# Licence Informatique 3e année

# POO & Java

# TP n°6 – Application Graphique de Compression/Décompression avec JavaFX et Maven

---

# Contexte Étendu

Après avoir développé une application console pour la compression et décompression de fichiers textes, ce TP propose de concevoir une **interface graphique utilisateur (GUI)** avec **JavaFX**, tout en intégrant **Maven** comme gestionnaire de projet.

L'objectif est de permettre aux utilisateurs de :

- **Charger un ou plusieurs fichiers** par glisser-déposer ou via un explorateur.
- **Choisir le type d'opération** : compression ou décompression.
- **Afficher une barre de progression** pendant les traitements.
- **Vérifier la conformité des fichiers** avant de lancer les opérations.

Ce projet combinera les notions précédentes (POO, multithreading, gestion des exceptions) avec la gestion des interfaces graphiques et des contrôles interactifs.

---

# Énoncé du TP

## *1. Intégration de JavaFX avec Maven*

1. **Configuration de Maven** :
    - Ajoutez les dépendances nécessaires pour JavaFX dans votre `pom.xml`.
    - Configurez le plugin Maven pour exécuter l'application :

```
<dependencies>
    <dependency>
        <groupId>org.openjfx</groupId>
        <artifactId>javafx-controls</artifactId>
        <version>21.0.0</version>
    </dependency>
    <dependency>
        <groupId>org.openjfx</groupId>
        <artifactId>javafx-fxml</artifactId>
        <version>21.0.0</version>
    </dependency>
</dependencies>

<build>
    <plugins>
        <plugin>
            <groupId>org.openjfx</groupId>
            <artifactId>javafx-maven-plugin</artifactId>
            <version>0.0.14</version>
            <configuration>
                <mainClass>com.example.filecompression.MainApp</mainClass>
            </configuration>
        </plugin>
    </plugins>
</build>
```

2. **Création de la structure Maven** :

   Organisez vos fichiers conformément à la structure Maven standard :

```
src/
├── main/
│   ├── java/         # Code source Java
│   ├── resources/    # Fichiers FXML, CSS
└── test/
    ├── java/         # Tests unitaires
```

---

## *2. Développement de l'Interface Graphique*

1. **Conception de l’interface** :

   Créez une interface avec :

   - Une **zone de drag-and-drop** pour ajouter des fichiers.
   - Une **liste des fichiers** en attente de traitement.
   - Des **boutons** pour lancer les opérations (`Compresser`, `Décompresser`).
   - Une **barre de progression**.

Exemple d'organisation :

```
+-----------------------------------------+
|          [Zone de Drag-and-Drop]        |
|  [ Liste des fichiers chargés ]         |
+-----------------------------------------+
|  [Compresser] [Décompresser]            |
|  Barre de progression : [========   ]   |
|  Logs : "Compression réussie."          |
+-----------------------------------------+
```

2. **Exemple de Drag-and-Drop** :

   Implémentez la gestion du glisser-déposer avec JavaFX :

```
zone.setOnDragOver(event -> {
    if (event.getGestureSource() != zone && event.getDragboard().hasFiles()) {
        event.acceptTransferModes(TransferMode.COPY);
    }
    event.consume();
});

zone.setOnDragDropped(event -> {
    Dragboard db = event.getDragboard();
    if (db.hasFiles()) {
        db.getFiles().forEach(file -> {
            fileList.add(file);
        });
    }
    event.setDropCompleted(true);
    event.consume();
});
```

---

## *3. Compression/Décompression en Arrière-Plan*

1. **Utilisation de `Task`** :

   Gérer les opérations longues en arrière-plan pour ne pas bloquer l'interface :

```
Task<Void> compressionTask = new Task<>() {
    @Override
    protected Void call() throws Exception {
        for (int i = 0; i < files.size(); i++) {
            File file = files.get(i);
            compressFile(file); // Méthode de compression
            updateProgress(i + 1, files.size());
        }
        return null;
    }
};
progressBar.progressProperty().bind(compressionTask.progressProperty());
new Thread(compressionTask).start();
```

2. **Feedback à l'utilisateur** :

   Affichez des alertes ou des messages dans l'interface :

```
compressionTask.setOnSucceeded(event -> showAlert("Succès", "Compression terminée !"));
compressionTask.setOnFailed(event -> showAlert("Erreur", "Une erreur est survenue."));
```

---

### *4. Vérification des Fichiers*

1. **Validation des fichiers** :

   Avant de les traiter, vérifiez leurs extensions et leur taille :

```
if (!file.getName().endsWith(".txt") && !file.getName().endsWith(".png")) {
    showAlert("Format non supporté", "Le fichier " + file.getName() + " n'est pas valide.");
}
```

---

# Ressources

- JavaFX Documentation (https://openjfx.io/)
- JavaFX Drag-and-Drop Tutorial (https://www.geeksforgeeks.org/javafx-drag-and-drop/)
- Task API in JavaFX (https://docs.oracle.com/javase/8/javafx/api/javafx/concurrent/Task.html)
- Maven and JavaFX Guide (https://openjfx.io/openjfx-docs/#maven)

---