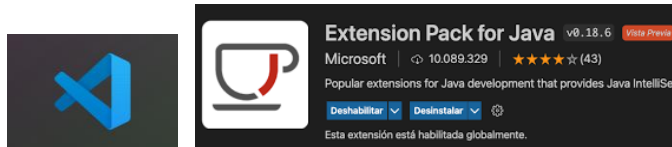


Compte Rendu

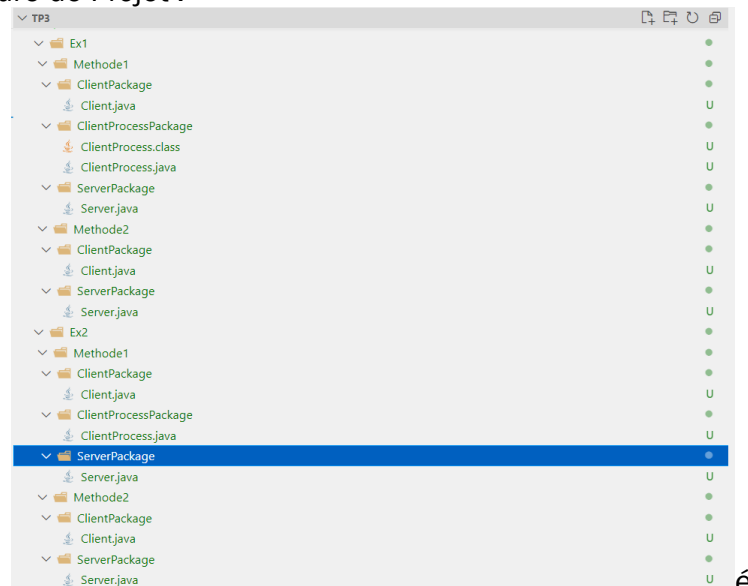
Tp1 : Développement d'applications réparties
Hammami Mouhamed Yassine
GLSI3-1C-gp2

Environnement de travail :

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



- Structure de Projet :



- Ex1

```
package Ex1.Methode1.ServerPackage;
```

```

import java.net.ServerSocket;
import java.net.Socket;

import Ex1.Methode1.ClientProcessPackage.ClientProcess;

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

        try (ServerSocket serverSocket = new ServerSocket(2023)) {
            System.out.println("En attente de client...");

            while (true) {
                // acceptation des processus client
                Socket serviceSocket = serverSocket.accept();
                System.out.println("Un nouveau client est connecté ");
                ClientProcess clp = new ClientProcess(serviceSocket);
                // lancement du processus client
                clp.start();
            }

        }

    }
}

```

→ Dans la boucle while on accepte et on lance les processus qui implémentent l'interface Runnable avec la méthode start() .

- Client.java

```
1 package Ex1.Methode1.ClientPackage;
2
3 import java.io.BufferedReader;
4 import java.io.IOException;
5 import java.io.InputStream;
6 import java.io.InputStreamReader;
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 IOException {
13         for (int i = 0; i < 10; i++) {
14             Socket socketClient = new Socket("localhost", 2023);
15             InputStream cltInput = socketClient.getInputStream();
16             InputStreamReader clientIsr = new InputStreamReader(cltInput);
17             BufferedReader clientBfrIn = new BufferedReader(clientIsr);
18             // Pour envoyer des flux de données au serveur
19             OutputStream cltOutput = socketClient.getOutputStream();
20             //
21             PrintWriter pw = new PrintWriter(cltOutput, true);
22             //
23             pw.println("Eco pour Client " + (i + 1));
24             System.out.println("client " + (i + 1) + " a envoye " + (9999 + i));
25             System.out.println("response server pour client " + (i + 1) + "= " +
                clientBfrIn.readLine());
26             socketClient.close();
27         }
28     }
29 }
```

→ Lancement 10 processus pour tester le multithreading.

- ClientProcess.java

```
package Ex1.Methode1.ClientPackage;

import java.io.BufferedReader;
import java.io.IOException;
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 IOException {
        for (int i = 0; i < 10; i++) {
            Socket socketClient = new Socket("localhost", 2023);
            InputStream cltInput = socketClient.getInputStream();
            InputStreamReader clientIsr = new
InputStreamReader(cltInput);
            BufferedReader clientBfrIn = new BufferedReader(clientIsr);
            // Pour envoyer des flux de données au serveur
            OutputStream cltOutput = socketClient.getOutputStream();
            //
            PrintWriter pw = new PrintWriter(cltOutput, true);
            //
            pw.println("Eco pour Client " + (i + 1));
            System.out.println("client " + (i + 1) + " a envoye " +
(9999 + i));
            System.out.println("response server pour client " + (i + 1)
+ "= " + clientBfrIn.readLine());
            socketClient.close();
        }
    }
}
```

Exécution :

Server

```
Yacin ▶ Tp3 ▶ main = 79 ~1 -5 ▶ 0ms & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Yacin\Desktop\GL3\AppRepartie\HammamiMouhamedYassineLCS3_DepAppReparties\Tp3\Tp3\bin' 'Ex1.Methode1.ServerPackage.Server'
En attente de client...
```

```
En attente de client...
Un nouveau client est connecté
Client Process en execution
req : Eco pour Client 1
client is done
Un nouveau client est connecté
Client Process en execution
req : Eco pour Client 2
client is done
Un nouveau client est connecté
Client Process en execution
req : Eco pour Client 3
client is done
Un nouveau client est connecté
Client Process en execution
req : Eco pour Client 4
client is done
Un nouveau client est connecté
Client Process en execution
req : Eco pour Client 5
client is done
Un nouveau client est connecté
Client Process en execution
req : Eco pour Client 6
client is done
Un nouveau client est connecté
Client Process en execution
req : Eco pour Client 7
client is done
Un nouveau client est connecté
Client Process en execution
req : Eco pour Client 8
client is done
Un nouveau client est connecté
Client Process en execution
req : Eco pour Client 9
client is done
Un nouveau client est connecté
```

Client :

```
Yacin ▶ Tp3 ▶ main = 79 ~1 -5 ▶ 0ms & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Yacin\Desktop\GL3\AppRepartie\HammamiMouhamedYassineLCS3_DepAppReparties\Tp3\Tp3\bin' 'Ex1.Methode1.ClientPackage.Client'
client 1 a envoye 9999
response server pour client 1= Eco pour Client 1:Validé
client 2 a envoye 10000
response server pour client 2= Eco pour Client 2:Validé
client 3 a envoye 10001
response server pour client 3= Eco pour Client 3:Validé
client 4 a envoye 10002
response server pour client 4= Eco pour Client 4:Validé
client 5 a envoye 10003
response server pour client 5= Eco pour Client 5:Validé
client 6 a envoye 10004
response server pour client 6= Eco pour Client 6:Validé
client 7 a envoye 10005
response server pour client 7= Eco pour Client 7:Validé
client 8 a envoye 10006
response server pour client 8= Eco pour Client 8:Validé
client 9 a envoye 10007
response server pour client 9= Eco pour Client 9:Validé
client 10 a envoye 10008
response server pour client 10= Eco pour Client 10:Validé
```

Méthode 2 (seulement deux classes) :

NB : Il n'y a pas de changement dans la class Client.java

Server.java :

```
package Ex1.Methode2.ServerPackage;

import java.io.BufferedReader;
import java.io.IOException;
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 extends Thread {
    public void run() {

        try (ServerSocket serverSocket = new ServerSocket(2023)) {
            System.out.println("En attente de client...");

            while (true) {
                Socket serviceSocket = serverSocket.accept();
                System.out.println("Un nouveau client est connecté ");
                ClientProcess clp = new ClientProcess(serviceSocket);
                clp.start();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

class ClientProcess extends Thread {
    Socket socket;
```

```

public ClientProcess(Socket socket) {
    super();
    this.socket = socket;
}

@Override
public void run() {
    System.out.println("Client Process en execution");
    try {
        InputStream clpInput = this.socket.getInputStream();
        InputStreamReader reader = new InputStreamReader(clpInput);
        BufferedReader clpBfr = new BufferedReader(reader);
        // Pour envoyer des flux de données au serveur
        OutputStream clpOutput = this.socket.getOutputStream();
        PrintWriter clpWriter = new PrintWriter(clpOutput, true);

        // int input = clpInput.read();
        String req = clpBfr.readLine();
        System.out.println("req: " + req);
        clpWriter.println(req + ":Validé");
        System.out.println("client is done");
        // System.out.println("input: " + input);
        // System.out.println("Resutat envoyé");

        socket.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

// lancement du server qui implemente l'interface Runnable
public static void main(String[] args) {
    new Server().start();
}
}

```

Exécution :

Server :

```
Yacin ▶ Tp3 ▶ main = ? ?10 ~1 -5 ▶ 0ms & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Yacin\Desktop\GL3\AppRepartie\HamamiMouhamedYassine\CS3_DepAppReparties\Tp3\Tp3\bin' 'Ex1.Methode2.ServerPackage.ClientProcess'
En attente de client...
Un nouveau client est connecté
Client Process en execution
req : Eco pour Client 1
client is done
Un nouveau client est connecté
Client Process en execution
req : Eco pour Client 2
client is done
Un nouveau client est connecté
Client Process en execution
req : Eco pour Client 3
client is done
Un nouveau client est connecté
Client Process en execution
req : Eco pour Client 4
client is done
Un nouveau client est connecté
Client Process en execution
req : Eco pour Client 5
client is done
Un nouveau client est connecté
Client Process en execution
req : Eco pour Client 6
client is done
Un nouveau client est connecté
Client Process en execution
req : Eco pour Client 7
client is done
Un nouveau client est connecté
Client Process en execution
req : Eco pour Client 8
client is done
```

Client :

```
Yacin ▶ Tp3 ▶ main = ? ?10 ~1 -5 ▶ 0ms & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Yacin\Desktop\GL3\AppRepartie\HamamiMouhamedYassine\CS3_DepAppReparties\Tp3\Tp3\bin' 'Ex1.Methode2.ClientPackage.Client'
client 1 a envoye 9999
response server pour client 1= Eco pour Client 1:Validé
client 2 a envoye 10000
response server pour client 2= Eco pour Client 2:Validé
client 3 a envoye 10001
response server pour client 3= Eco pour Client 3:Validé
client 4 a envoye 10002
response server pour client 4= Eco pour Client 4:Validé
client 5 a envoye 10003
response server pour client 5= Eco pour Client 5:Validé
client 6 a envoye 10004
response server pour client 6= Eco pour Client 6:Validé
client 7 a envoye 10005
response server pour client 7= Eco pour Client 7:Validé
client 8 a envoye 10006
response server pour client 8= Eco pour Client 8:Validé
client 9 a envoye 10007
response server pour client 9= Eco pour Client 9:Validé
client 10 a envoye 10008
response server pour client 10= Eco pour Client 10:Validé
Yacin ▶ Tp3 ▶ main = ? ?10 ~1 -5 ▶ 419ms
```


Ex2 : Calculatrice :

Méthode 1 :

Server.java :

```
package Ex2.Methode1.ServerPackage;

import java.net.ServerSocket;
import java.net.Socket;

import Ex2.Methode1.ClientProcessPackage.ClientProcess;

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

        try (ServerSocket serverSocket = new ServerSocket(2023)) {
            System.out.println("En attente de client...");

            while (true) {
                Socket serviceSocket = serverSocket.accept();
                System.out.println("Un nouveau client est connecté ");
                ClientProcess clp = new ClientProcess(serviceSocket);
                clp.start();
            }
        }
    }
}
```

Client.java :

```
package Ex2.Methode1.ClientPackage;

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

public class Client {

    public static void main(String[] args) throws IOException {
        Scanner userInput = new Scanner(System.in);
        for (int i = 0; i < 10; i++) {
            Socket socketClient = new Socket("localhost", 2023);
            InputStream cltInput = socketClient.getInputStream();
            InputStreamReader clientIsr = new
InputStreamReader(cltInput);
            BufferedReader clientBfrIn = new BufferedReader(clientIsr);
            // Pour envoyer des flux de données au serveur
            OutputStream cltOutput = socketClient.getOutputStream();
            //
            PrintWriter pw = new PrintWriter(cltOutput, true);
            //
            // Demander d'utilisateur d'ecrire
            System.out.println("donnez une operation pour envoyer au
serveur");

            String val = userInput.nextLine();
            // envoi d'expression
            pw.println(val);
            // lire le résultat
            String res = clientBfrIn.readLine();
            System.out.println(res);
            socketClient.close();
        }
    }
}
```

```
    }  
    userInput.close();  
}  
}
```

ClientProcess.java :

```
package Ex2.Methode1.ClientProcessPackage;  
  
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 ClientProcess extends Thread {  
    Socket socket;  
  
    public ClientProcess(Socket socket) {  
        super();  
        this.socket = socket;  
    }  
  
    @Override  
    public void run() {  
        System.out.println("Client Process en execution");  
        try {  
            InputStream clpInput = this.socket.getInputStream();  
            InputStreamReader reader = new InputStreamReader(clpInput);  
            BufferedReader clpBfr = new BufferedReader(reader);  
            // Pour envoyer des flux de données au serveur  
            OutputStream clpOutput = this.socket.getOutputStream();  
            PrintWriter clpWriter = new PrintWriter(clpOutput, true);  
  
            float value = 0;  
            String expression = clpBfr.readLine();
```

```

try {
    // ==> On assume que l'opération est de la forme suivante
    // 3+4/3-4/3/4/3*4 alors
    // si un opérateur existe dans l'expression alors son
    // index est le maximum
    // parmi les autres 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 calculs
    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 invalide");
            System.exit(1);
            break;
    }
    // gestion des exceptions
}

```

```

    } catch (Exception e) {
        clpWriter.println("Opération invalide");
        System.out.println(e.toString());
        socket.close();
        Thread.currentThread().interrupt();
        // System.exit(1);
    }
    // envoi des résultats valide
    System.out.println("Result : " + value);
    clpWriter.println("Result : " + value);

    socket.close();
} catch (Exception e) {
    e.printStackTrace();
}
}
}
}

```

Exécution :

The left screenshot shows the server-side output in a command prompt. It displays the following text:

```

Yacine > java -cp 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Yacine\Desktop\GL3\AppRepartie\HamamiMouhamedYassine\CS3_DepAppReparties\Tp3\Tp3\bin' 'Ex2.Methode1.ServerPackage.Server'
En attente de client...
Un nouveau client est connecté
Client Process en execution
pos=2
opr=*
Result : 5500.0
Un nouveau client est connecté
Client Process en execution
pos=1
java.lang.StringIndexOutOfBoundsException: Index -1 out of bounds for length 6
Result : 0.0
Un nouveau client est connecté
Client Process en execution
pos=5
opr=+
java.lang.NumberFormatException: For input string: "97*99"
Result : 0.0
Un nouveau client est connecté
Client Process en execution
pos=2
opr=/
Result : 77.0

```

The right screenshot shows the client-side output in a command prompt. It displays the following text:

```

Yacine > java -cp 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Yacine\Desktop\GL3\AppRepartie\HamamiMouhamedYassine\CS3_DepAppReparties\Tp3\Tp3\bin' 'Ex2.Methode1.ClientPackage.Client'
donnez une operation pour envoyer au serveur
55*100
Result : 5500.0
donnez une operation pour envoyer au serveur
berbet
Opération invalide
donnez une operation pour envoyer au serveur
97*99+
Opération invalide
donnez une operation pour envoyer au serveur
77/1
Result : 77.0

```

Remarque :

La gestion des exception des opération invalide se fait avec la méthode `Thread.interrupt()` non par `System.exit()` .

```

    } catch (Exception e) {
        clpWriter.println(x:"Opération ivalide");
        System.out.println(e.toString());
        socket.close();
        Thread.currentThread().interrupt();
        // System.exit(1);
    }

```

Méthode 2 :

Client.java

```

package Ex2.Methode2.ClientPackage;

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

public class Client {

    public static void main(String[] args) throws IOException {
        Scanner userInput = new Scanner(System.in);
        for (int i = 0; i < 10; i++) {
            Socket socketClient = new Socket("localhost", 2023);
            InputStream cltInput = socketClient.getInputStream();
            InputStreamReader clientIsr = new
InputStreamReader(cltInput);
            BufferedReader clientBfrIn = new BufferedReader(clientIsr);
            // Pour envoyer des flux de données au serveur
            OutputStream cltOutput = socketClient.getOutputStream();
            //
            PrintWriter pw = new PrintWriter(cltOutput, true);
            //
            // Demander d'utilisateur d'ecrire
            System.out.println("donnez une operation pour envoyer au
serveur");

            String val = userInput.nextLine();

```

```

        // envoi d'expression
        pw.println(val);
        // lire le résultat
        String res = clientBfrIn.readLine();
        System.out.println(res);
        socketClient.close();
    }
    userInput.close();
}
}

```

Server.java

```

package Ex2.Methode2.ServerPackage;

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

public class Server extends Thread {
    public void run() {
        try (ServerSocket serverSocket = new ServerSocket(2023)) {
            System.out.println("En attente de client...");

            while (true) {
                Socket serviceSocket = serverSocket.accept();
                System.out.println("Un nouveau client est connecté ");
                ClientProcess clp = new ClientProcess(serviceSocket);
                clp.start();
            }

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

```

```

    }

}

public class ClientProcess extends Thread {
    Socket socket;

    public ClientProcess(Socket socket) {
        super();
        this.socket = socket;
    }

    @Override
    public void run() {
        System.out.println("Client Process en execution");
        try {
            InputStream clpInput = this.socket.getInputStream();
            InputStreamReader reader = new
InputStreamReader(clpInput);
            BufferedReader clpBfr = new BufferedReader(reader);
            // Pour envoyer des flux de données au serveur
            OutputStream clpOutput = this.socket.getOutputStream();
            PrintWriter clpWriter = new PrintWriter(clpOutput, true);

            float value = 0;
            String expression = clpBfr.readLine();
            try {
                // ==> On assume que l'opération est de la forme suivante
:3+4/3-4/3/4/3*4 alors
                // si un opérateur existe dans l'expression alors son
index est le maximum
                // parmi les autres 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 invalide");
                System.exit(1);
                break;
        }
        // gestion des exceptions
    } catch (Exception e) {
        clpWriter.println("Opération invalide");
        System.out.println(e.toString());
        socket.close();
        Thread.currentThread().interrupt();
        // System.exit(1);
    }
    // envoi des résultats valide
    System.out.println("Result : " + value);
    clpWriter.println("Result : " + value);

    socket.close();
} catch (Exception e) {

```

```

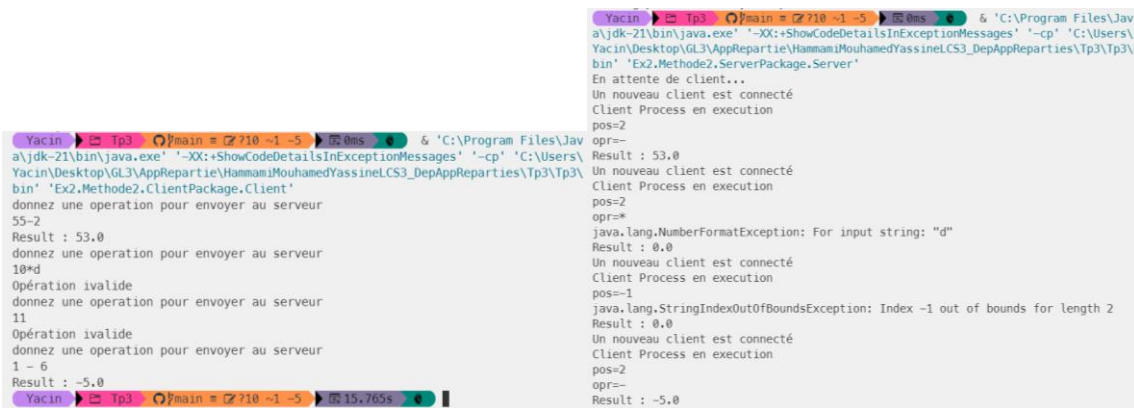
        e.printStackTrace();
    }
}

}

public static void main(String[] args) {
    new Server().start();
}
}

```

Exécution :



The screenshot displays the execution of a Java application in an IDE. The left pane shows the client's input and output, and the right pane shows the server's logs.

Client Output (Left Pane):

```

Yacin> Tp3
main = 710 ~1 ~5
& 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Yacin\Desktop\GL3AppRepartie\Hamam\MouhamedYassine\CS3_DepAppReparties\Tp3\Tp3\bin' 'Ex2.Methode2.ClientPackage.Client'
donnez une operation pour envoyer au serveur
55-2
Result : 53.0
donnez une operation pour envoyer au serveur
10*d
Opération invalide
donnez une operation pour envoyer au serveur
11
Opération invalide
donnez une operation pour envoyer au serveur
1 - 6
Result : -5.0
Yacin> Tp3
main = 710 ~1 ~5
15.765s

```

Server Output (Right Pane):

```

Yacin> Tp3
main = 710 ~1 ~5
& 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Yacin\Desktop\GL3AppRepartie\Hamam\MouhamedYassine\CS3_DepAppReparties\Tp3\Tp3\bin' 'Ex2.Methode2.ServerPackage.Server'
En attente de client...
Un nouveau client est connecté
Client Process en execution
pos=2
opr=-
Result : 53.0
Un nouveau client est connecté
Client Process en execution
pos=2
opr=*
java.lang.NumberFormatException: For input string: "d"
Result : 0.0
Un nouveau client est connecté
Client Process en execution
pos=-1
java.lang.StringIndexOutOfBoundsException: Index -1 out of bounds for length 2
Result : 0.0
Un nouveau client est connecté
Client Process en execution
pos=2
opr=-
Result : -5.0

```