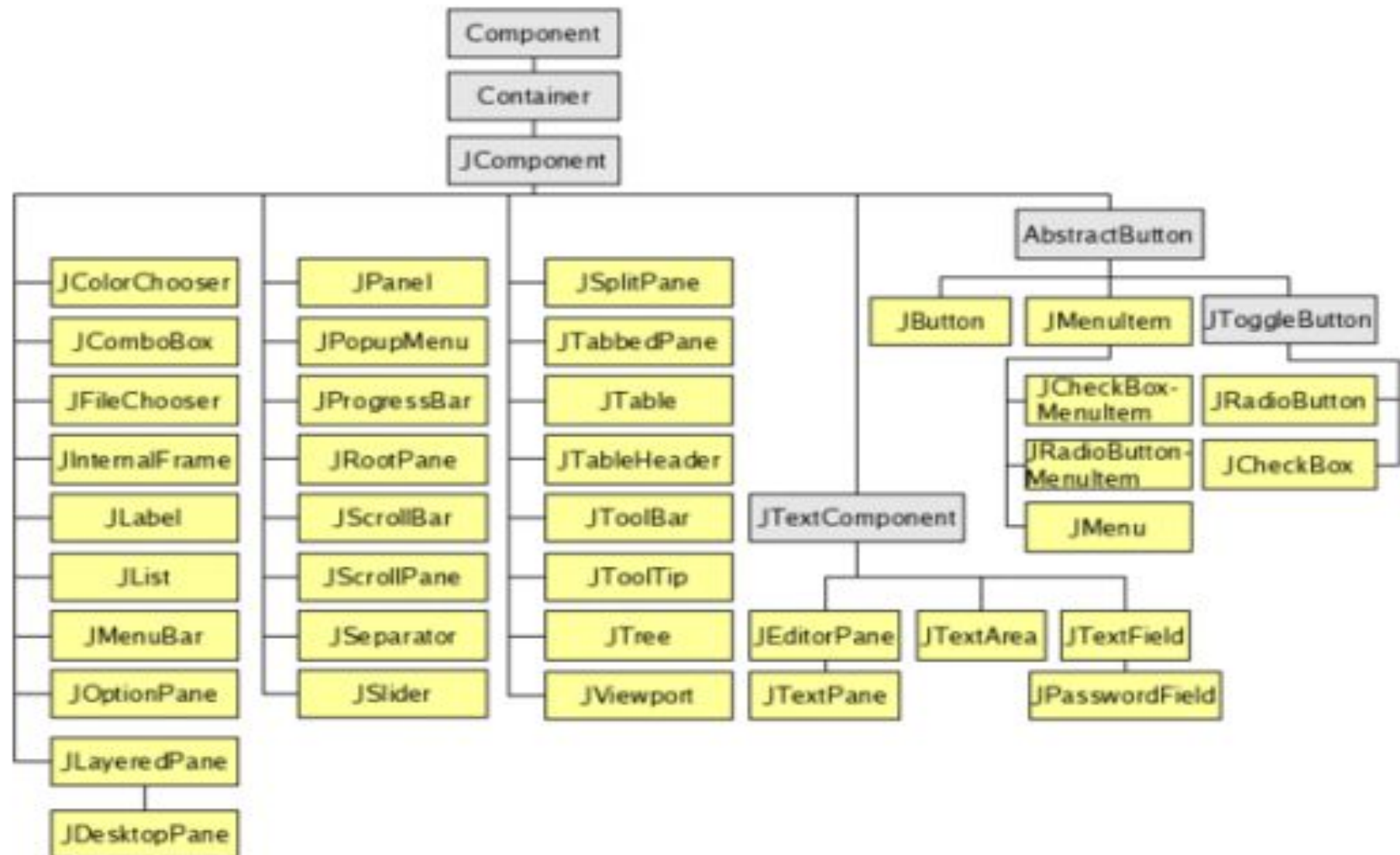


# AWT & SWING

- La librairie AWT (Abstract Window Toolkit) a été introduite dès les premières versions de Java
- Les classes d'AWT permettent de développer des interfaces graphiques indépendantes du système d'exploitation sur lesquels elles vont fonctionner.
- Swing fait partie de la bibliothèque Java Foundation Classes (JFC).
- C'est une API dont le but est similaire à celui de l'API AWT seulement les modes de leur fonctionnement et leur utilisation sont totalement différents



# Hiérarchie des classes de la librairie Swing





# Les composants Swing

- C'est un beans
- Ses bords, sa taille et sa couleur peuvent être changés
- Peut être activé ou désactivé
- Peut être visible ou non
- Peut être personnalisé en apparence et en comportement

# Les composants Swing

- Les composants de présentation
- Les composants de choix
- Les composants de saisie

# • Les composants de présentation : Le label

- Un label ou une étiquette est l'élément le plus fondamental dans la bibliothèque Swing.
- Il sert à afficher un texte et/ou une icône (image) non modifiable par l'utilisateur.

- JLabel()
- JLabel(String)
- JLabel(Icon)



- Les composants de présentation : Le ToolTip

- C'est une bulle d'aide qui s'affiche momentanément pour décrire un composant

```
label.setToolTipText("Ceci est un label");
```



- Les composants de présentation : Le séparateur

- C'est un composant simple qui divise visuellement l'interface en sections en utilisant des lignes horizontales ou verticales
- JSeparator



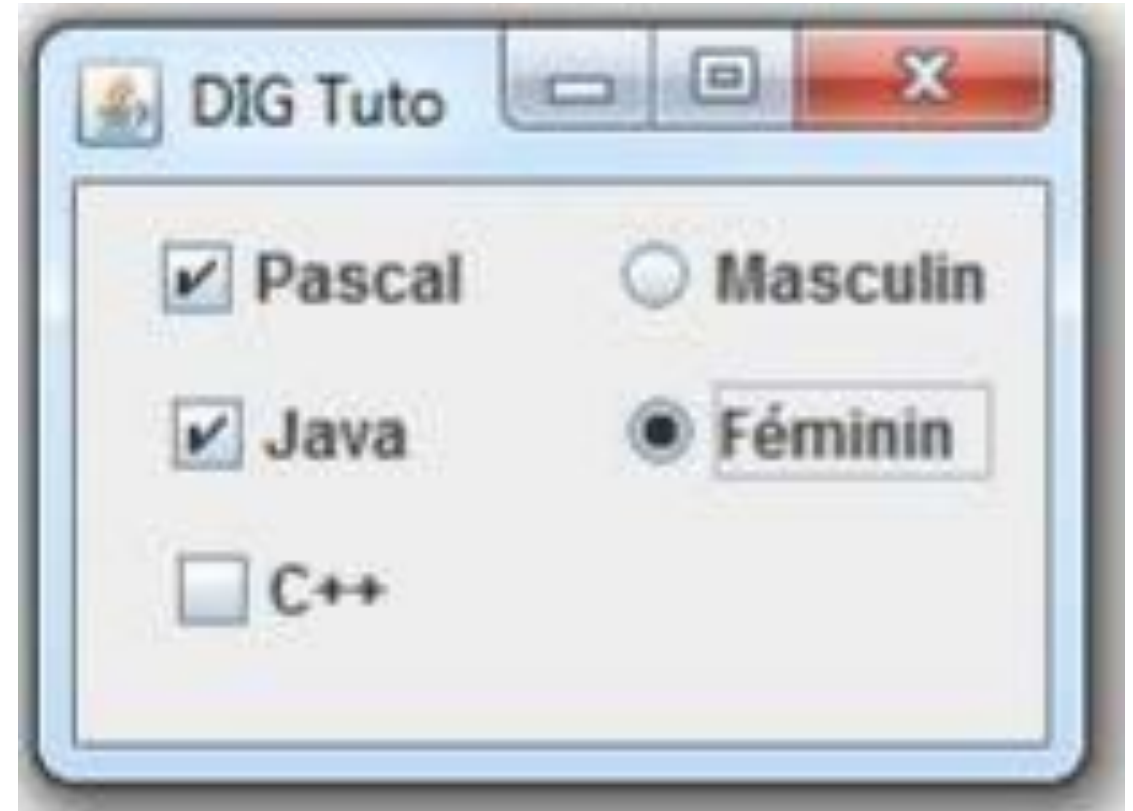
# Les composants de choix :Le Bouton

- Un bouton est le composant d'action de base dans Swing.
- C'est tout simplement un élément graphique sur lequel l'utilisateur peut cliquer pour déclencher une action
- JButton
- getText
- get/setHorizontalAlignment
- get/setVerticalAlignment
- actif / inactif
- sélectionnés / non sélectionné



# Les cases à côcher et les boutons radio

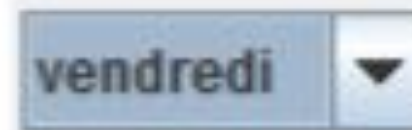
- Ces deux composants offrent à l'utilisateur des options (ou des réponses à une question) généralement sous forme de choix multiples. Ceux sont des objets graphiques à côcher/sélectionner
- JcheckBox
- JradioButton
- ButtonGroup



# Le combo box

- C'est un composant sous la forme d'une liste déroulante préétablie non modifiable

```
String [] jours= { "lundi", "mardi",  
                  "mercredi", "jeudi", "vendredi", "samedi", "dimanche"};  
JComboBox cb= new JComboBox(jours);  
cb.setBounds(50,50,90,20);  
f.add(cb);
```

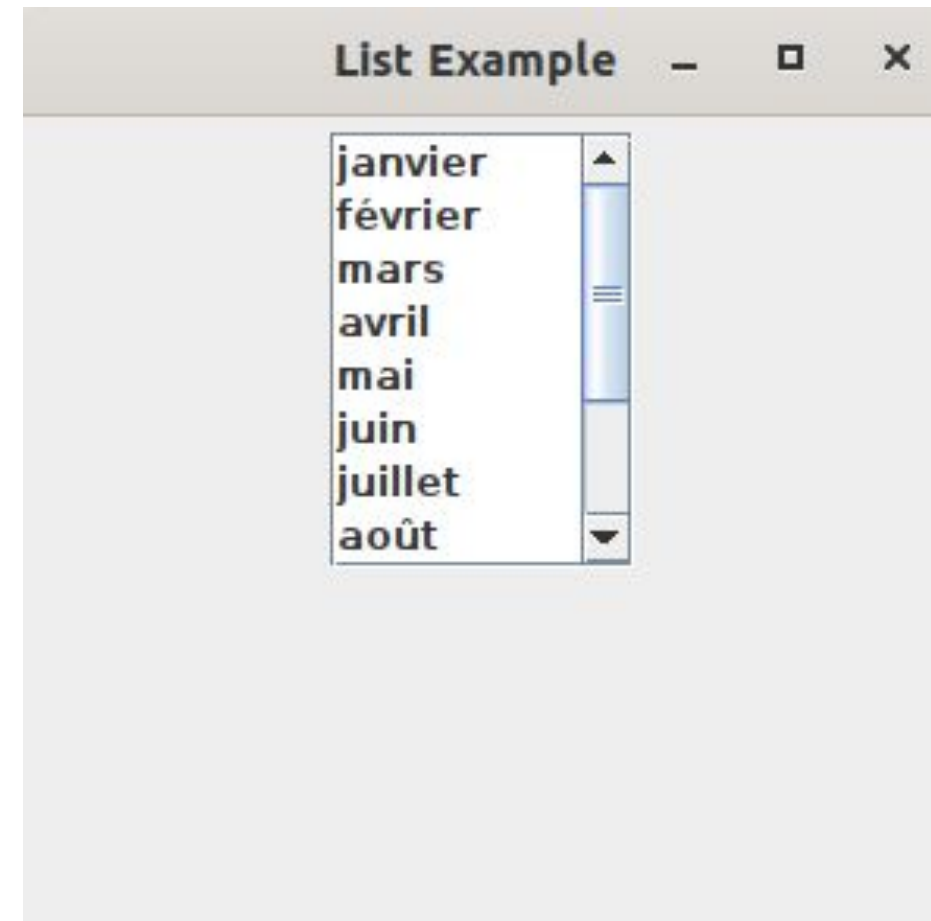




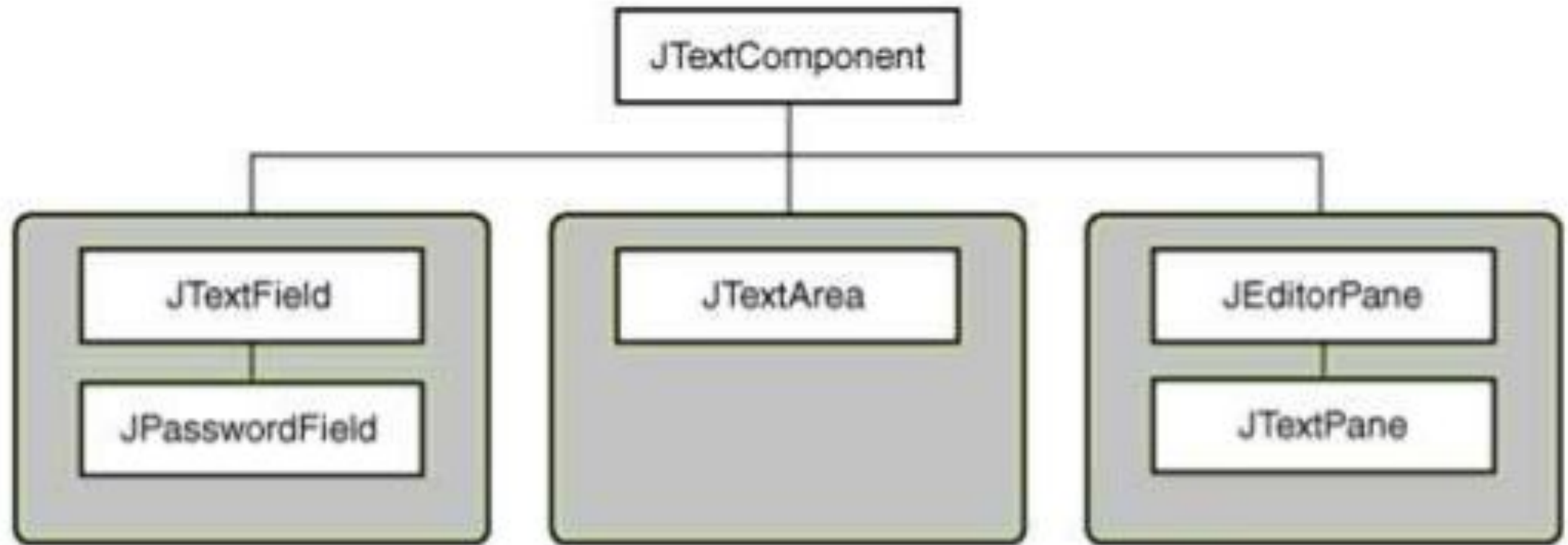
# La liste de valeurs

```
public class JListExample extends JFrame{
    JList list;
    String [] mois={"janvier", "février", "mars",
        "avril", "mai", "juin", "juillet", "août",
        "septembre", "octobre", "novembre", "décembre"};

    Container contentpane;
    public JListExample(){
        super("List Example ");
        contentpane = getContentPane();
        contentpane.setLayout(new FlowLayout());
        list = new JList(mois);
        list.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
        contentpane.add(new JScrollPane(list));
        list.setSelectedIndex(0);
        setSize(300, 300);
        setVisible(true);
    }
    public static void main(String[] args) {
        JListExample test = new JListExample();
        test.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```



# Les composants de saisie

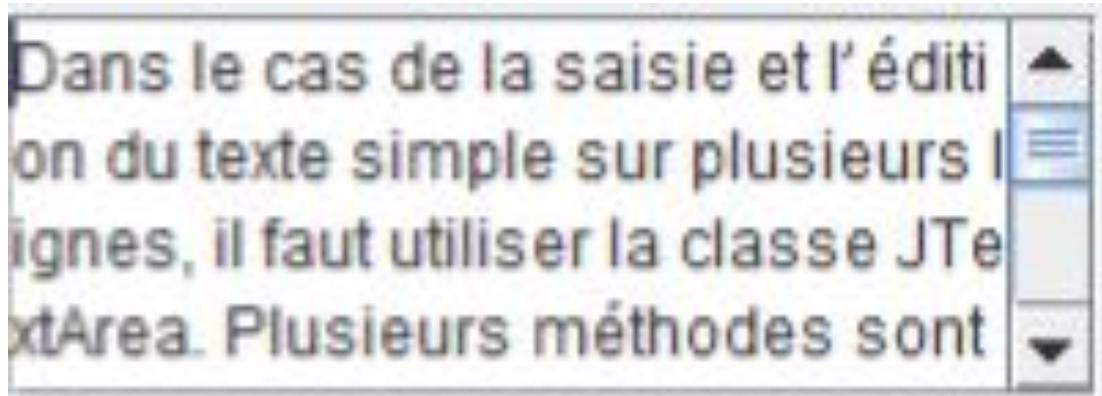


# Un champ de texte

- Pour afficher, saisir et éditer du texte simple sur une seule ligne, il suffit d'utiliser la classe `JTextField` et la sous classe `JPasswordField`
- La classe `JTextField` dispose également de ses propres méthodes comme par exemple `setFont()` pour spécifier la police du texte et `setColumns()` pour définir le nombre de caractères dans le champ de texte
- Il est toutefois possible de lire ou modifier le texte à l'aide des méthodes `getText()` et `setText()` de la super classe `JTextComponent`.



# Zone de texte



`.JTextArea`

`.setText()`

`.append()`

`.insert()`

# COMMUNICATION BASE DE DONNÉES

- Gestion des bases de données avec l'API JDBC



- Gestion des bases de données avec l'API JDBC

- JDBC (Java DataBase Connectivity) est l'API(interface d'application) standard pour interagir avec les bases données relationnelles en Java.
- Cette API peut être utilisée dans une application Web

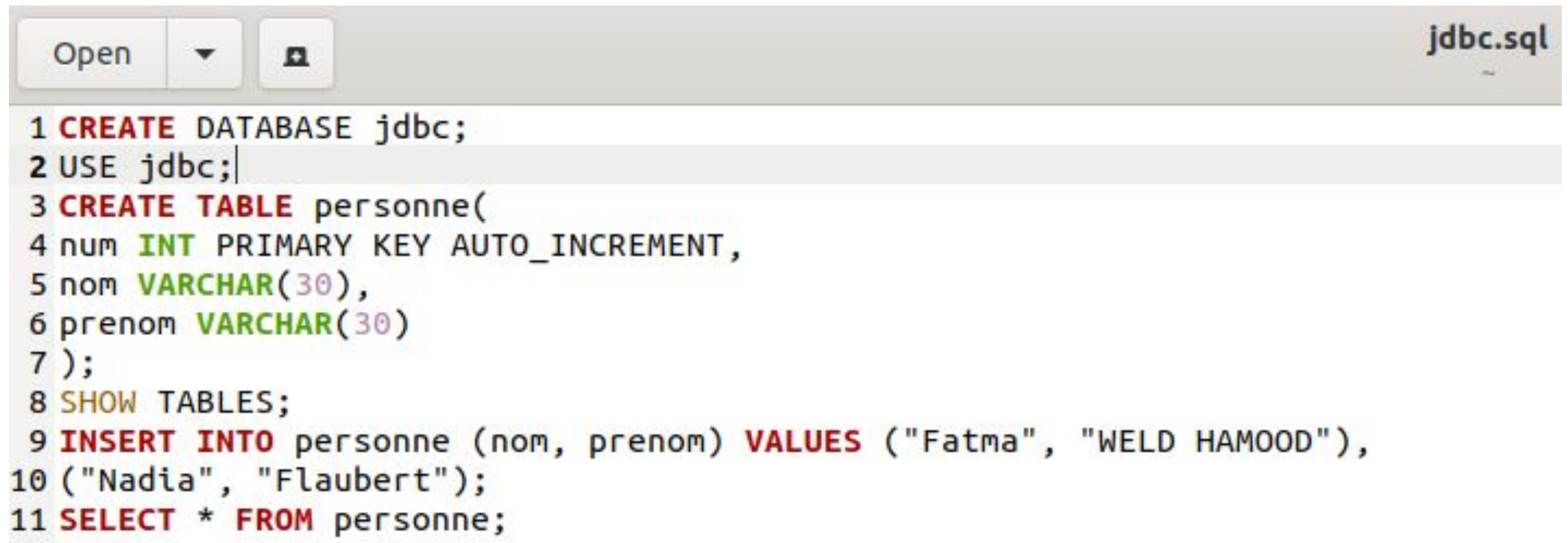


- Gestion des bases de données avec l'API JDBC

- Aller sur <https://dev.mysql.com/downloads/workbench/>
- Faire clic droit sur le projet dans Package Explorer et aller dans Properties
- Dans Java Build Path, aller dans l'onglet Libraries
- Cliquer sur Add External JARs
- Indiquer le chemin du .jar qui se trouve dans l'archive décompressée
- Appliquer



- Gestion des bases de données avec l'API JDBC



```
1 CREATE DATABASE jdbc;
2 USE jdbc;
3 CREATE TABLE personne(
4 num INT PRIMARY KEY AUTO_INCREMENT,
5 nom VARCHAR(30),
6 prenom VARCHAR(30)
7 );
8 SHOW TABLES;
9 INSERT INTO personne (nom, prenom) VALUES ("Fatma", "WELD HAMOOD"),
10 ("Nadia", "Flaubert");
11 SELECT * FROM personne;
```



- Charger le driver JDBC

- Charger le driver JDBC (pour MySQL dans notre cas)
- Etablir la connexion avec la base de données
- Créer et exécuter des requêtes SQL

# Chargement du driver 8

```
try {  
    Class.forName("com.mysql.cj.jdbc.Driver");  
    connexion = DriverManager.getConnection( url, utilisateur, motDePasse );  
} catch ( Exception e ) {  
    e.printStackTrace();  
}  
}
```

# Se connecter à la base de données

```
private static String url = "jdbc:mysql://localhost:3306/jdbc?useSSL=false&serverTimezone=UTC";  
private static String utilisateur = "root";  
private static String motDePasse = "";  
private static Connection connexion = null;
```

# JDBC

- Création de la requête (statement)
- `Statement statement = connexion.createStatement();`
- Préparation de la requête
- `String request = "SELECT * FROM Personne;";`
- Exécution de la requête
- `ResultSet result = statement.executeQuery(request);`

# Recupération des données

```
if (r.next())  
    personne = new Personne(r.getInt("num"), r.getString("nom"), r.getString("prenom));
```

# Pour faire une insertion

```
try {  
    PreparedStatement ps = c.prepareStatement("insert into  personne (nom,prenom) values (?,?); ",  
        PreparedStatement.RETURN_GENERATED_KEYS);  
    ps.setString(1, personne.getNom());  
    ps.setString(2, personne.getPrenom());  
    ps.executeUpdate();  
}
```

# Pratiquons ensemble !



# INTRODUCTION AUX THREADS D'EXÉCUTION

- Un thread est une unité d'exécution faisant partie d'un programme.
- Cette unité fonctionne de façon autonome et parallèlement à d'autres threads.
- Le principal avantage des threads est de pouvoir répartir différents traitements d'un même programme en plusieurs unités distinctes pour permettre leurs exécutions "simultanées".



# L'interface Runnable

- Cette interface doit être implémentée par toute classe qui contiendra des traitements à exécuter dans un thread.
- Cette interface ne définit qu'une seule méthode : `void run()`.

```
RunnableImpl.java
1 package Threads;
2
3 public class RunnableImpl implements Runnable{
4
5     @Override
6     public void run() {
7         int i = 0;
8         for (i = 0; i > 10; i++) {
9             System.out.println(i);
10        }
11
12    }
13
14 }
```

# Classes et groupes de threads

- La classe **java.lang.Thread** et l'interface **java.lang.Runnable** sont les bases pour le développement des threads en java