

# Projet commun

## Logiciel de réservation

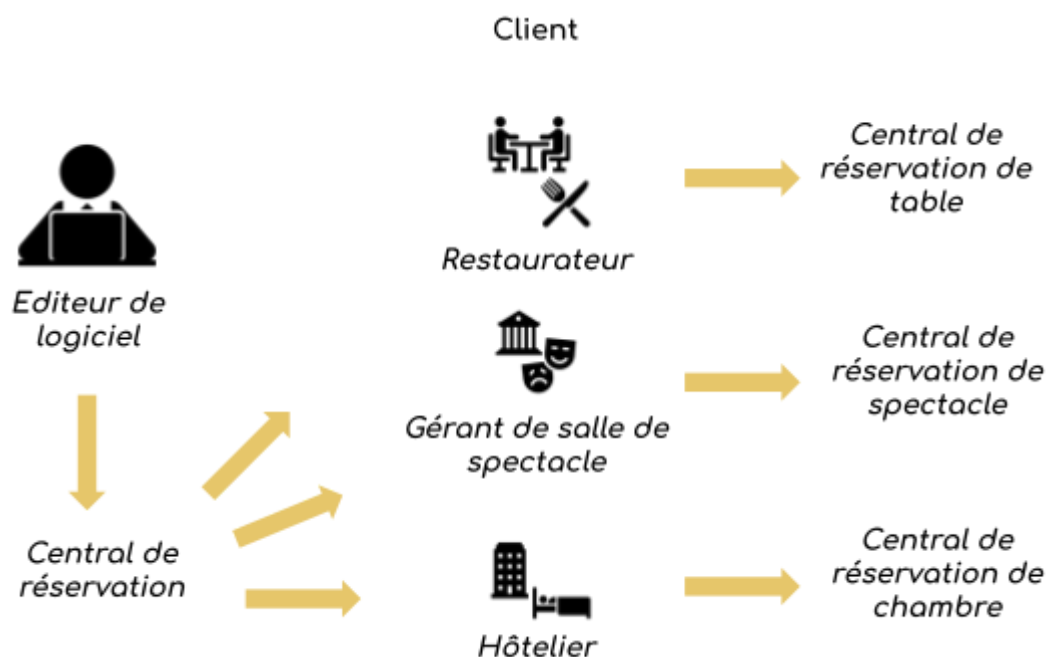
### 1 Description globale du sujet et attentes pédagogiques

Le TP commun porte sur la réalisation d'un logiciel permettant de réaliser des réservations. Ce logiciel peut être déployé chez des clients de métiers différents. Ces clients cibles peuvent être restaurateur, gérant de salle de spectacle ou hôtelier.

Les demandes de réservation peuvent donc prendre différentes formes, par exemple :

- pour un restaurant on aura besoin d'un nombre de convives,
- pour une salle de spectacle, de la zone du siège que l'on veut réserver,
- pour un hôtel, de la taille du lit (simple, double, enfant).

Pourtant nous, éditeurs de logiciel, nous souhaitons créer un logiciel permettant de s'adapter à n'importe quel client.



L'objectif en POO est de vous faire travailler sur les différentes notions vues en cours, en autonomie et en TP :

- les exceptions,
- les classes internes,
- les classes abstraites,
- les interfaces,
- les classes génériques,
- l'architecture ECB.

## 2 Première étape : définir les différentes fonctionnalités auxquelles devra répondre le logiciel

### Fonctionnalité

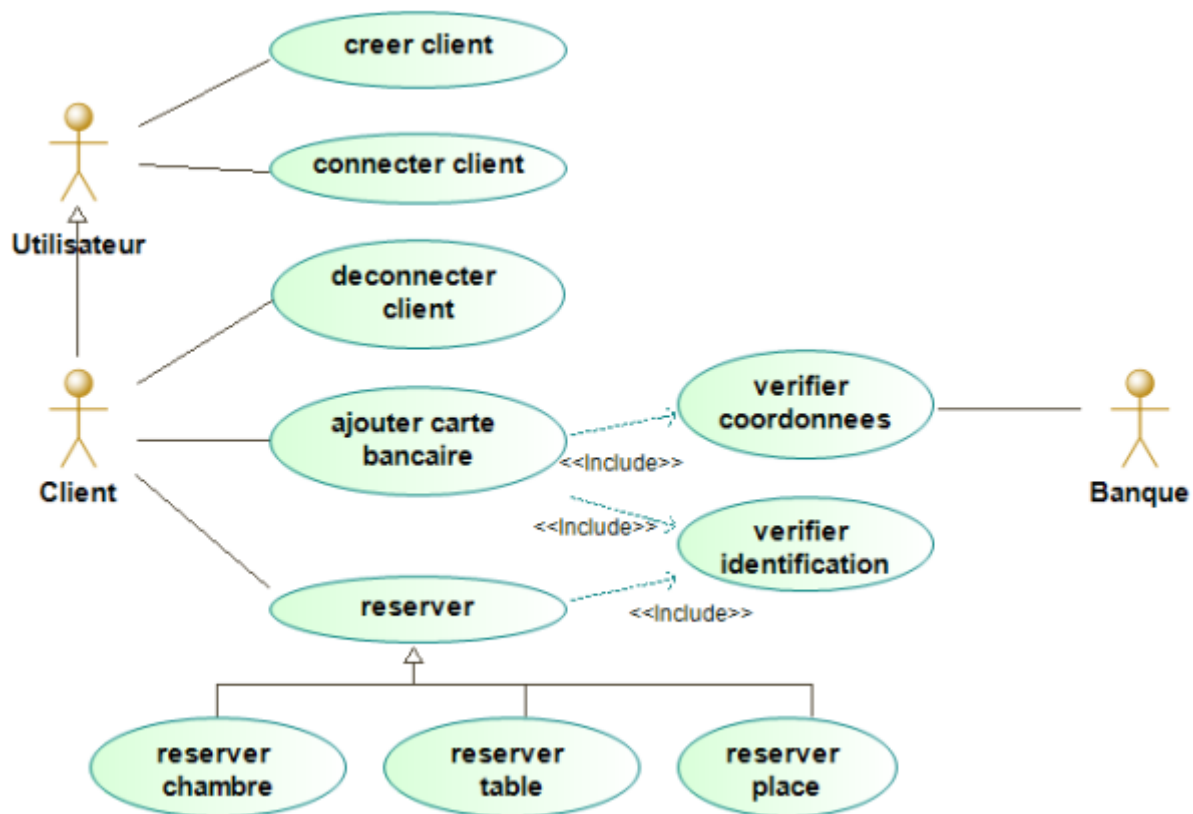
L'utilisateur pourra :

- se créer un compte,
- se connecter,

Le client pourra :

- ajouter une carte de crédit (carte dont la validité sera vérifiée par la banque),
- effectuer une réservation
- se déconnecter

### Diagramme de cas d'utilisation



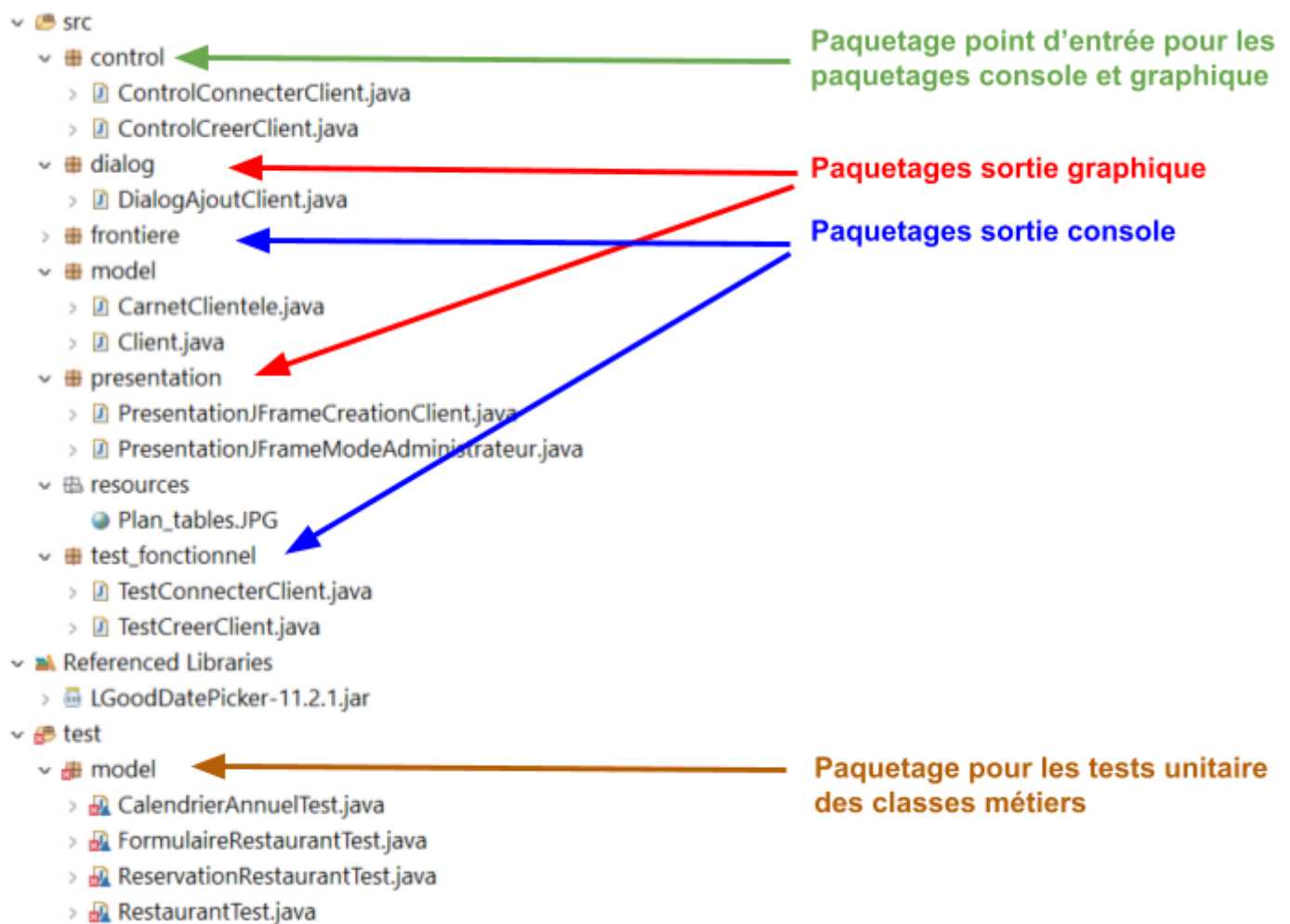
Les cas "créer client" et "connecter client" ont été traités en cours. Dans ce TP nous allons traiter le cas de la réservation.

### 3 Deuxième étape : Comprendre l'architecture existante

Le cas "créer client" a été traité en cours d'IHM et en cours de POO. Le cas "connecter client" a été traité en TD de POO.

Le code java de ses 2 cas sont fournis, récupérer le code sur git lien sous Moodle.

#### Architecture du logiciel initiale



## Premier cas : créer client

Lancer le cas "créer client" avec une sortie console : exécuter le main de la classe "TestCreerClient" du paquetage "test\_fonctionnel".

Exemple d'exécution :

```
----- CREATION CLIENT -----
```

Nom :

CHAUDET

Prénom :

Christelle

Adresse mail :

christelle.chaudet@irit.fr

Mot de Passe :

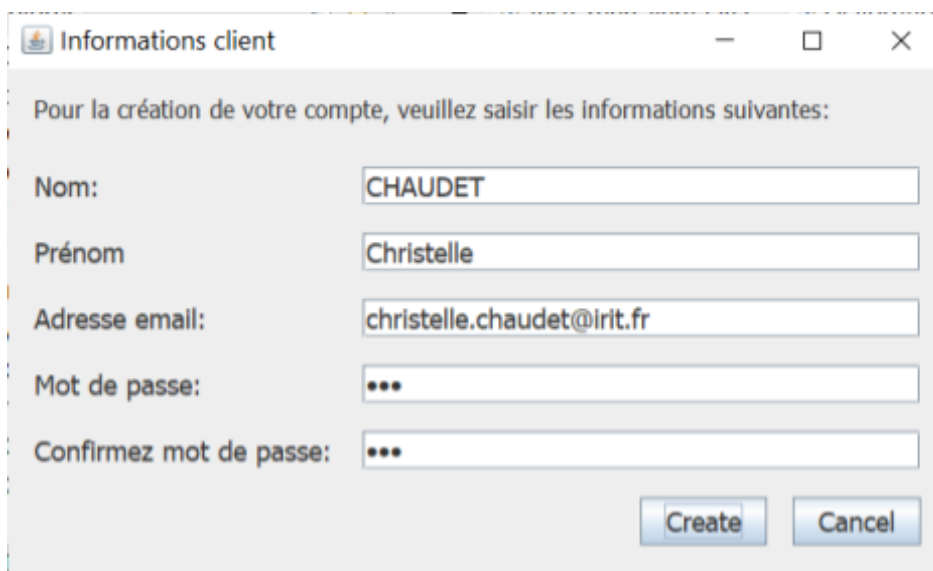
mdp

```
----- CONTROLE DES DONNEES -----
```

```
nom=CHAUDET,      prenom=Christelle,      adresseMail=christelle.chaudet@irit.fr,  
mdp=mdp
```

Lancer ce cas avec une sortie graphique : exécuter le main de la classe "DialogAjoutClient" du paquetage dialog.

Exemple d'exécution :



Informations client

Pour la création de votre compte, veuillez saisir les informations suivantes:

Nom: CHAUDET

Prénom: Christelle

Adresse email: christelle.chaudet@irit.fr

Mot de passe: \*\*\*

Confirmez mot de passe: \*\*\*

Create Cancel

### Analyse du code

Que ce soit la classe "BoundaryCreerClient" (ligne 24) ou "DialogAjoutClient" (ligne 52) les deux classes utilisent la méthode "creerClient" du contrôleur "ContrôleurCreerClient". A partir de là le contrôleur appelle la méthode "creerClient"

de l'entité "*CarnetClientele*" afin de créer un objet de type "*Client*" et le stocker dans tableau "*clients*".

## Deuxième cas : connecter client

Le cas "connecter client" est seulement codé en sortie console.

Lancer ce cas : exécuter le main de la classe "TestConnecterClient" du paquetage "test\_fonctionnel".

Exemple d'exécution :

```
----- CREATION CLIENT -----
Nom :
CHAUDET
Prénom :
Christelle
Adresse mail :
christelle.chaudet@irit.fr
Mot de Passe :
mdp
----- CONNECTION CLIENT -----
Adresse mail :
christelle.chaudet@irit.fr
Mot de passe :
mdp
Vous êtes connecté

----- CONTROLE DES DONNEES -----
nom=CHAUDET,      prenom=Christelle,      adresseMail=christelle.chaudet@irit.fr,
mdp=mdp
```

Si le mot de passe n'est pas le bon :

```
----- CONNECTION CLIENT -----
Adresse mail :
christelle.chaudet@irit.fr
Mot de passe :
mauvaisMDP
Votre mail ou votre mot de passe est erroné
```

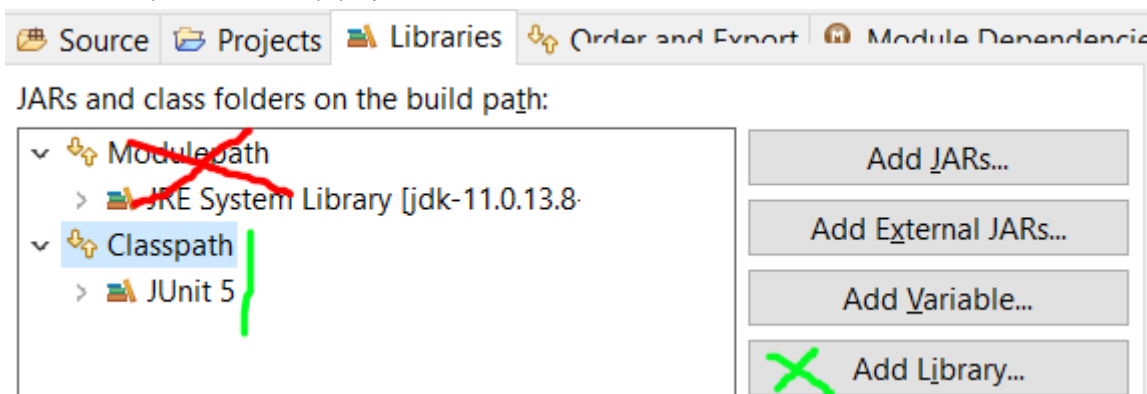
## 4 Troisième étape : Coder les nouveaux cas

Nous allons commencer par coder le cas "réserver" sans nous préoccuper du cas inclus "vérifier identification" ou de payer avec une carte bancaire (qui dépend du cas "ajouter carte bancaire"). Puis dans les réservations nous commencerons par la réservation d'une table au restaurant afin de pouvoir connecter l'interface que vous avez codée dans le TP d'IHM. Vous poursuivrez par les autres réservations (chambre d'hôtel, place de spectacle).

Pour tester vos classes au fur et à mesure nous vous fournissons des tests unitaires que vous complétez dans la votre thématique Test.

Afin de pouvoir lancer les test vous devez ajouter la bibliothèque JUnit version Jupiter :

- cliquer droit sur votre projet
- sélectionner "Properties"
- sélectionner "Java Build Path"
- sélectionner l'onglet "Libraries"
- sélectionner "classpath" (et pas ModulePath)
- appuyer sur le bouton "Add Library"
- sélectionner "JUnit" et cliquer sur le bouton "next"
- sélectionner "JUnit 5" et cliquer sur "Finish"
- cliquer sur "Apply and close"



Les autres cas pour avoir un logiciel complet seront à effectuer en autonomie : "déconnecter client", "ajouter carte bancaire" et "vérifier identification".