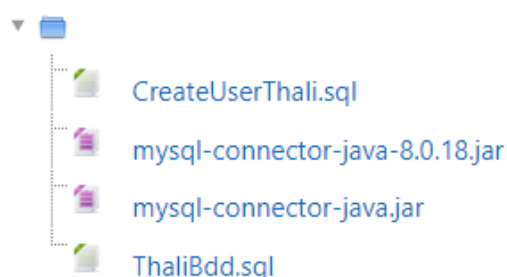


Etape 1 – Prise en main de l'existant

1 – Créer la base de données « Thali » sur le serveur MySQL local à l'aide du script « ThaliBdd.sql » fourni

Dans un premier temps, il faut récupérer les ressources du projet dans Moodle

Compléments pour les sources : SQL, connecteurs ...



Télécharger le dossier


Ensuite, il faut lancer les serveurs locaux Apache et MySQL (dans notre cas, à l'aide de XAMPP)



Une fois fait, il faut aller dans un navigateur et aller sur PhpMyAdmin (localhost/phpmyadmin), puis créer une base de donnée nommé Thali en utf9mb4_general_ci



Ensuite il faut importer « thalibdd.sql » qui va permettre de créer les tables « étape » et « excursion »

 Aucune table n'a été trouvée dans cette base de données.

Fichier à importer :


Le fichier peut être compressé (gzip, bzip2) ou non.
Le nom du fichier compressé doit se terminer par .[format].[compression]. Exemple : .sql.zip

Parcourir les fichiers : (Taille maximale : 40Mo)

Choisir un fichier


Il est également possible de glisser-déposer un fichier sur n'importe quelle page.

Jeu de caractères du fichier :

 L'importation a réussi, 21 requêtes exécutées. (ThaliBdd.sql)

2 – Créer l'utilisateur MySQL « thali_util » (mdp = « secret ») à l'aide du script « CreateUserThali.sql » fourni

Maintenant, il faut importer « CreateUserThali.sql » qui va permettre de créer un utilisateur ayant toutes les permissions. Cela va nous servir à pouvoir se connecter à notre base de donnée depuis ce compte et récupérer les informations comprises dans les tables.

 L'importation a réussi, 2 requêtes exécutées. (CreateUserThali.sql)

 Utilisateurs ayant accès à « thali »

	Nom d'utilisateur	Nom d'hôte	Type	Privilèges	« Grant »	Action
<input type="checkbox"/>	root	127.0.0.1	global	ALL PRIVILEGES	Oui	 Éditer les privilèges  Exporter
<input type="checkbox"/>	root	:::1	global	ALL PRIVILEGES	Oui	 Éditer les privilèges  Exporter
<input type="checkbox"/>	root	localhost	global	ALL PRIVILEGES	Oui	 Éditer les privilèges  Exporter
<input type="checkbox"/>	thali_util	localhost	spécifique à cette base de données	ALL PRIVILEGES	Non	 Éditer les privilèges  Exporter

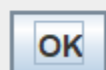
 ☐ Tout cocher Avec la sélection :  Exporter

3 – Résolution des éventuelles erreurs

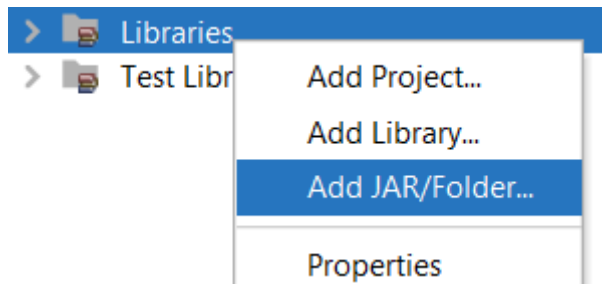
Lanceur



Echec de lecture initiale des données dans la BDD

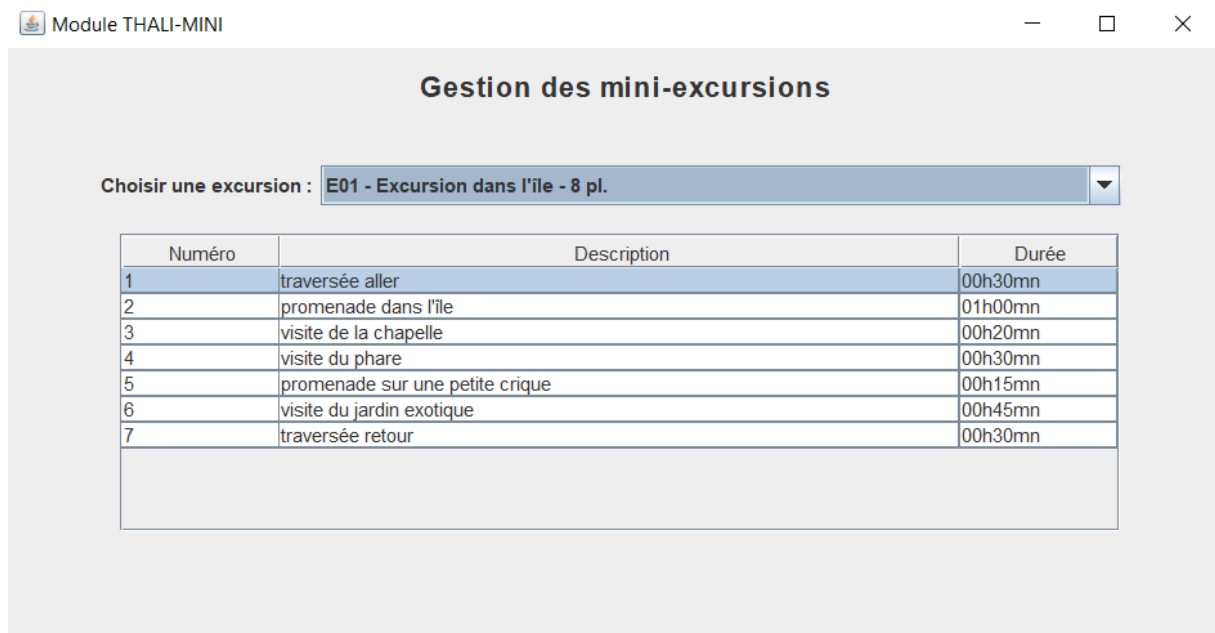


Pour éviter cette erreur, il faut importer le fichier « mysql-connector-java.jar » qui va permettre de se connecter à la base de donnée



4 – Vérification

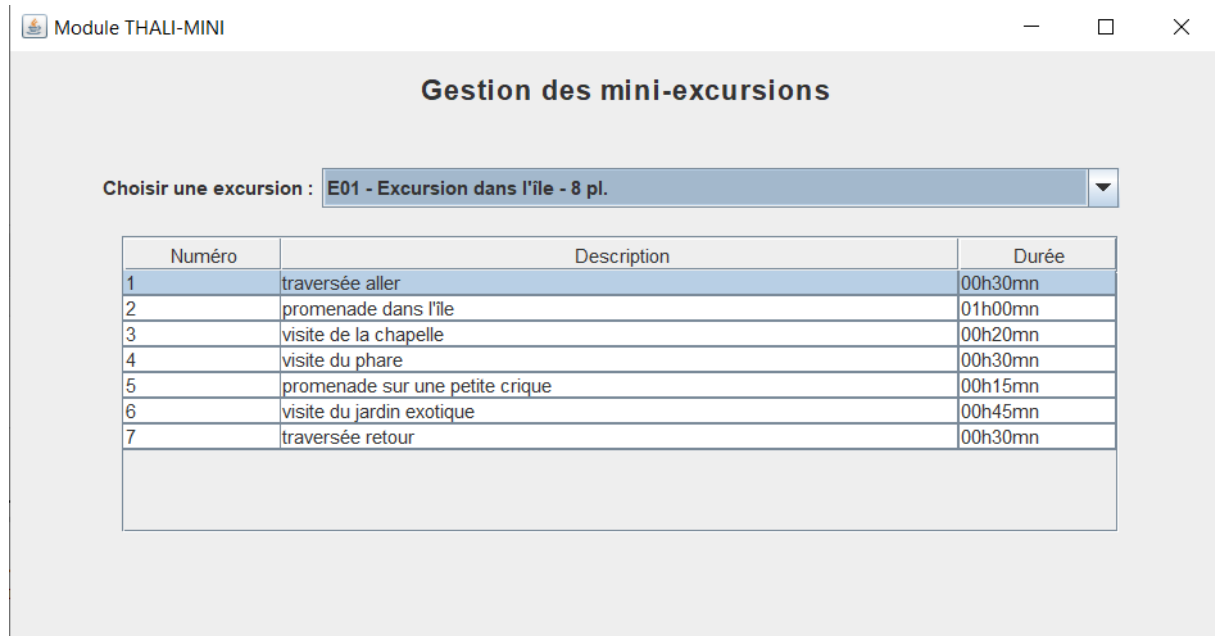
En ayant suivi les étapes, nous obtenons cette interface graphique



Etape 2 – Découverte des classes DAO

1 – Exécuter la classe TestDaoMiniExcursion

En exécutant la classe TestDaoMiniExcursion, nous obtenons cette interface graphique

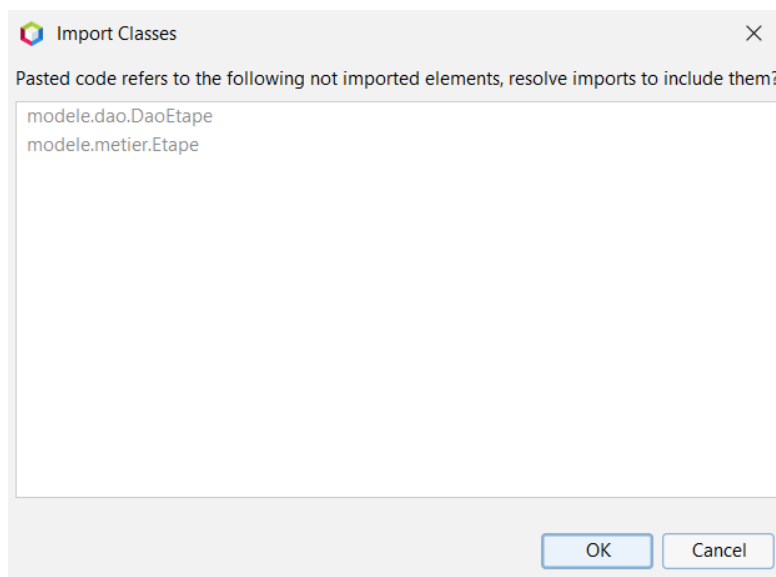


2 – Ajouter la méthode getOneById

Pour ajouter la méthode GetOneById dans la classe DaoMiniExcursion, nous utiliserons la méthode GetOneById présente dans la classe DaoEtape

```
public static Etape getOneById(String codeExcursion, int numEtape) throws SQLException {
    Etape etapeTrouvee = null;
    Connection cnx = ConnexionBDD.getConnexion();
    PreparedStatement pstmt = cnx.prepareStatement("SELECT * FROM Etape WHERE CodeExcursion = ? AND NumEtape = ?");
    pstmt.setString(parameterIndex: 1, x: codeExcursion);
    pstmt.setInt(parameterIndex: 2, x: numEtape);
    ResultSet rs = pstmt.executeQuery();
    if (rs.next()) {
        etapeTrouvee = new Etape(
            num: rs.getInt(columnLabel: "NumEtape"),
            description: rs.getString(columnLabel: "Description"),
            dureePrevue: rs.getInt(columnLabel: "DureePrevue")
        );
    }
    return etapeTrouvee;
}
```

En copiant le code de DaoEtape à DaoMiniExcursion nous avons ce message



Pour pouvoir utiliser le code nous devons inclure les classes proposés par NetBeans. Après avoir fait cela, nous avons des erreurs

```
43 codeExcursion = "E02";
44 numEtape = 2;
45 Etape etp = DaoEtape.getOneById(codeExcursion, numEtape);
46 if (etp != null) {
47     System.out.println("Etape d'id \" + codeExcursion + "\", " + numEtape + ") trouvée : \n" + etp.toString())
48 } else {
49     System.out.println("Etape d'id \" + codeExcursion + "\", " + numEtape + ") non trouvée : \n");
50 }
51
52 } catch (SQLException ex) {
53     System.out.println("TestDaoEtape - échec getOneById : " + ex.getMessage());
54     Logger.getLogger(name: TestDaoEtape.class.getName()).log(level: Level.SEVERE, msg: "Echec test 1", thrown: ex);
55 }
```

Pour régler ces problèmes il faut ajouter les paramètres codeExcursion et numEtape

```
* @param codeExcursion
* @param numEtape
```

Enfin, pour que le code fonctionne, il faut créer les variables codeExcursion et numEtape dans DaoMiniExcursion

```
String codeExcursion;
int numEtape;
```

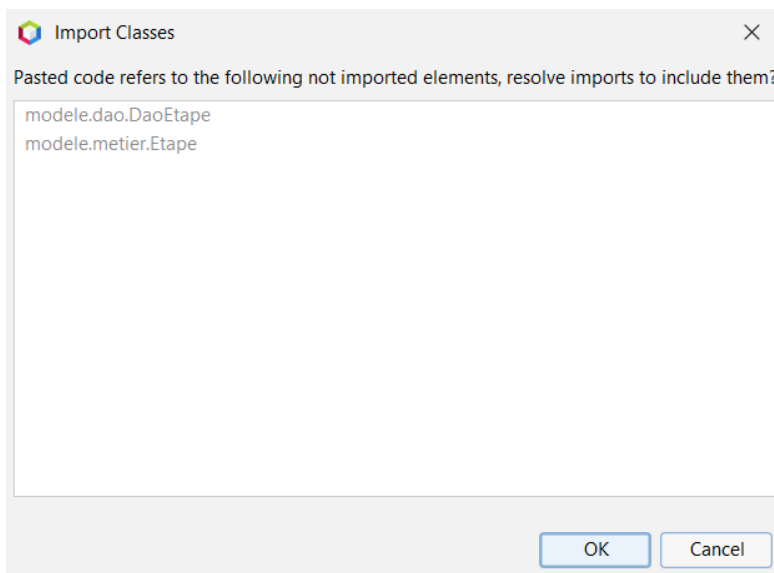
3 – Ajouter le test nécessaire dans TestDaoMiniExcursion

Maintenant, il faut copier le test `getOneById` de la classe `TestDaoEtape` à la classe `TestDaoMiniExcursion`

```
// Test 1 getOneById
System.out.println(x: "\n Test 1 : TestDaoEtape.getOneById");
try {
    codeExcursion = "E02";
    numEtape = 2;
    Etape etp = DaoEtape.getOneById(codeExcursion, numEtape);
    if (etp != null) {
        System.out.println("Etape d'id (\\"" + codeExcursion + "\", " + numEtape + ") trouvée : \n" + etp.toString())
    } else {
        System.out.println("Etape d'id (\\"" + codeExcursion + "\", " + numEtape + ") non trouvée : \n");
    }
}

} catch (SQLException ex) {
    System.out.println("TestDaoEtape - échec getOneById : " + ex.getMessage());
    Logger.getLogger(name: TestDaoEtape.class.getName()).log(level: Level.SEVERE, msg: "Echec test 1", thrown: ex);
}
```

En copiant le code de `TestDaoEtape` à `TestDaoMiniExcursion` nous avons ce message



Pour pouvoir utiliser le code nous devons inclure les classes proposés par NetBeans. Après avoir fait cela, nous avons des erreurs

```
43      codeExcursion = "E02";
44      numEtape = 2;
45      Etape etp = DaoEtape.getOneById(codeExcursion, numEtape);
46      if (etp != null) {
47          System.out.println("Etape d'id (\\"" + codeExcursion + "\", " + numEtape + ") trouvée : \n" + etp.toString())
48      } else {
49          System.out.println("Etape d'id (\\"" + codeExcursion + "\", " + numEtape + ") non trouvée : \n");
50      }
51
52  } catch (SQLException ex) {
53      System.out.println("TestDaoEtape - échec getOneById : " + ex.getMessage());
54      Logger.getLogger(name: TestDaoEtape.class.getName()).log(level: Level.SEVERE, msg: "Echec test 1", thrown: ex);
55  }
```

Pour régler ces problèmes il faut ajouter les paramètres `codeExcursion` et `numEtape`

```
* @param codeExcursion
* @param numEtape
```

Enfin, pour que le code fonctionne, il faut créer les variables codeExcursion et numEtape dans TestDaoMiniExcursion

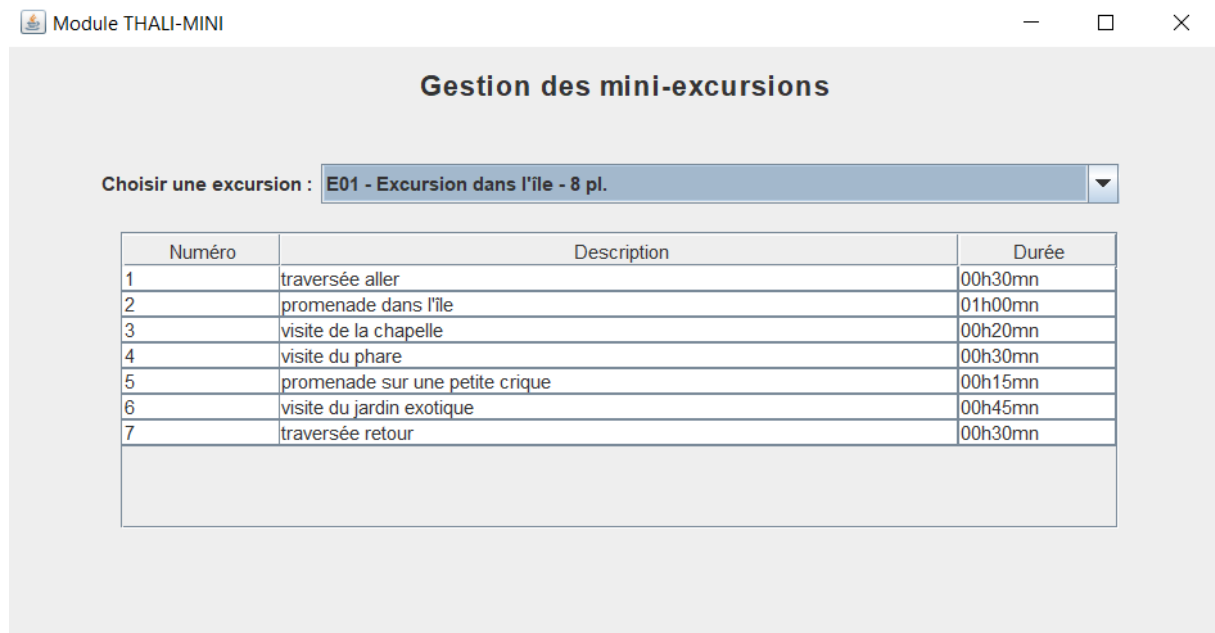
```
String codeExcursion;  
int numEtape;
```

Etape 3 – Modification d’une interface graphique

Etape 3-1)

1 – Modifier la fenêtre « JFrameLesExcursions » pour qu’elle affiche la durée totale de l’excursion courante sous le tableau

En lançant JFrameLesExcursions nous obtenons cet affichage



Pour afficher la durée totale de l’excursion, nous allons ajouter un JLabel nommé « JLabelTextTempsTotal » qui affichera le même texte à chaque fois et un JTextField nommé « JTextFieldTempsTotalCalculee » qui affichera une variable contenant la durée totale de l’excursion

Dans le code source de JFrameLesExcursions, on crée une variable DureeTotal qui va permettre de faire la somme de la durée de chaque mini-excursions. On instancie la variable de durée totale à 0 pour que la valeur de la variable soit réinitialisé à chaque fois que l’on change d’excursions.

```
int DureeTotal = 0;
```

Puis pour faire la somme on fait la valeur de la variable DureeTotal égal la durée total + la durée de l’étape

```
DureeTotal = DureeTotal + uneEtape.getDureePrevue();
```

Enfin pour afficher la durée avec le bon format on fait :

```
jTextFieldTempsTotalCalculee.setText(String.format("%1$02dh%2$02dmn",  
(DureeTotal / 60), (DureeTotal - (DureeTotal / 60) * 60)));
```


2 – Vérification

Choisir une excursion : **E01 - Excursion dans l'île - 8 pl.** ▼

Numéro	Description	Durée
1	traversée aller	00h30mn
2	promenade dans l'île	01h00mn
3	visite de la chapelle	00h20mn
4	visite du phare	00h30mn
5	promenade sur une petite crique	00h15mn
6	visite du jardin exotique	00h45mn
7	traversée retour	00h30mn

Temps total de l'excursion : 03h50mn

Etape 3-2)

1 – Modifier la fenêtre existante pour ajouter les boutons nécessaires pour démarrer chaque action et créer la nouvelle fenêtre qui permettra de travailler sur les attributs d'une excursion

On ajoute 4 boutons (création, lecture, modification et suppression) et on modifie le nom des boutons et leurs affichage et on obtient ceci

Gestion des mini-excursions

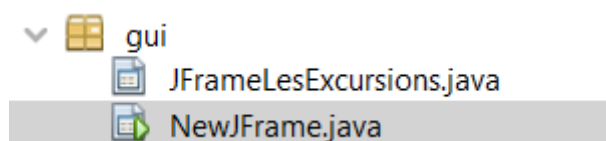
Choisir une excursion : **E01 - Excursion dans l'île - 8 pl.** ▼

Numéro	Description	Durée
1	traversée aller	00h30mn
2	promenade dans l'île	01h00mn
3	visite de la chapelle	00h20mn
4	visite du phare	00h30mn
5	promenade sur une petite crique	00h15mn
6	visite du jardin exotique	00h45mn
7	traversée retour	00h30mn

Temps total de l'excursion : 03h50mn

Création **Lecture** **Modification** **Suppression**

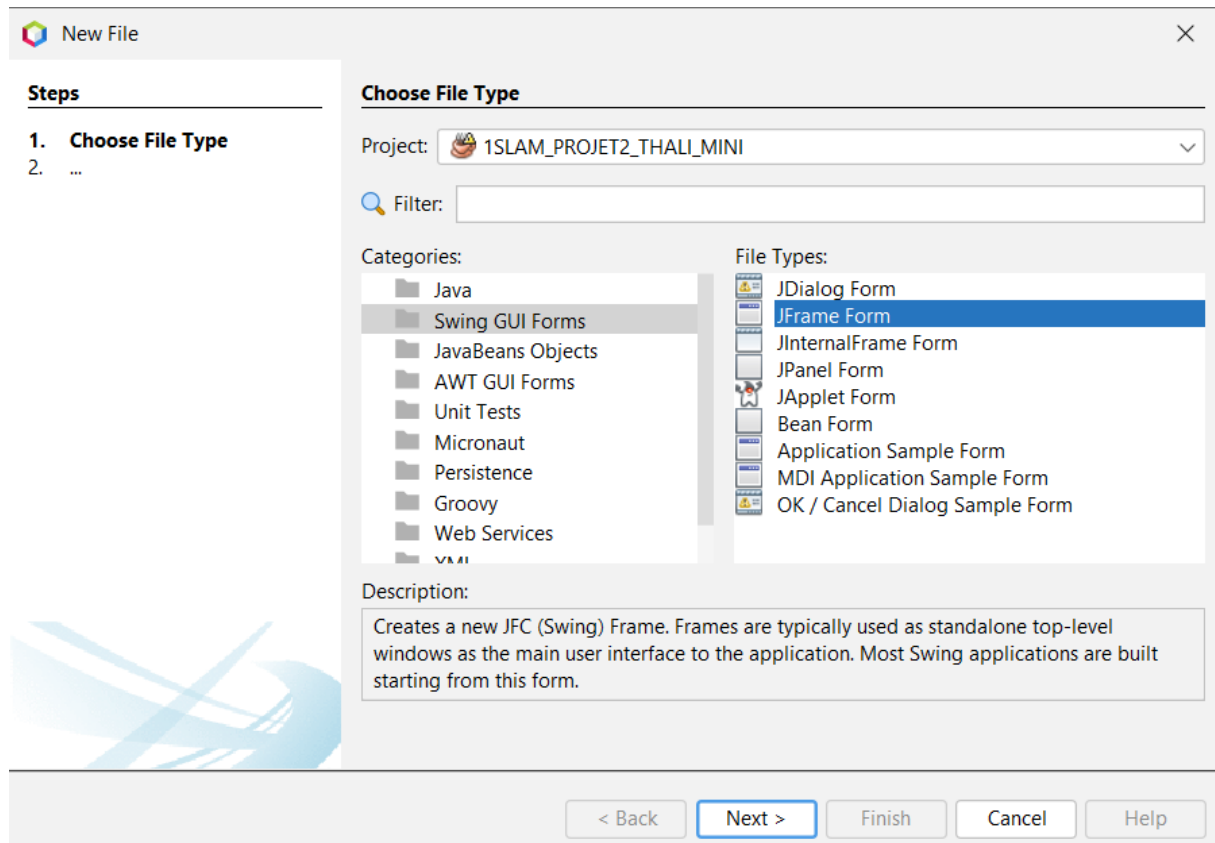
Puis nous créons une nouvelle fenêtre JFrame



Etape 4 – Création d'une excursion

1 – Créer, afficher et fermer la fenêtre de création d'une excursion

Pour créer une nouvelle fenêtre JFrame on crée un nouveau fichier JFrame Form

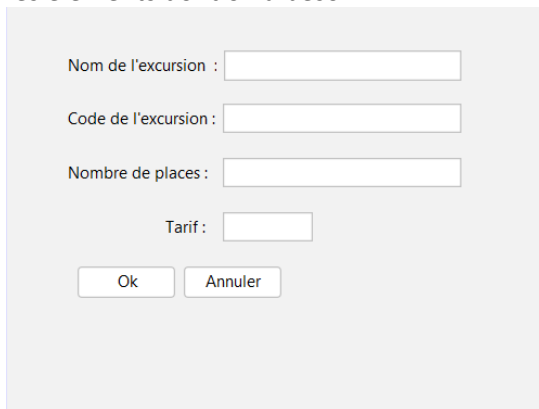


ensuite pour afficher la fenêtre, on rentre ce code dans JButtonAnnuler action perform

```
private void jButtonCreationActionPerformed(java.awt.event.ActionEvent evt) {  
    NewJFrame g = new NewJFrame();  
    g.setVisible(b: true);  
}
```

Pour permettre de fermer la fenêtre NewJFrame, on doit créer un bouton et ajouter le code `dispose()`;
Dans action perform

Après avoir fait en sorte qu'une nouvelle fenêtre s'ouvre et se ferme, on ajoute dans la fenêtre tout les éléments dont on a besoin



Une fois fait, on fait en sorte que chaque élément dans lequel on doit écrire ai une variable qui récupère le contenu rentré par l'utilisateur (exemple pour le nom de l'excursion)

```
private void jTextFieldNomExcursionActionPerformed(java.awt.event.ActionEvent evt) {  
    // récupérer le texte entré par l'utilisateur  
    String nomExcursion = jTextFieldNomExcursion.getText();  
}
```

Pour la suite, nous devons faire en sorte que les valeurs rentrés soit récupérés quand on clique sur le bouton Ok puis que ces informations soient rentrés dans la base de donnée

```
// récupérer les informations des jTextField  
String nomExcursion = jTextFieldNomExcursion.getText();  
String codeExcursion = jTextFieldCodeExcursion.getText();  
String nbPlaces = jTextFieldNombrePlaces.getText();  
String tarif = jTextFieldTarif.getText();  
String query = "INSERT INTO Excursion (Code, Libelle, NbPLaces, Tarif) VALUES (?, ?, ?, ?);";
```

Le dispose fait en sorte que après que les informations soient enregistrés dans la base de donnée, la fenêtre se ferme automatiquement.

En cas d'erreur on affiche une fenêtre pop-up qui affiche un message d'erreur. On a choisi de faire un système d'erreur simple afin de pouvoir avancer dans notre projet. On souhaite y revenir

```
try(PreparedStatement stmt = cnx.prepareStatement(query)){  
    stmt.setString(1, x:codeExcursion);  
    stmt.setString(2, x:nomExcursion);  
    stmt.setString(3, x:nbPlaces);  
    stmt.setString(4, x:tarif);  
    stmt.executeUpdate();  
    JOptionPane.showMessageDialog(parentComponent: null, message: "Excursion ajoutée !");  
}  
catch (SQLException ex) {  
    Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, msg: null, thrown: ex);  
    JOptionPane.showMessageDialog(parentComponent: null, message: "Mauvaise saisie des valeurs !");  
}
```

Pour le déroulé alternatif du cas d'utilisation, on crée un bouton annuler dans lequel, on met *dispose()*;

Qui permet de fermer la fenêtre.

```
private void jButtonAnnulerActionPerformed(java.awt.event.ActionEvent evt) {  
    dispose();  
}
```

Etape 5 – Visualisation d'une excursion

1 – Créer la fenêtre de visualisation

Dans un premier temps nous faisons une nouvelle fenêtre et nous créons un interface graphique

Visualisation d'une excursion

Code de l'excursion :

Nom de l'excursion :

Nombre de places :

Tarif :

Quitter

Puis nous créons un bouton qui permet d'aller sur notre nouvelle fenêtre depuis notre fenêtre principale

Gestion des mini-excursions

Choisir une excursion : Item 1

Title 1	Title 2	Title 3	Title 4

Temps total de l'excursion :

Visualiser l'excursion

Création Lecture Modification Suppression Ajouter une excursion

Nous rajoutons le code qui permet d'ouvrir et fermer la fenêtre

Dans la fenêtre principale pour accéder à la nouvelle fenêtre

```
private void jButtonVisualiserActionPerformed(java.awt.event.ActionEvent evt) {  
    JFrameVisualisationExcursion vis = new JFrameVisualisationExcursion();  
    vis.setFenetrePrincipale( fenetrePrincipale: this );  
    vis.setVisible( b: true );  
    this.setEnabled( b: false );  
}
```

Dans la nouvelle fenêtre pour la fermer et accéder à la fenêtre principale

```
private void jButtonQuitterActionPerformed(java.awt.event.ActionEvent evt) {  
    fenetrePrincipale.setEnabled( b:true);  
    dispose();  
}
```

2 – Afficher les données de l'excursion sélectionné

Sur la fenêtre principale nous faisons en sorte de pouvoir afficher la nouvelle fenêtre et nous envoyons les informations de la fenêtre principale a la nouvelle fenêtre

```
MiniExcursion excursionCourante = (MiniExcursion) modeleJComboExcursions.getElementAt( index:iSelect);  
JFrameVisualisationExcursion vis = new JFrameVisualisationExcursion();  
vis.setFenetrePrincipale( fenetrePrincipale:this);  
vis.setExcursionAffichee( excursionAffichee:excursionCourante);  
vis.afficherDonnees();  
vis.setVisible( b:true);  
this.setEnabled( b:false);  
MiniExcursionPayante excursionCourantePayante = (MiniExcursionPayante) modeleJComboExcursions.getElementAt(  
vis.setExcursionAfficheePayante( excursionAfficheePayante:excursionCourantePayante);  
vis.afficherDonnesPayante());
```

Dans un premier temps nous devons nous connecter a la base de donnée

```
try {  
    Connection cnx = ConnexionBDD.getConnexion();  
} catch (SQLException ex) {  
    Logger.getLogger( JFrameAjouterExcursion.class.getName()).log( level:Level.SEVERE, msg:null, thrown:ex);  
}
```

Puis nous allons chercher les informations de l'excursion sélectionné

```
// récupérer les informations des jTextField  
String nomExcursion = jTextFieldNomExcursion.getText();  
String codeExcursion = jTextFieldCodeExcursion.getText();  
String nbPlaces = jTextFieldNombrePlaces.getText();  
String tarif = jTextFieldTarif.getText();
```

On rajoute des méthodes qui permettent de récupérer les informations à afficher

```
public void setExcursionAffichee(MiniExcursion excursionsAffichee){
    this.excursionAffichee = excursionsAffichee;
}

public void setExcursionAfficheePayante(MiniExcursionPayante excursionAfficheePayante){
    this.excursionAfficheePayante = excursionAfficheePayante;
}

public void afficherDonnees(){
    jTextFieldCodeExcursion.setText(t:excursionAffichee.getCode());
    jTextFieldNomExcursion.setText(t:excursionAffichee.getLibelle());
    jTextFieldNbPlaces.setText(String.valueOf(i:excursionAffichee.getNbPlaces())+" places");
    jTextFieldTarif.setText(t:"0,00€");
}

public void afficherDonneesPayante(){
    jTextFieldCodeExcursion.setText(t:excursionAfficheePayante.getCode());
    jTextFieldNomExcursion.setText(t:excursionAfficheePayante.getLibelle());
    jTextFieldNbPlaces.setText(String.valueOf(i:excursionAfficheePayante.getNbPlaces())+" places");
    jTextFieldTarif.setText(String.valueOf(i:excursionAfficheePayante.getTarif())+"€");
}
```