**TD2 EJB Session Sans Etat**

**VS**

**EJB Session Avec Etat**

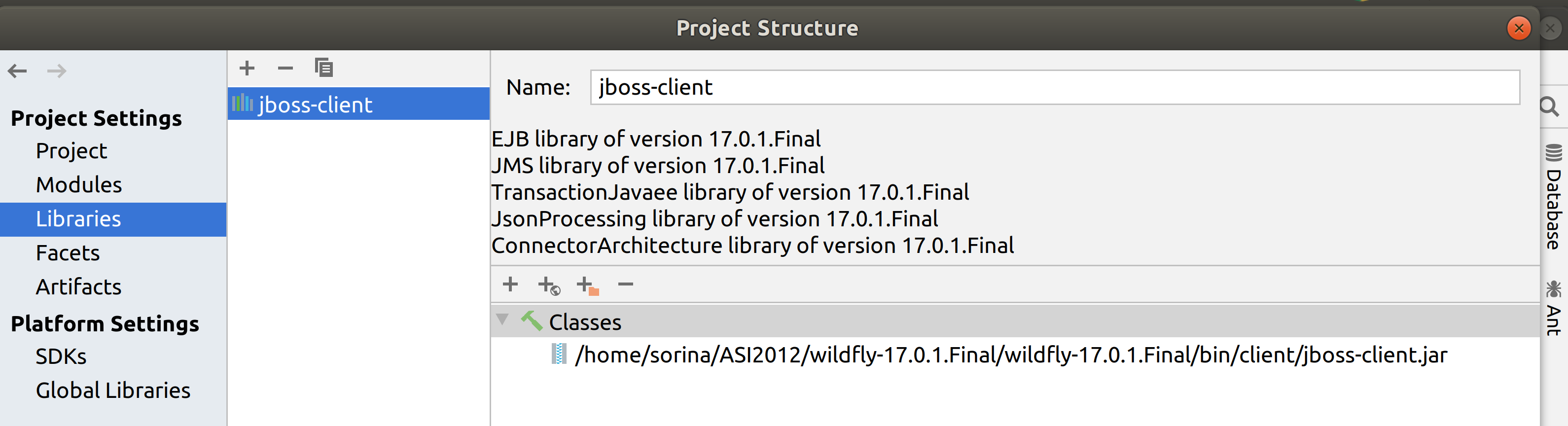
## Exercice 1 : Ecriture d’une application client

Nous reprenons le projet ProjetEJB de TD2. Nous allons écrire une application client en Java, qui

fera executer les méthodes de vos classes ejb via le protocole RMI.

Dans IntelliJ, vous créez un simple projet Java.

Dans **l’onglet Libraries** ajoutez la bibliothèque nécessaire côté client pour le fonctionnement d’une application utilisant un EJB.



## 1. Ajouter la classe client

**public** **class** HelloClient {

**static** {

Security.*addProvider*(**new** JBossSaslProvider());

}

**public** **static** **void** main(String[] args) **throws** Exception {

// Appel du stateless bean

HelloRemote helloBean = *lookupRemoteStatelessHello*();

System.*out*.println(helloBean.sayHello());

}

// Connexion au serveur et lookup du bean

**private** **static** HelloRemote lookupRemoteStatelessHello() **throws** NamingException {

HelloRemote remote=**null**;

**try** {

Properties jndiProps = **new** Properties();

jndiProps.setProperty(Context.*URL\_PKG\_PREFIXES*, "org.jboss.ejb.client.naming");

InitialContext ctx = **new** InitialContext(jndiProps);

remote = (HelloRemote) ctx.lookup("ejb:/ProjetEJB\_war\_exploaded/Hello!hw.HelloRemote");

} **catch** (Exception e) {

e.printStackTrace();

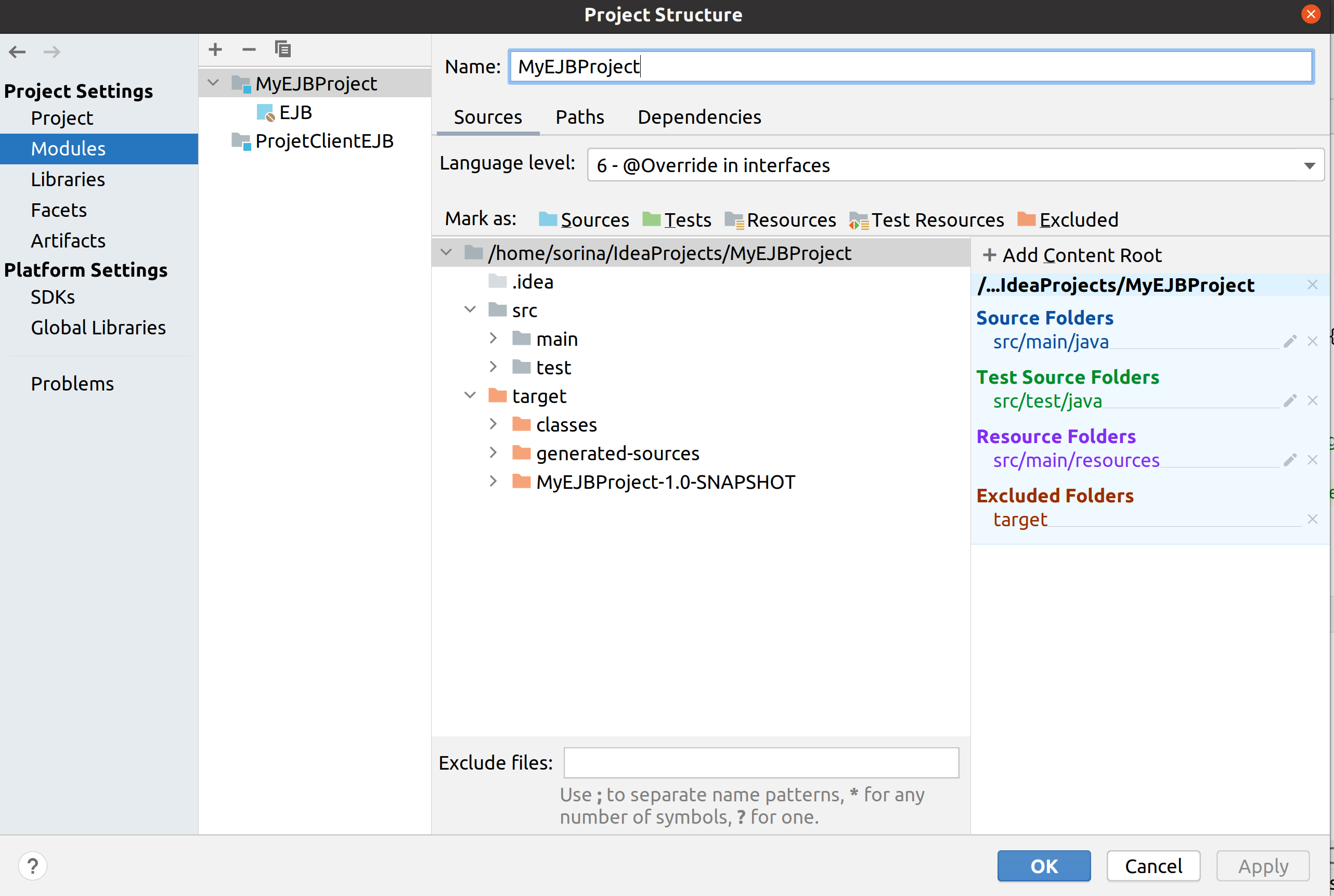
}

**return** remote;

}

}

Depuis Project structure, il faudrait inclure également le projet EJB écrit pour le côté serveur :



## 2. Paramètres de connexion à JBOSS

Il ne nous reste plus qu’à donner les informations de connexion à JBoss. Comme toujours en J2EE, ces paramètres ne sont pas stockés « en dur » mais dans un fichier xml. Dans Eclipse, clic droit sur ProjetEJBClient\src > New > Other > General > File. Créer un fichier jboss-ejb-client.properties dans ProjetEJBClient/src.

Copier dans ce fichier les informations qui permettront à l’application cliente de se connecter au service de nommage du serveur JBoss :

**java.naming.factory.initial**=**org.jnp.interfaces.NamingContextFactory**

**java.naming.factory.url.pkgs**=**org.jboss.naming:org.jnp.interfaces**

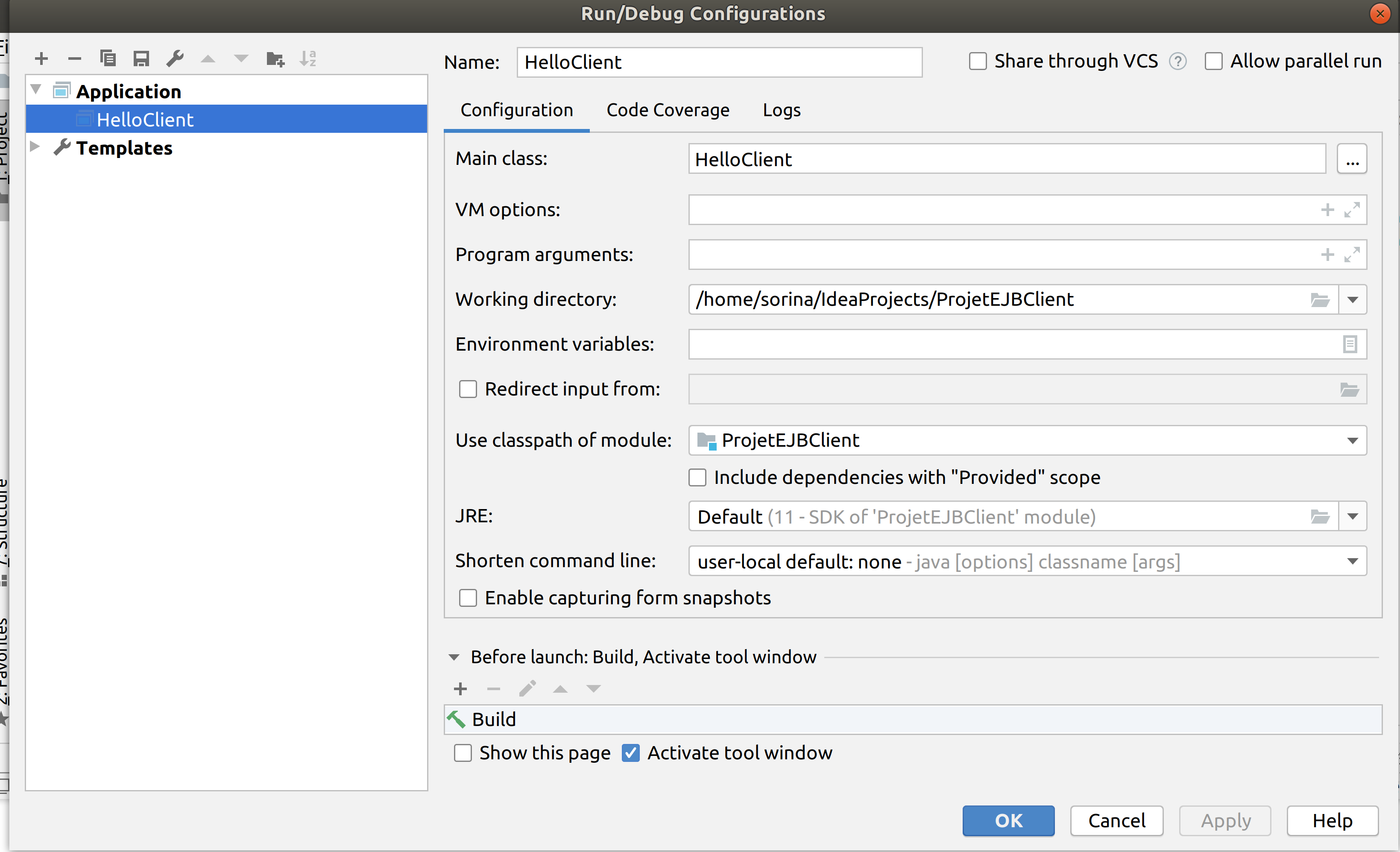
**remote.connections**=**default**

**remote.connection.default.host**=**localhost**

**remote.connection.default.port** = **8080**

## 3. Exécution de l’application

Afin de compiler et exécuter le code, vous rajoutez une configuration Application, comme ci-dessous :



# 

# 

# Exercice 2: le tableau d’amortissement

Dans ce TD non guidé, vous allez écrire un EJB session. A vous de déterminer si l'application nécessite un EJB avec

état ou sans état.

Une banque souhaite mettre à disposition de ses clients la génération de tableaux d’amortissements de prêts. Vous devez réaliser une application client qui génère le tableau d’amortissement d’un emprunt du type à partir des données saisies par le client (montant du prêt, durée, début du prêt et taux d’intérêt annuel). Le client peut relancer le calcul autant de fois qu’il le souhaite. Ce n’est pas l’application du client mais le serveur J2EE qui fait les calculs d’amortissement dans un EJB.

|  |
| --- |
|  |

****

**1.1. Le client**

**Création d’un JFrame nomme AmortissementClient**

Toujours dans le projet que nous avons créé au début, créez un package pour le côté client et ajoutez une JFrame (dans la catégorie New->Other/Swing designer/JFrame).

Recuperez le code la classe AmortissementClient.java  depuis la page du cours.

Ensuite ajoutez les éléments d’interface suivants :

* Le bouton Envoyer (JButton) pour créer le lien avec l'EJB et récuperer le résultat de la requête.
* Un tableau (JTable) permet d’afficher le résultat de la requête

**1.2. Le serveur**

L'application serveur effectue le calcul des mensualités a partir des données envoyées par le client.

La mensualité d'un prêt se calcule avec la formule :

mensualité= (montant\_total\*t/12)/(1-(1+t/12)-n) , ou t est la taux d'intérêt et n est le nombre de mensualités.