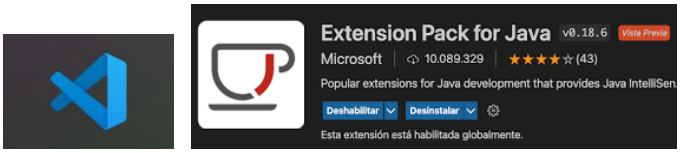
Compte Rendu

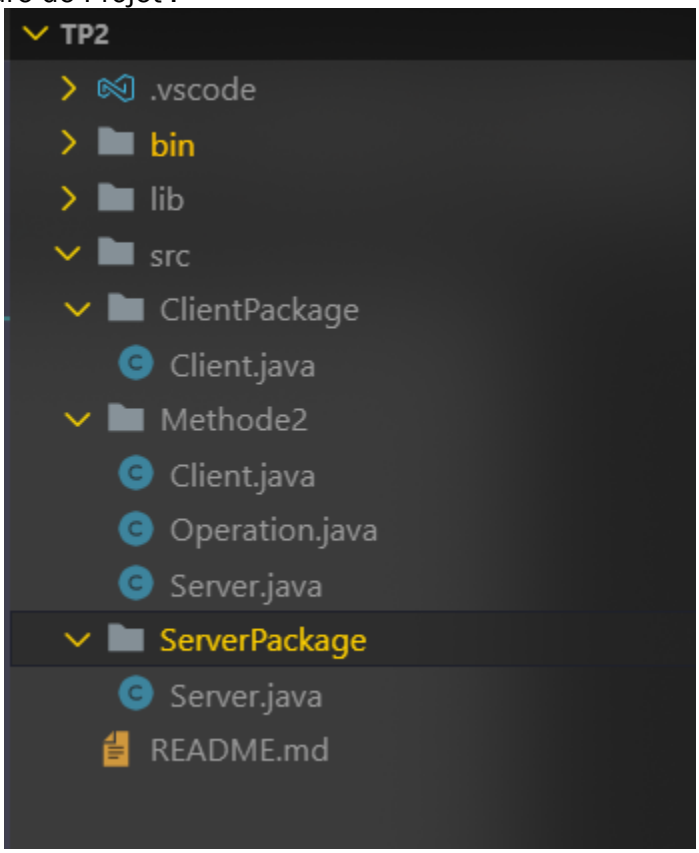
Tp1 : Développement d'applications réparties
Hammami Mouhamed Yassine
GLSI3-1C-gp2

Environnement de travail :

- L'éditeur du text VSCode + Java Extension pack :



- Structure de Projet :



- Serveur.java

```java
package ServerPackage;

import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;

public class Server {
  public static void main(String[] args) throws Exception {
    ServerSocket ss = new ServerSocket(2000);
    Socket s = ss.accept();
    System.out.println("Client connecté");
    // lecture et ecriture seulement avec un seul octet
    InputStream is = s.getInputStream();
    OutputStream os = s.getOutputStream();
    // lecture et ecriture seulement avec plus un seul octet
    InputStreamReader isr = new InputStreamReader(is);
    BufferedReader bfrIn = new BufferedReader(isr);
    PrintWriter pw = new PrintWriter(os, true);
    // initialization result
    float value = 0;
    String expression = bfrIn.readLine();
    try {

  // ==> On assume que l'oeration est de la forme suivante :3+4/3-4/3/4/3*4 alors

  // si une opérateur existe dans l'expression alors son index est le maximum
      // parmis les autre opérateurs
      int pos = Math.max(expression.indexOf('+'),
          Math.max(expression.indexOf('-'), Math.max(expression.indexOf('/'),
              expression.indexOf('*'))));
      System.out.println("pos=" + pos);
      char opr = expression.charAt(pos);
      System.out.println("opr=" + opr);
      // extraction des opérandes
      float opr1 = Float.parseFloat(expression.substring(0, pos));
      float opr2 = Float.parseFloat(expression.substring(pos + 1));
      // la logique des calcul
      switch (opr) {
        case '+':
          value = opr1 + opr2;
          break;

        case '-':
          value = opr1 - opr2;
          break;
        case '/':
          value = opr1 / opr2;
          break;
        case '*':
          value = opr1 * opr2;
          break;
        default:
          System.out.println("Opération ivalide");
          break;
      }
      // gestion des exceptions
    } catch (Exception e) {
      pw.println("Opération ivalide");
      System.out.println(e.toString());
      System.exit(1);
    }
    // envoi des résultas valide
    System.out.println("Result : " + value);
    pw.println("Result : " + value);

    ss.close();
  }
}

```

Explication de la logique :

Les données reçus de l'utilisateur sont sous la forme d'une CC,

Alors on doit extraire les opérandes et l'opérateur et selon cette dernière valeur on effectue le calcule

L'operateur :

On assume que la forme de l'opération est ***oprd op oprd*** alors on peut avoir un seul opérateur (+,-,/,*) donc sa position est la maximum entre les positions des qui ne existent pas (pos=-1) .

Les opérandes :

Les opérandes sont les sous chaine qui précèdent et suivent l'opérande

Notez qu'il faut les transformer en entier

- Client.java

```java
package ClientPackage;

import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.PrintWriter;
import java.net.Socket;

public class Client {
  public static void main(String[] args) throws Exception {
    Socket s = new Socket("localhost", 2000);
    System.out.println("Client Connecté au serveur");
    // lecture et ecriture seulement avec un seul octet
    InputStream is = s.getInputStream();
    OutputStream os = s.getOutputStream();
    // lecture et ecriture seulement avec plus un seul octet
    InputStreamReader isr = new InputStreamReader(is);
    BufferedReader bfrIn = new BufferedReader(isr);
    PrintWriter pw = new PrintWriter(os, true);

    // lecture des dinnées utilisateur de taille supérieur qu'un octet
    InputStreamReader isrSys = new InputStreamReader(System.in);
    BufferedReader bfrInSys = new BufferedReader(isrSys);

    // Demander d'utlisateur d'ecrire
    System.out.println("donnez un nombre pour envoyer au serveur");
    String val = bfrInSys.readLine();
    // envoi d'expression
    pw.println(val);
    // lire le résultat
    String res = bfrIn.readLine();
    System.out.println(res);

    s.close();
  }
}
```

Résultat:


TERMINAL
pwsh - ClientPackage
Yacin
ServerPackage
main ≡ ?1
3ms
jav
a .\Server.java
Client connecté
pos=3
opr=+
Result : 143.0
Yacin
ServerPackage
main ≡ ?1
40.922s
pwsh
70%
19:33:09
Yacin
ClientPackage
main ≡ ?1
4ms
jav
a .\Client.java
Client Connecté au serveur
donnez une operation pour envoyer au serveur
44 + 99
Result : 143.0
Yacin
ClientPackage
main ≡ ?1
21.995s
pwsh
70%
19:33:09

Deuxième Méthode :

Opération.java :

```java
package Methode2;

import java.io.Serializable;

public class Operation implements Serializable {
  private int val1;
  private int val2;
  private char opr;

  public Operation() {
  }

  public Operation(int val1, int val2, char opr) {
    this.val1 = val1;
    this.val2 = val2;
    this.opr = opr;
  }

  public int getVal1() {
    return val1;
  }

  public int getVal2() {
    return val2;
  }

  public char getOpr() {
    return opr;
  }

  public void setVal1(int val1) {
    this.val1 = val1;
  }

  public void setVal2(int val2) {
    this.val2 = val2;
  }

  public void setOpr(char opr) {
    this.opr = opr;
  }

  public static void main(String[] args) {
    System.out.println("Testing");
  }
}

```

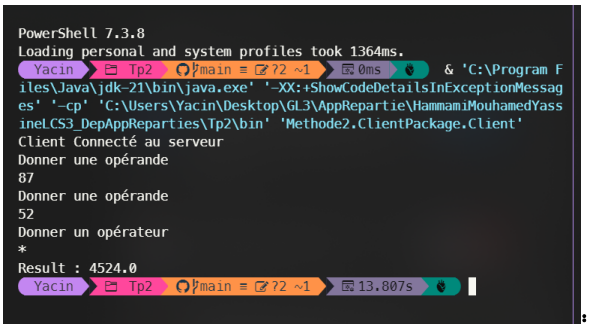
Server.java :

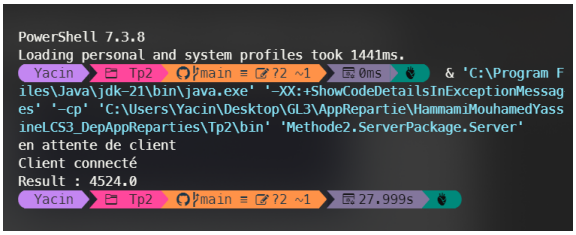
```java
package Methode2;

import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.ObjectInputStream;
import java.io.OutputStream;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;

public class Server {
  public static void main(String[] args) throws Exception {
    ServerSocket ss = new ServerSocket(2000);
    Socket s = ss.accept();
    System.out.println("Client connecté");
    // lecture et ecriture seulement avec un seul octet
    InputStream is = s.getInputStream();
    OutputStream os = s.getOutputStream();
    // lecture et ecriture seulement avec plus un seul octet
    InputStreamReader isr = new InputStreamReader(is);
    BufferedReader bfrIn = new BufferedReader(isr);
    PrintWriter pw = new PrintWriter(os, true);
    // lecture des objets
    ObjectInputStream ois = new ObjectInputStream(is);
    Operation myOpr = (Operation) ois.readObject();
    int val1 = myOpr.getVal1();
    int val2 = myOpr.getVal2();
    char opr = myOpr.getOpr();
    // initialization result
    float value = 0;
    // la logique des calcul
    switch (opr) {
      case '+':
        value = val1 + val2;
        break;
      case '-':
        value = val1 - val2;
        break;
      case '/':
        value = val1 / val2;
        break;
      case '*':
        value = val1 * val2;
        break;
      default:
        System.out.println("Opération ivalide");
        break;
    }
    // envoi des résultas valide
    System.out.println("Result : " + value);
    pw.println("Result : " + value);
    ss.close();
  }
}
```

Client.java :

```java
package Methode2;

import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.ObjectOutputStream;
import java.io.OutputStream;
import java.io.PrintWriter;
import java.net.Socket;

public class Client {
  public static void main(String[] args) throws Exception {
    Socket s = new Socket("localhost", 2000);
    System.out.println("Client Connecté au serveur");
    // lecture et ecriture seulement avec un seul octet
    InputStream is = s.getInputStream();
    OutputStream os = s.getOutputStream();

// lecture et ecriture seulement avec plus un seul octet
    InputStreamReader isr = new InputStreamReader(is);
    BufferedReader bfrIn = new BufferedReader(isr);
    PrintWriter pw = new PrintWriter(os, true);

    // lecture des objects
    ObjectOutputStream oos = new ObjectOutputStream(os);

    // Creation d'un instance de classe opération
    Operation myOp = new Operation(3, 4, '*');
    // envoi de l'objet serializé
    oos.writeObject(myOp);
    // affichage de resultat
    System.out.println(bfrIn.readLine());

    s.close();
  }
}

```

Résultat


PowerShell 7.3.8
Loading personal and system profiles took 1364ms.
Yacin Tp2 main ≡ ?2 ~1 0ms & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Yacin\Desktop\GL3\AppRepartie\HammamiMouhamedYassineLCS3_DepAppReparties\Tp2\bin' 'Methode2.ClientPackage.Client'
Client Connecté au serveur
Donner une opérande
87
Donner une opérande
52
Donner un opérateur
*
Result : 4524.0
Yacin Tp2 main ≡ ?2 ~1 13.807s


PowerShell 7.3.8
Loading personal and system profiles took 1441ms.
Yacin Tp2 main ≡ ?2 ~1 0ms & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Yacin\Desktop\GL3\AppRepartie\HammamiMouhamedYassineLCS3_DepAppReparties\Tp2\bin' 'Methode2.ServerPackage.Server'
en attente de client
Client connecté
Result : 4524.0
Yacin Tp2 main ≡ ?2 ~1 27.999s