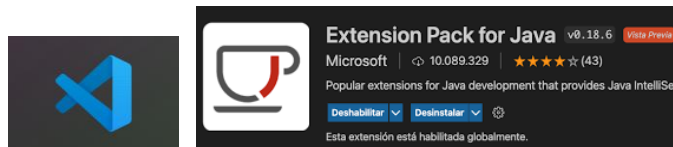


Compte Rendu

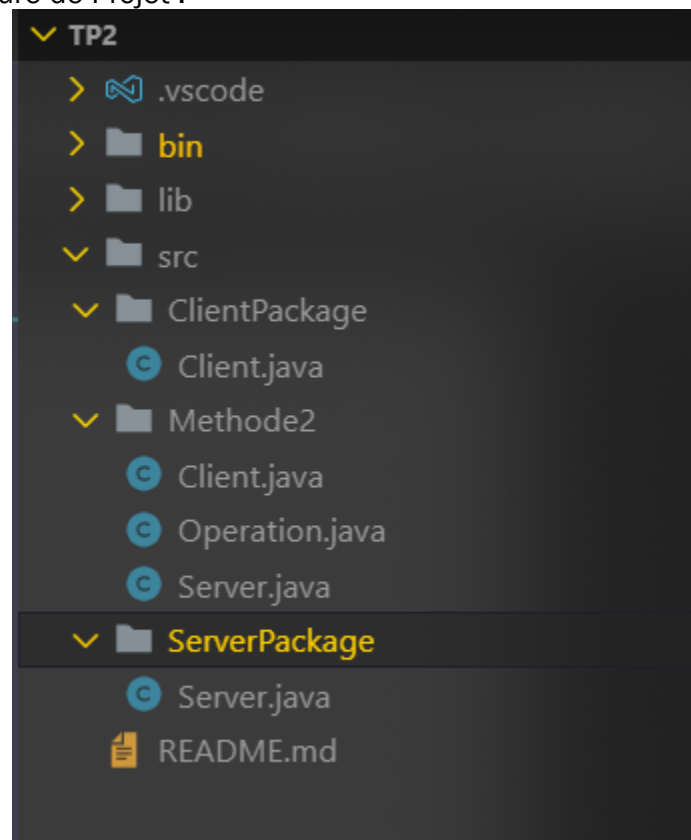
Tp1 : Développement d'applications réparties
Hammami Mouhamed Yassine
GLSI3-1C-gp2

Environnement de travail :

- L'éditeur de texte VSCode + Java Extension pack :



- Structure de Projet :



- Serveur.java

```
1 package ServerPackage;
2
3 import java.io.BufferedReader;
4 import java.io.InputStream;
5 import java.io.InputStreamReader;
6 import java.io.OutputStream;
7 import java.io.PrintWriter;
8 import java.net.ServerSocket;
9 import java.net.Socket;
10
11 public class Server {
12     public static void main(String[] args) throws Exception {
13         ServerSocket ss = new ServerSocket(2000);
14         Socket s = ss.accept();
15         System.out.println("Client connecté");
16         // lecture et écriture seulement avec un seul octet
17         InputStream is = s.getInputStream();
18         OutputStream os = s.getOutputStream();
19         // lecture et écriture seulement avec plus un seul octet
20         InputStreamReader isr = new InputStreamReader(is);
21         BufferedReader bfrIn = new BufferedReader(isr);
22         PrintWriter pw = new PrintWriter(os, true);
23         // initialisation result
24         float value = 0;
25         String expression = bfrIn.readLine();
26         try {
27
28             // ==> On assume que l'opération est de la forme suivante : 3+4/3-4/3/4/3*4 alors
29             // si un opérateur existe dans l'expression alors son index est le maximum
30             // parmi les autres opérateurs
31             int pos = Math.max(expression.indexOf('+'),
32                               Math.max(expression.indexOf('-'), Math.max(expression.indexOf('/'),
33                                     expression.indexOf('*'))));
34             System.out.println("pos=" + pos);
35             char opr = expression.charAt(pos);
36             System.out.println("opr=" + opr);
37             // extraction des opérandes
38             float opr1 = Float.parseFloat(expression.substring(0, pos));
39             float opr2 = Float.parseFloat(expression.substring(pos + 1));
40             // la logique des calculs
41             switch (opr) {
42                 case '+':
43                     value = opr1 + opr2;
44                     break;
45                 case '-':
46                     value = opr1 - opr2;
47                     break;
48                 case '/':
49                     value = opr1 / opr2;
50                     break;
51                 case '*':
52                     value = opr1 * opr2;
53                     break;
54                 default:
55                     System.out.println("Opération invalide");
56                     break;
57             }
58             // gestion des exceptions
59         } catch (Exception e) {
60             pw.println("Opération invalide");
61             System.out.println(e.toString());
62             System.exit(1);
63         }
64         // envoi des résultats valides
65         System.out.println("Result : " + value);
66         pw.println("Result : " + value);
67
68         ss.close();
69     }
70 }
71
```

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 calcul

L'opérateur :

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 chaîne qui précèdent et suivent l'opérateur

Notez qu'il faut les transformer en entier

- Client.java

```
1 package ClientPackage;
2
3 import java.io.BufferedReader;
4 import java.io.InputStream;
5 import java.io.InputStreamReader;
6 import java.io.OutputStream;
7 import java.io.PrintWriter;
8 import java.net.Socket;
9
10 public class Client {
11     public static void main(String[] args) throws Exception {
12         Socket s = new Socket("localhost", 2000);
13         System.out.println("Client Connecté au serveur");
14         // lecture et ecriture seulement avec un seul octet
15         InputStream is = s.getInputStream();
16         OutputStream os = s.getOutputStream();
17         // lecture et ecriture seulement avec plus un seul octet
18         InputStreamReader isr = new InputStreamReader(is);
19         BufferedReader bfrIn = new BufferedReader(isr);
20         PrintWriter pw = new PrintWriter(os, true);
21
22         // lecture des données utilisateur de taille supérieure qu'un octet
23         InputStreamReader isrSys = new InputStreamReader(System.in);
24         BufferedReader bfrInSys = new BufferedReader(isrSys);
25
26         // Demander d'utilisateur d'ecrire
27         System.out.println("donnez un nombre pour envoyer au serveur");
28         String val = bfrInSys.readLine();
29         // envoi d'expression
30         pw.println(val);
31         // lire le résultat
32         String res = bfrIn.readLine();
33         System.out.println(res);
34
35         s.close();
36     }
37 }
38
```

Résultat:

```
TERMINAL
pwhsh - ClientPackage  + ~  🗑  📄
Yacin  ➤ ServerPackage  🔄main = 71  ⏱️3ms  🟢 jav
a .\Server.java
Client connecté
pos=3
opr=+
Result : 143.0
Yacin  ➤ ServerPackage  🔄main = 71  ⏱️40,922s  🟢
pwsh  ⚡70%  ⌚19:33:09

Yacin  ➤ ClientPackage  🔄main = 71  ⏱️4ms  🟢 jav
a .\Client.java
Client Connécté au serveur
donnez une operation pour envoyer au serveur
44 + 99
Result : 143.0
Yacin  ➤ ClientPackage  🔄main = 71  ⏱️21,995s  🟢
pwsh  ⚡70%  ⌚19:33:09
```

Deuxième Méthode :

Opération.java :

```
1 package Methode2;
2
3 import java.io.Serializable;
4
5 public class Operation implements Serializable {
6     private int val1;
7     private int val2;
8     private char opr;
9
10    public Operation() {
11    }
12
13    public Operation(int val1, int val2, char opr) {
14        this.val1 = val1;
15        this.val2 = val2;
16        this.opr = opr;
17    }
18
19    public int getVal1() {
20        return val1;
21    }
22
23    public int getVal2() {
24        return val2;
25    }
26
27    public char getOpr() {
28        return opr;
29    }
30
31    public void setVal1(int val1) {
32        this.val1 = val1;
33    }
34
35    public void setVal2(int val2) {
36        this.val2 = val2;
37    }
38
39    public void setOpr(char opr) {
40        this.opr = opr;
41    }
42
43    public static void main(String[] args) {
44        System.out.println("Testing");
45    }
46 }
47
```

Server.java :

```
1 package Methode2;
2
3 import java.io.BufferedReader;
4 import java.io.InputStream;
5 import java.io.InputStreamReader;
6 import java.io.ObjectInputStream;
7 import java.io.OutputStream;
8 import java.io.PrintWriter;
9 import java.net.ServerSocket;
10 import java.net.Socket;
11
12 public class Server {
13     public static void main(String[] args) throws Exception {
14         ServerSocket ss = new ServerSocket(2000);
15         Socket s = ss.accept();
16         System.out.println("Client connecté");
17         // lecture et ecriture seulement avec un seul octet
18         InputStream is = s.getInputStream();
19         OutputStream os = s.getOutputStream();
20         // lecture et ecriture seulement avec plus un seul octet
21         InputStreamReader isr = new InputStreamReader(is);
22         BufferedReader bfrIn = new BufferedReader(isr);
23         PrintWriter pw = new PrintWriter(os, true);
24         // lecture des objets
25         ObjectInputStream ois = new ObjectInputStream(is);
26         Operation myOpr = (Operation) ois.readObject();
27         int val1 = myOpr.getVal1();
28         int val2 = myOpr.getVal2();
29         char opr = myOpr.getOpr();
30         // initialization result
31         float value = 0;
32         // la logique des calcul
33         switch (opr) {
34             case '+':
35                 value = val1 + val2;
36                 break;
37             case '-':
38                 value = val1 - val2;
39                 break;
40             case '/':
41                 value = val1 / val2;
42                 break;
43             case '*':
44                 value = val1 * val2;
45                 break;
46             default:
47                 System.out.println("Opération invalide");
48                 break;
49         }
50         // envoi des résultats valide
51         System.out.println("Result : " + value);
52         pw.println("Result : " + value);
53         ss.close();
54     }
55 }
56
```

Client.java :

```
1 package Methode2;
2
3 import java.io.BufferedReader;
4 import java.io.InputStream;
5 import java.io.InputStreamReader;
6 import java.io.ObjectOutputStream;
7 import java.io.OutputStream;
8 import java.io.PrintWriter;
9 import java.net.Socket;
10
11 public class Client {
12     public static void main(String[] args) throws Exception {
13         Socket s = new Socket("localhost", 2000);
14         System.out.println("Client Connecté au serveur");
15         // lecture et ecriture seulement avec un seul octet
16         InputStream is = s.getInputStream();
17         OutputStream os = s.getOutputStream();
18
19         // lecture et ecriture seulement avec plus un seul octet
20         InputStreamReader isr = new InputStreamReader(is);
21         BufferedReader bfrIn = new BufferedReader(isr);
22         PrintWriter pw = new PrintWriter(os, true);
23
24         // lecture des objects
25         ObjectOutputStream oos = new ObjectOutputStream(os);
26
27         // Creation d'un instance de classe opération
28         Operation myOp = new Operation(3, 4, '*');
29         // envoi de l'objet serialisé
30         oos.writeObject(myOp);
31         // affichage de resultat
32         System.out.println(bfrIn.readLine());
33
34         s.close();
35     }
36 }
```

Résultat

```
PowerShell 7.3.8
Loading personal and system profiles took 1364ms.
Yacin > .\Tp2 .\main.exe 72 ~1 0ms & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Yacin\Desktop\GL3\AppRepartie\Hamman1MouhamedYassine\CS3_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.exe 72 ~1 13.807s
```

```
PowerShell 7.3.8
Loading personal and system profiles took 1441ms.
Yacin > .\Tp2 .\main.exe 72 ~1 0ms & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Yacin\Desktop\GL3\AppRepartie\Hamman1MouhamedYassine\CS3_DepAppReparties\Tp2\bin' 'Methode2.ServerPackage.Server'
en attente de client
Client connecté
Result : 4524.0
Yacin > .\Tp2 .\main.exe 72 ~1 27.999s
```