Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

**МЕХАНІКО-ТЕХНОЛОГІЧНИЙ ФАКУЛЬТЕТ**

Кафедра кібербезпеки та програмного забезпечення

**Звіт**

з виконаної лабораторної роботи № 5

Дисципліна: Паралельні та розподілені обчислення

на тему

«Розподілені обчислення на базі технології Клієнт-Сервер»

Виконав :

студент академічної групи КІ-15

Аннаєв А. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Перевірив :

Викладач

Минайленко Р. М.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кропивницкий- 2018

**Мета:** Одержати навички створення і налагодження додатків для розподілених обчислень.

**Завдання:**

- Створити проект для Клієнтської і Серверної програми.

- Реалізувати програмне забезпечення Сервера.

- Реалізувати програмне забезпечення Клієнта.

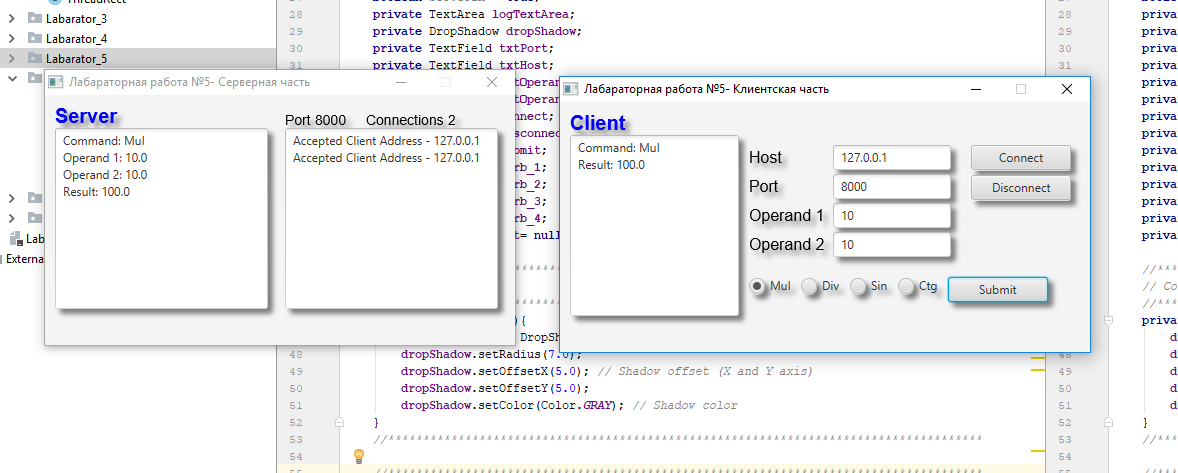
- Здійснити мережну взаємодію Клієнта і Сервера (установити з'єднання).

- На стороні Сервера забезпечити виконання обчислень за даними, отриманими від   
 Клієнта.

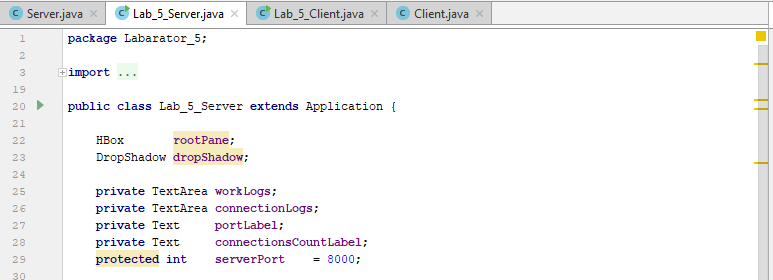
- Результати обчислень повернути Клієнту.

- На стороні клієнта забезпечити виведення результатів обчислень.

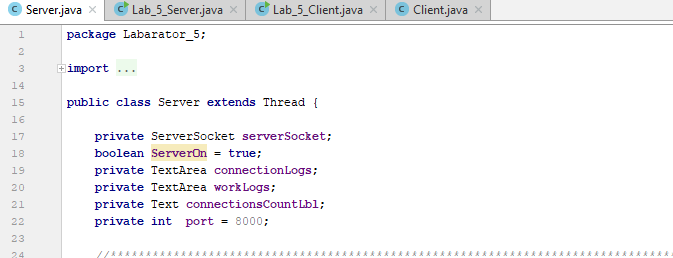
Хід роботи



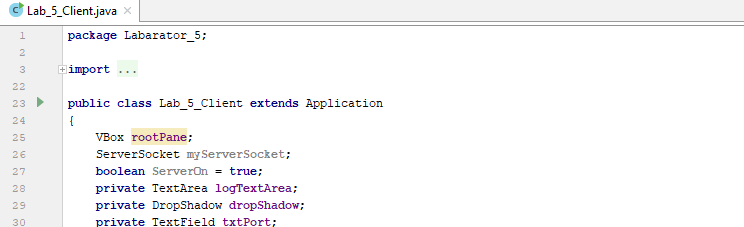
**package** Labarator\_5;  
  
**public class** Lab\_5\_Server **extends** Application {  
  
 HBox **rootPane**;  
 DropShadow **dropShadow**;  
  
 **private** TextArea **workLogs**;  
 **private** TextArea **connectionLogs**;  
 **private** Text **portLabel**;  
 **private** Text **connectionsCountLabel**;  
 **protected int serverPort** = 8000;  
 *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
// Создание тени  
//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\** **private void** shadow(){  
 **dropShadow** = **new** DropShadow();  
 **dropShadow**.setRadius(7.0);  
 **dropShadow**.setOffsetX(5.0); *// Shadow offset (X and Y axis)* **dropShadow**.setOffsetY(5.0);  
 **dropShadow**.setColor(Color.***GRAY***); *// Shadow color* }  
*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
// Определение отображения  
//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\** **private void** createLayout(){  
 **rootPane** = **new** HBox();  
  
 **workLogs** = **new** TextArea();  
 **connectionLogs** = **new** TextArea();  
 **portLabel** = createLabel(**""**, Color.***BLACK***, FontWeight.***NORMAL***, 14);  
 **connectionsCountLabel** = createLabel(**""**, Color.***BLACK***, FontWeight.***NORMAL***, 14);  
  
 **workLogs**.setWrapText(**true**);  
 **connectionLogs**.setWrapText(**true**);  
  
 VBox leftLayout = **new** VBox();  
 VBox rightLayout = **new** VBox();  
 HBox conTitlesLayout = **new** HBox();  
  
 Text serverTitle = createLabel(**"Server"**, Color.***BLUE***, FontWeight.***BOLD***, 20);  
 Text portTitle = createLabel(**"Port "**, Color.***BLACK***, FontWeight.***NORMAL***, 14);  
 Text connectionsTitle = createLabel(**"Connections "**, Color.***BLACK***, FontWeight.***NORMAL***, 14);  
 conTitlesLayout.getChildren().addAll(portTitle, **portLabel**, connectionsTitle, **connectionsCountLabel**);  
 leftLayout.getChildren().addAll(serverTitle, **workLogs**);  
 rightLayout.getChildren().addAll(conTitlesLayout, **connectionLogs**);  
  
 **rootPane**.setSpacing(10);  
 rightLayout.setPadding(**new** Insets(7, 7, 7, 7));  
 **rootPane**.setPadding(**new** Insets(10, 10, 10, 10));  
 **rootPane**.getChildren().addAll(leftLayout, rightLayout);  
 shadow();  
 leftLayout.setEffect(**dropShadow**);  
 rightLayout.setEffect(**dropShadow**);  
 }  
 *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
// Определение нового потока с параметрами  
//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\** **private** Text createLabel(String text, Color color, FontWeight fontWeight, **int** fontSize){  
 Text t = **new** Text();  
 t.setText(text);  
 t.setFont(Font.*font*(**"Arial"**, fontWeight, fontSize));  
 t.setFill(color);  
  
 **return** t;  
 }  
*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\** @Override  
 **public void** start(Stage primaryStage) **throws** Exception {  
  
 createLayout();  
  
 primaryStage.setTitle(**"Лабараторная работа №5- Серверная часть"**);  
 primaryStage.setResizable(**false**);  
  
 Scene scene = **new** Scene(**rootPane**, 460, 240, Color.***TRANSPARENT***);  
 primaryStage.setScene(scene);  
 primaryStage.show();  
  
 **connectionsCountLabel**.setText(**"0"**);  
 **portLabel**.setText(String.*valueOf*(**serverPort**) + **" "**);  
  
  
 Server server = **new** Server(**workLogs**, **connectionLogs**, **connectionsCountLabel**, **serverPort**);  
 server.start();  
  
 primaryStage.setOnCloseRequest(**new** EventHandler<WindowEvent>() {  
 **public void** handle(WindowEvent we) {  
 server.stopServer();  
 }  
 });  
 }  
  
 **public static void** main(String[] args) {  
 *launch*(args);  
 }  
}



**package** Labarator\_5;  
  
**public class** Server **extends** Thread {  
  
 **private** ServerSocket **serverSocket**;  
 **boolean ServerOn** = **true**;  
 **private** TextArea **connectionLogs**;  
 **private** TextArea **workLogs**;  
 **private** Text **connectionsCountLbl**;  
 **private int port** = 8000;  
 *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
// Конструктор  
//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\****public** Server(TextArea workLogs, TextArea connectionLogs, Text connectionsLabel, **int** port)  
 {  
 **this**.**connectionLogs** = connectionLogs;  
 **this**.**workLogs** = workLogs;  
 **this**.**connectionsCountLbl** = connectionsLabel;  
 **this**.**port** = port;  
 }  
*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  
 *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
// Запуск сервера (пользовательский метод)  
//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\** **public void** run(){  
 startServer();  
 } *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
// Запуск сервера  
//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\** **private void** startServer() {  
 **try** {  
 **serverSocket** = **new** ServerSocket(**port**);  
 } **catch** (IOException ioe)  
 {  
 printWorkLog(**"Could not create server socket on port "** + String.*valueOf*(**port**) + **". Quitting."**);  
 System.*exit*(-1);  
 }  
  
 *// Successfully created Server Socket. Now wait for connections.* **while** (**ServerOn**) {  
 **try** {  
 *// Accept incoming connections.* Socket clientSocket = **serverSocket**.accept();  
  
 **int** n = Integer.*valueOf*(**connectionsCountLbl**.getText());  
 n++;  
 **connectionsCountLbl**.setText(String.*valueOf*(n));  
  
 *// Start a Service thread* ClientServiceThread cliThread = **new** ClientServiceThread(clientSocket);  
 cliThread.start();  
  
 } **catch** (IOException ioe) {  
 printWorkLog(**"Exception encountered on accept. Ignoring. Stack Trace :"**);  
 ioe.printStackTrace();  
 }  
 }  
 **try** {  
 **serverSocket**.close();  
 printWorkLog(**"Server Stopped"**);  
 } **catch** (Exception ioe) {  
 printWorkLog(**"Problem stopping server socket"**);  
 System.*exit*(-1);  
 }  
 } *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
// Остановка сервера  
//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\** **public void** stopServer(){  
 **ServerOn** = **false**;  
 **try** {  
 **serverSocket**.close();  
 printWorkLog(**"Server Stopped"**);  
 } **catch** (IOException ioe) {  
 printWorkLog(**"Problem stopping server socket"**);  
 System.*exit*(-1);  
 }  
 } *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
// Распечатка логов  
//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\** **private void** printWorkLog(String str){  
 **if**(**workLogs** != **null**)  
 **workLogs**.appendText(str + **"\n"**);  
 }  
*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
// Распечатка логов  
//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\** **private void** printConnectionLog(String str){  
 **if**(**connectionLogs** != **null**)  
 **connectionLogs**.appendText(str + **"\n"**);  
 }  
*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
// Обработка запросов клиента  
//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\** **public class** ClientServiceThread **extends** Thread{  
 Socket **clientSocket**;  
 **boolean m\_bRunThread** = **true**;  
  
 **public** ClientServiceThread() {  
 **super**();  
 }  
 ClientServiceThread(Socket s) {  
 **clientSocket** = s;  
 }  
 **public void** run() {  
BufferedReader in = **null**;  
 PrintWriter out = **null**;  
printConnectionLog(**"Accepted Client Address - "** + **clientSocket**.getInetAddress().getHostName());  
  
 **try** {  
 in = **new** BufferedReader(**new** InputStreamReader(**clientSocket**.getInputStream()));  
out = **new** PrintWriter(**new** OutputStreamWriter(**clientSocket**.getOutputStream()));  
**while**(**m\_bRunThread**) {  
 **if**(!**ServerOn**) {  
 out.flush();  
 **m\_bRunThread** = **false**;  
 }  
 String clientCommand = in.readLine();  
 **if**(clientCommand == **null**)  
 **continue**;  
 printWorkLog(**"Command: "** + clientCommand);  
 **double** res = 0.0f;  
 **if**(clientCommand != **null**)  
 {  
 **if** (clientCommand.contentEquals(**"Mul"**))  
 {  
 String op1 = in.readLine();  
 String op2 = in.readLine();  
 printWorkLog(**"Operand 1: "** + op1);  
 printWorkLog(**"Operand 2: "** + op2);  
 res = Double.*valueOf*(op1) \* Double.*valueOf*(op2);  
 out.println(String.*valueOf*(res));  
 printWorkLog(**"Result: "** + String.*valueOf*(res) + **"\n"**);  
 }  
 **else  
 if** (clientCommand.contentEquals(**"Div"**)) {  
 String op1 = in.readLine();  
 String op2 = in.readLine();  
 printWorkLog(**"Operand 1: "** + op1);  
 printWorkLog(**"Operand 2: "** + op2);  
 res = Double.*valueOf*(op1) / Double.*valueOf*(op2);  
 out.println(String.*valueOf*(res));  
 printWorkLog(**"Result: "** + String.*valueOf*(res) + **"\n"**);  
 }  
 **else  
 if** (clientCommand.contentEquals(**"Sin"**)) {  
 String op1 = in.readLine();  
 printWorkLog(**"Operand 1: "** + op1);  
 res = *sin*(Double.*valueOf*(op1));  
 out.println(String.*valueOf*(res));  
 printWorkLog(**"Result: "** + String.*valueOf*(res) + **"\n"**);  
 }  
 **else  
 if** (clientCommand.contentEquals(**"Ctg"**)) {  
 String op1 = in.readLine();  
 printWorkLog(**"Operand 1: "** + op1);  
 **double** z = *sin*(Double.*valueOf*(op1));  
 **if** (z != 0) {  
 res = *cos*(Double.*valueOf*(op1)) / z;  
 out.println(String.*valueOf*(res));  
 printWorkLog(**"Result: "** + String.*valueOf*(res) + **"\n"**);  
 }  
 **else** {  
 out.println(**"Error: zero! division"**);  
 printWorkLog(**"Error: zero! division\n"**);  
 }  
 }  
 **else  
 if** (clientCommand.contentEquals(**"quit"**))  
 {  
 printConnectionLog(**"Close connection - "** +  
 **clientSocket**.getInetAddress().getHostName());  
 **m\_bRunThread** = **false**;  
 **int** n = Integer.*valueOf*(**connectionsCountLbl**.getText());  
 n--;  
 **connectionsCountLbl**.setText(String.*valueOf*(n));  
 }  
 out.flush();  
 }  
 }  
 }  
 **catch**(Exception e)  
 {  
 e.printStackTrace();  
 }  
 **finally** {  
 *// Clean up* **try** {  
 in.close();  
 out.close();  
 **clientSocket**.close();  
 printConnectionLog(**"...Stopped"**);  
 }  
 **catch**(IOException ioe)  
 {  
 ioe.printStackTrace();  
 }  
 }  
 }  
 }  
 *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**}



**package** Labarator\_5;  
  
**public class** Lab\_5\_Client **extends** Application  
{  
 VBox **rootPane**;  
 ServerSocket **myServerSocket**;  
 **boolean ServerOn** = **true**;  
 **private** TextArea **logTextArea**;  
 **private** DropShadow **dropShadow**;  
 **private** TextField **txtPort**;  
 **private** TextField **txtHost**;  
 **private** TextField **txtOperand1**;  
 **private** TextField **txtOperand2**;  
 **private** Button **btnConnect**;  
 **private** Button **btnDisconnect**;  
 **private** Button **btnSubmit**;  
 **private** RadioButton **rb\_1**;  
 **private** RadioButton **rb\_2**;  
 **private** RadioButton **rb\_3**;  
 **private** RadioButton **rb\_4**;  
 **private** Client **client**= **null**;  
  
 *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
// Создание тени  
//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\** **private void** shadow(){  
 **dropShadow** = **new** DropShadow();  
 **dropShadow**.setRadius(7.0);  
 **dropShadow**.setOffsetX(5.0); *// Shadow offset (X and Y axis)* **dropShadow**.setOffsetY(5.0);  
 **dropShadow**.setColor(Color.***GRAY***); *// Shadow color* }  
*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
// Определение отображения  
//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\** **public void** createLayouts() {  
 **rootPane** = **new** VBox();  
 VBox mainLayout = **new** VBox();  
 HBox mainCenterHLayout = **new** HBox();  
 VBox rightV = **new** VBox();  
 HBox centerBottom = **new** HBox();  
 HBox centerH = **new** HBox();  
 VBox centerV = **new** VBox();  
 GridPane centerGrid = **new** GridPane();  
 ToggleGroup radioGroup = **new** ToggleGroup();  
 **rb\_1** = **new** RadioButton(**"Mul"**);  
 **rb\_2** = **new** RadioButton(**"Div"**);  
 **rb\_3** = **new** RadioButton(**"Sin"**);  
 **rb\_4** = **new** RadioButton(**"Ctg"**);  
  
 **btnConnect** = **new** Button(**"Connect"**);  
 **btnDisconnect** = **new** Button(**"Disconnect"**);  
 **btnSubmit** = **new** Button(**"Submit"**);  
  
 **btnConnect**.setMinWidth(100);  
 **btnDisconnect**.setMinWidth(100);  
 **btnSubmit**.setMinWidth(100);  
  
 **rb\_1**.setToggleGroup(radioGroup);  
 **rb\_2**.setToggleGroup(radioGroup);  
 **rb\_3**.setToggleGroup(radioGroup);  
 **rb\_4**.setToggleGroup(radioGroup);  
 radioGroup.selectToggle(**rb\_1**);  
  
 Text clientTitle = createLabel(**"Client"**, Color.***BLUE***, FontWeight.***BOLD***, 20);  
 Text portTitle = createLabel(**"Port"**, Color.***BLACK***, FontWeight.***NORMAL***, 16);  
 Text hostTitle = createLabel(**"Host"**, Color.***BLACK***, FontWeight.***NORMAL***, 16);  
 Text op1Title = createLabel(**"Operand 1"**, Color.***BLACK***, FontWeight.***NORMAL***, 16);  
 Text op2Title = createLabel(**"Operand 2"**, Color.***BLACK***, FontWeight.***NORMAL***, 16);  
 **logTextArea** = **new** TextArea();  
 **txtPort** = **new** TextField();  
 **txtHost** = **new** TextField();  
 **txtOperand1** = **new** TextField();  
 **txtOperand2** = **new** TextField();  
 **logTextArea**.setWrapText(**true**);  
 **logTextArea**.setMaxWidth(200);  
  
 centerGrid.add(hostTitle, 0, 0);  
 centerGrid.add(portTitle, 0, 1);  
 centerGrid.add(op1Title, 0, 2);  
 centerGrid.add(op2Title, 0, 3);  
  
 centerGrid.add(**txtHost**, 1, 0);  
 centerGrid.add(**txtPort**, 1, 1);  
 centerGrid.add(**txtOperand1**, 1, 2);  
 centerGrid.add(**txtOperand2**, 1, 3);  
  
 centerGrid.setPadding(**new** Insets(10, 10, 10, 10));  
 centerGrid.setHgap(8);  
 centerGrid.setVgap(4);  
  
 rightV.getChildren().addAll(**btnConnect**, **btnDisconnect**);  
 rightV.setPadding(**new** Insets(10, 10, 10, 10));  
 rightV.setSpacing(5);  
 centerH.getChildren().addAll(centerGrid, rightV);  
 centerH.setPadding(**new** Insets(0, 0, 0, 0));  
 centerBottom.getChildren().addAll(**rb\_1**, **rb\_2**, **rb\_3**, **rb\_4**, **btnSubmit**);  
 centerV.getChildren().addAll(centerH, centerBottom);  
  
 centerBottom.setSpacing(10);  
 centerBottom.setPadding(**new** Insets(10, 10, 10, 10));  
  
 shadow();  
 mainCenterHLayout.getChildren().addAll(**logTextArea**, centerV);  
  
 mainLayout.getChildren().addAll(clientTitle, mainCenterHLayout);  
 mainLayout.setEffect(**dropShadow**);  
 mainLayout.setPadding(**new** Insets(10, 10, 10, 10));  
 **rootPane**.getChildren().addAll(mainLayout);  
 }  
 *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
// Обработчик события  
//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\** **private void** onAction(){  
 **btnConnect**.setOnAction(**new** EventHandler<ActionEvent>() {  
 @Override  
 **public void** handle(ActionEvent e) {  
 String host = **""**;  
 **int** port = 0;  
  
 **try** {  
 host = **txtHost**.getText();  
 port = Integer.*valueOf*(**txtPort**.getText());  
 }  
 **catch** (Exception exp) {  
 **logTextArea**.appendText(**"Host or port is empty.\n"**);  
 }  
 **client** = **new** Client(host, port, **logTextArea**);  
 **client**.start();  
 }  
 });  
 **rb\_1**.setOnAction(**new** EventHandler<ActionEvent>() {  
 @Override  
 **public void** handle(ActionEvent e)  
 {  
 **txtOperand2**.setDisable(**false**);  
 }  
 });  
  
 **rb\_2**.setOnAction(**new** EventHandler<ActionEvent>() {  
 @Override  
 **public void** handle(ActionEvent e)  
 {  
 **txtOperand2**.setDisable(**false**);  
 }  
 });  
  
 **rb\_3**.setOnAction(**new** EventHandler<ActionEvent>() {  
 @Override  
 **public void** handle(ActionEvent e)  
 {  
 **txtOperand2**.setDisable(**true**);  
 }  
 });  
  
 **rb\_4**.setOnAction(**new** EventHandler<ActionEvent>() {  
 @Override  
 **public void** handle(ActionEvent e)  
 {  
 **txtOperand2**.setDisable(**true**);  
 }  
 });  
  
 **btnSubmit**.setOnAction(**new** EventHandler<ActionEvent>() {  
 @Override  
 **public void** handle(ActionEvent e)  
 {  
 String cmd = **""**;  
 **double** operand1 = Double.*valueOf*(**txtOperand1**.getText());  
 **double** operand2 = Double.*valueOf*(**txtOperand2**.getText());  
  
 **if**(**rb\_1**.isSelected()) {  
 cmd = **"Mul"**;  
 **client**.sendBinaryOperation(cmd, operand1, operand2);  
 }**else if**(**rb\_2**.isSelected()) {  
 cmd = **"Div"**;  
 **client**.sendBinaryOperation(cmd, operand1, operand2);  
 }**else if**(**rb\_3**.isSelected()) {  
 cmd = **"Sin"**;  
 **client**.sendUnaryOperation(cmd, operand1);  
 } **else if**(**rb\_4**.isSelected()) {  
 cmd = **"Ctg"**;  
 **client**.sendUnaryOperation(cmd, operand1);  
 } } });  
 **btnDisconnect**.setOnAction(**new** EventHandler<ActionEvent>() {  
 @Override  
 **public void** handle(ActionEvent e)  
 {  
 **client**.disconnect();  
 }  
 });  
 }  
*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
// Определение отображения  
//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\** **private** Text createLabel(String text, Color color, FontWeight fontWeight, **int** fontSize){  
 Text t = **new** Text();  
 t.setText(text);  
 t.setFont(Font.*font*(**"Arial"**, fontWeight, fontSize));  
 t.setFill(color);  
 **return** t;  
 }  
 *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\** @Override  
 **public void** start(Stage primaryStage) **throws** Exception{  
 createLayouts();  
 onAction();  
  
 Scene scene = **new** Scene(**rootPane**, 520, 240, Color.***TRANSPARENT***);  
 primaryStage.setResizable(**false**);  
 primaryStage.setTitle(**"Лабараторная работа №5- Клиентская часть"**);  
 primaryStage.setScene(scene);  
 primaryStage.show();  
 primaryStage.setOnCloseRequest(**new** EventHandler<WindowEvent>() {  
 **public void** handle(WindowEvent we) {  
 **if**(**client** != **null**)  
 **client**.disconnect();  
 }  
 });  
 }  
 **public static void** main(String[] args) {  
 *launch*(args);  
 }  
}



**package** Labarator\_5;  
  
**public class** Client **extends** Thread {  
  
 **private** TextArea **logs**;  
 **private** String **ip** = **"127.0.0.1"**;  
 **private int port** = 8000;  
 **private** Socket **clientSocket**;  
 **private** BufferedReader **in** = **null**;  
 **private** PrintWriter **out** = **null**;  
  
 *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
// Конструктор  
//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\** **public** Client(String host, **int** port, TextArea lotTextArea) {  
 **this**.**ip** = host;  
 **this**.**port** = port;  
 **this**.**logs** = lotTextArea;  
 }  
*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
// Запуск клиента(Пользовательский)  
//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\** **public void** run() {  
 startClient();  
 }  
*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
// Запуск клиента  
//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\** **private void** startClient() {  
 **clientSocket** = **null**;  
  
 **try** {  
 **clientSocket** = **new** Socket(**ip**, **port**);  
 **in** = **new** BufferedReader(**new** InputStreamReader(**clientSocket**.getInputStream()));  
 **out** = **new** PrintWriter(**new** OutputStreamWriter(**clientSocket**.getOutputStream()));  
  
 } **catch** (UnknownHostException uhe) {**logs**.appendText(**"Unknown Host :"** + **ip**);  
 **clientSocket** = **null**;  
 } **catch** (IOException ioe) {**logs**.appendText(**"Cant connect to server at "** + String.*valueOf*(**port**) +  
 **". Make sure it is running."**);  
 **clientSocket** = **null**;  
 }  
 **if** (**clientSocket** == **null**)  
 **return**;  
 }  
 *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
// Отправка данных на сервер  
//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\** **public void** sendUnaryOperation(String cmd, **double** operand1) {  
 **try** {  
 **logs**.appendText(**"Command: "** + cmd + **"\n"**);  
 **out**.println(cmd);  
 **out**.println(operand1);  
 **out**.flush();  
  
 String res = **in**.readLine();  
 **logs**.appendText(**"Result: "** + res + **"\n"**);  
 } **catch** (IOException ioe) {  
 **logs**.appendText(**"Exception during communication. Server probably closed connection.\n"**);  
 }  
 } *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
// Отправка данных на сервер  
//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\** **public void** sendBinaryOperation(String cmd, **double** operand1, **double** operand2) {  
 **try** {  
 **logs**.appendText(**"Command: "** + cmd + **"\n"**);  
  
 **out**.println(cmd);  
 **out**.println(operand1);  
 **out**.println(operand2);  
 **out**.flush();  
  
 String res = **in**.readLine();  
 **logs**.appendText(**"Result: "** + res + **"\n"**);  
 } **catch** (IOException ioe) {  
 **logs**.appendText(**"Exception during communication. Server probably closed connection.\n"**);  
 }  
 } *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\** **public void** disconnect() { }  
}

