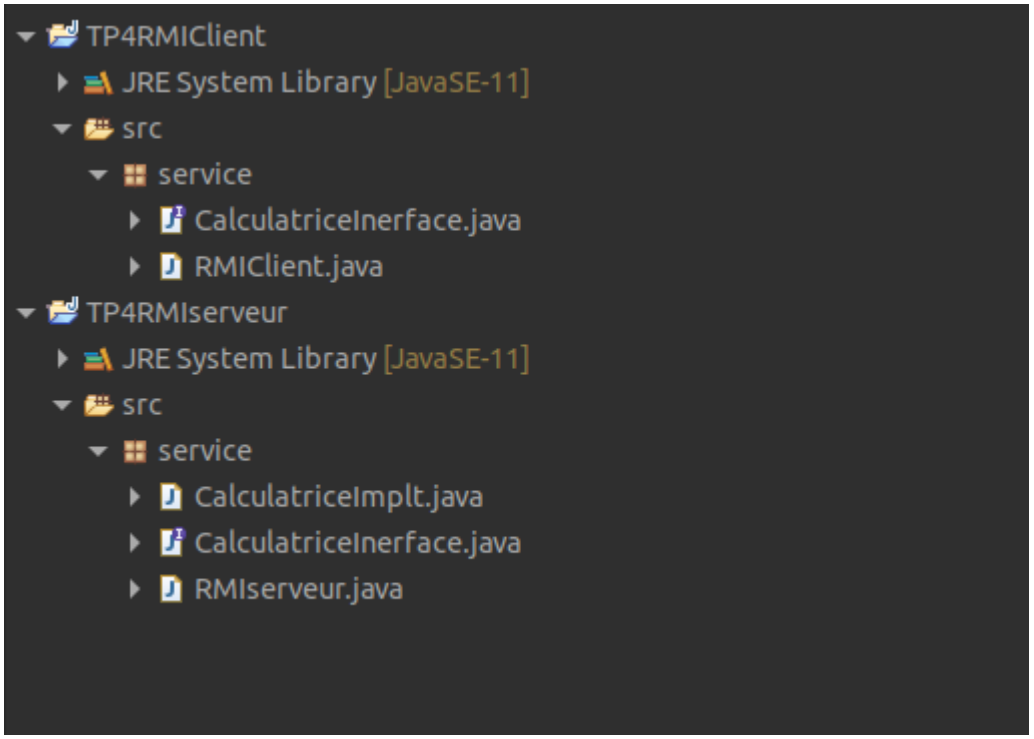
TP 4

La structure des fichiers des projets créés à l'aide d'Eclipse tout au long de ce TP est illustrée ci-dessous :



1. Côté serveur :

➔ Créer un nouveau projet TP4RMIServeur :

➔ Sous le projet TP4RMIServeur, sélectionnez **New -> Interface**

➔ Définissez le nom de l'interface comme suit : **CalculatriceInterface –>Cliquez sur Terminer.**

➔ Ajoutez le code suivant dans CalaculatriceInterface.java :

CalculatriceInerface.java

```
1  package service;
2
3  import java.rmi.Remote;
4  import java.rmi.RemoteException;
5
6  public interface CalculatriceInerface extends Remote{
7      public int somme(int a,int b)throws RemoteException;
8      public int soustraction(int a,int b)throws RemoteException;
9      public int multiplication(int a,int b)throws RemoteException;
10     public float division(int a,int b)throws RemoteException;
11 }
```

➔ Sélectionnez le projet TP4RMIServeur, cliquez sur **New -> Class**, définissez le nom de la classe comme : **CalculatriceImplt**

➔ Ajoutez le code suivant dans CalaculatriceImplt.java :

CalculatriceImplt.java

```java
package service;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
@SuppressWarnings("serial")
public class CalculatriceImplt extends UnicastRemoteObject implements CalculatriceInerface{

    protected CalculatriceImplt() throws RemoteException {
        super();
    }

    @Override
    public int somme(int a, int b) throws RemoteException {
        return a+b;
    }
    @Override
    public int soustraction(int a, int b) throws RemoteException {
        return a-b;
    }
    @Override
    public int multiplication(int a, int b) throws RemoteException {
        return a*b;
    }
    @Override
    public float division(int a, int b) throws RemoteException {
        //Si vous ne convertissez pas explicitement l'une des deux valeurs en float avant de procéder
        //à la division, une division entière sera utilisée.
        //Il suffit que l'un des deux opérandes soit une valeur à virgule flottante pour que
        //la division normale soit utilisée (
        return (float)a/b;
            }
```

➔ Sélectionnez le projet TP4RMIServeur, cliquez sur **New -> Class**, définissez le nom de la classe comme : RMIServeur

➔ Ajoutez le code suivant dans RMIServeur.java :

RMIserveur.java

```java
package service;

import java.rmi.Naming;
import java.rmi.registry.LocateRegistry;

public class RMIserveur {

    public static void main(String[] args) {

        try {

            LocateRegistry.createRegistry(1099);
            CalculatriceImplt objetDistant = new CalculatriceImplt();
            Naming.rebind("rmi://localhost:1099/calculatrice", objetDistant);
            System.out.println(objetDistant.toString());
            System.out.println("Le serveur est prêt !");

        } catch (Exception e) {
            System.out.println("Échec de l'exécution du serveur :" +e);
        }
    }

}
```

➔ Cliquez sur Run :

Console

```
RMIserveur (1) [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (2 déc. 2021 à 21:49:29)
CalculatriceImplt[UnicastServerRef [liveRef: [endpoint:[127.0.1.1:34201](local),objID:[-580186fd:17d7ce8c414:-7fff, 607499196854282684 3]]]]
Le serveur est prêt !
```

2. Côté Client :

➔ Créer un nouveau projet TP4RMIClient

➔ Copier-Coller le Package ‘service’ depuis ‘TP4RMIServeu’r vers le projet ‘TP4RMIClient’.

➔ TP4RMIClient: Supprimer les deux classes et sauvegarde le fichier interface.

➔ Sélectionnez le projet TP4RMIClient, cliquez sur **New -> Class(**avec la méthode main()**)**, définissez le nom de la classe comme : RMIClient.

➔ Durant cette phase, le client RMI demande à la RMI registry de lui donner le serveur RMI associé à un certain nom dans son annuaire.Ajoutez le code suivant dans RMIClient.java :

RMIClient.java

```java
package service;

import java.rmi.Naming;
import service.CalculatriceInerface;
import java.util.Scanner;

public class RMIClient {

    public static void main(String[] args) {
        @SuppressWarnings("resource")
        Scanner saisirParClavier = new Scanner(System.in);

        // connecter à l'annuaire

        try{
            CalculatriceInerface stub = (CalculatriceInerface) Naming.lookup("rmi://localhost:1099/calculatrice");
            System.out.println("Le client est connecté au serveur");
            System.out.println("sélectionnez n'importe quelle option dans le menu");
            System.out.println("1.addition \n");
            System.out.println("2.Soustraction \n");
            System.out.println("3.Multiplication \n");
            System.out.println("4.Division \n");
            int choix  = saisirParClavier.nextInt();
            int x,y;
            switch(choix)
            {
                case 1:
                {
                    System.out.println("entrez le premier entier");
                    x = saisirParClavier.nextInt();
                    System.out.println("entrez le deuxième entier");
                    y = saisirParClavier.nextInt();
                    System.out.println("La somme de: "+x+" plus "+y+" vaut : "+stub.somme(x, y));
                    break;

                }
                   case 2:
                {
                    System.out.println("entrez le premier entier");
                    x = saisirParClavier.nextInt();
                    System.out.println("entrez le deuxième entier");
                    y = saisirParClavier.nextInt();
                    System.out.println("La différence entre: "+x+" et "+y+" vaut : "+stub.soustraction(x, y));
                    break;

                }
                   case 3:
                {
                    System.out.println("entrez le premier entier");
                    x = saisirParClavier.nextInt();
                    System.out.println("entrez le deuxième entier");
                    y = saisirParClavier.nextInt();
                    System.out.println("La multiplication de: "+x+" par "+y+" vaut : "+stub.multiplication(x, y));
                    break;
                }
```

```
                    case 4:
                {
                    System.out.println("entrez le premier entier");
                    x = saisirParClavier.nextInt();
                    System.out.println("entrez le deuxième entier");
                    y = saisirParClavier.nextInt();
                    System.out.println("La division de: "+x+" par "+y+" vaut : "+stub.division(x, y));
                    break;

                }
                    default:
                {
                    System.out.println("Choix incorrect");
                    break;
                }
            }
        }catch (Exception e) {
            System.out.println("Échec de connection avec le serveur :" +e);

        }

    }
}
```

➔ Sélectionnez le projet TP4RMIServeur, cliquez sur :**Run As -> java application**

➔ Sélectionnez le projet TP4RMIClient, cliquez sur :**Run As -> java application**

3. Suite TP4 :

Refaire l'exercice Calculatrice mais cette fois-ci :

- avec 4 interfaces , une interface pour chaque méthode.

TP5

Refaire TP4 “Calculatrice” mais cette fois-ci, le programme doit permettre à l'utilisateur de saisir deux **nombres** et un **opérateur** (+, -, / et *),puis affiche le résultat **selon l'opérateur entré au clavier.**

Côté serveur :

- ➔ Créer un nouveau projet TP5RMIServeur :
- ➔ Sous le projet TP5RMIServeur, sélectionnez **New -> Interface**
- ➔ Définissez le nom de l'interface comme suit : **CalculatriceInterface –>Cliquez sur Terminer.**
- ➔ Ajoutez le code suivant dans CalaculatriceInterface.java :

CalculatriceImplt.java | CalculatriceInerface.java | RMIserveur.java | RMIClient.java

```
1  package service;
2  import java.rmi.RemoteException;
3  import java.rmi.server.UnicastRemoteObject;
4  @SuppressWarnings("serial")
5  public class CalculatriceImplt extends UnicastRemoteObject implements CalculatriceInerface{
6
7      protected CalculatriceImplt() throws RemoteException {
8          super();
9      }
10
11     @Override
12     public double somme(double a, double b) throws RemoteException {
13         return a+b;
14     }
15     @Override
16     public double soustraction(double a, double b) throws RemoteException {
17         return a-b;
18     }
19     @Override
20     public double multiplication(double a, double b) throws RemoteException {
21         return a*b;
22     }
23     @Override
24     public double division(double a, double b) throws RemoteException {
25         return a/b;
26             }
27
28 }
```

- ➔ ……
- ➔ ……

Exemple d'exécution:

```
Le client est connecté au serveur,saisissez l'opération comme suit : x opérateur y , exemple 5+5.
9/6
Opération: 9/6
Résultat : 1.5
```