|  |
| --- |
| https://lh6.googleusercontent.com/QcftzNtI05T0Y6fjdSh1Rr2rt8oqZ1IvnLvbn1jLJ7CCyteVir3k-xBLv4SL1wAgWJsRhmmJSR0UW-RP63_GQenE4vVWv05BRoZTsmIcBccVTnfxwmsnNMvjg599x9SqZd8E3dkd |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждениевысшего образования«МИРЭА - Российский технологический университет»РТУ МИРЭА |

Институт Информационных Технологий (ИТ)

Кафедра инструментального и прикладного программного обеспечения (ИППО)

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №6** | |
| **по дисциплине** | |
| «Разработка клиент-серверных приложений» | |
|  | |
| Выполнил студент группы ИКБО-12-18 | Копотов М.А. |
| Принял ассистент кафедры ИППО | Строганкова Н.В. |

|  |  |  |
| --- | --- | --- |
| Практическая работа выполнена | «\_\_»\_\_\_\_\_\_\_2020 г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_2020 г. | *(подпись руководителя)* |

Москва 2020

**Содержание**

[1. Цель практической работы 3](#_Toc528748841)

[2. Задание 3](#_Toc528748842)

[3. Поэтапное выполнение практической работы 3](#_Toc528748843)

[5. Код программы 4](#_Toc528748844)

[6. Результат работы 4](#_Toc528748845)

[7. Вывод 4](#_Toc528748846)

[Список использованных источников 5](#_Toc528748847)

# 1 Цель практической работы

Ознакомление со средой разработки IntellijIDEA и написания простейшего серверного приложения на языке программирования Java, которое потом может быть использовано в качестве основы для дальнейших практических работ.

# Задание

Вариант 13: Калькулятор. Формула должна поступать в виде в виде текста (пример: «4/2»); отдельными значениями данных.

# Поэтапное выполнение практической работы

1. Используя класс java.net.ServerSocket, написать простейший echo-сервер. Технические требования, предъявляемые к серверу: а) порт, на котором запускается сервер — 12345. Транспортный протокол — TCP; б) количество одновременно подключенных клиентов – 1. То есть использование потоков (экземпляров класса Thread) на первом этапе не предусматривается; в) сервер должен обеспечивать echo-функционал, то есть просто передавать обратно клиенту полученный буквенный символ; г) на консоль (с помощью метода System.out.println()) должны выводиться основные этапы работы программы (клиент установил соединение, был получен и передан обратно символ, соединение разорвано).
2. Изменить написанный класс echo-сервера. Сервер должен производить вычисления в соответствии с индивидуальным заданием. В ответ клиент должен получить результат выполненной операции.
3. Написать клиентскую часть Socket: а) клиент должен будет подключиться к созданному ранее серверу; б) клиенту нужно передать данные параметров в соответствии с индивидуальным заданием, в ответ получить результат действий.
4. Произвести сборку серверной и клиентской части в jar-файл
5. Запустить jar-файлы серверной и клиентской части.

# Код программы

Server.java

package com.company;  
import java.io.\*;  
import java.net.ServerSocket;  
import java.net.Socket;  
  
public class Server {  
  
 private static Socket *clientSocket*;  
 private static ServerSocket *serverSocket*;  
 private static BufferedReader *inStream*;  
 private static BufferedWriter *outStream*;  
  
 public static void main(String[] args) {  
 try {  
 try {  
 *serverSocket* = new ServerSocket(8080);  
 System.out.println("Сервер: Запуск");  
 clientSocket = serverSocket.accept();  
 try {  
 inStream = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));  
 outStream = new BufferedWriter(new OutputStreamWriter(clientSocket.getOutputStream()));  
  
 //Обработка входящего сообщения  
 String task = inStream.readLine();  
 System.out.println("От клиента получена строка: " + task);  
  
 String result;  
 try {  
 result = Float.toString(calculate(task));  
 } catch (Exception e) {  
 result = e.getMessage();  
 }  
 System.out.println("Вычислен и отправлен результат: " + result);  
  
 outStream.write(result);  
 outStream.flush();  
  
 } finally {  
 clientSocket.close();  
 inStream.close();  
 outStream.close();  
 }  
 }finally {  
 System.out.println("Сервер: Остановка");  
 }  
 } catch (IOException e) {  
 System.err.println(e);  
 }  
 }  
  
 static float calculate(String task) throws Exception {  
 var strings = task.split(" ");  
 float a = Float.parseFloat(strings[0]);  
 float b = Float.parseFloat(strings[2]);  
  
 switch (strings[1]) {  
 case "+":  
 return (a + b);  
 case "-":  
 return (a - b);  
 case "\*":  
 return (a \* b);  
 case "/":  
 return (a / b);  
 default:  
 throw new Exception();  
 }  
 }  
}

Client.java

package com.company;  
import java.io.\*;  
import java.net.Socket;  
import java.util.Scanner;  
  
public class Client {  
  
 private static Socket *socket*;  
 private static BufferedReader *inStream*;  
 private static BufferedWriter *outStream*;  
  
 public static void main(String[] args) {  
 try {  
 try {  
 System.*out*.println("Клиент: Запуск");  
  
 *socket* = new Socket("localhost", 8080);  
 inStream = new BufferedReader(new InputStreamReader(socket.getInputStream()));  
 outStream = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));  
 Scanner scanner = new Scanner(System.in);  
  
 //Отправка сообщения  
 System.out.println("Введите выражение:");  
 String s = scanner.nextLine();  
  
 //"\n" необходим, поскольку оно является сигналом к тому, что из потока можно читать  
 outStream.write(s + "\n");  
 outStream.flush();  
 System.out.println("Сообщение отправлено");  
  
 //вывод  
 String answerLine;  
 StringBuilder serverAnswer = new StringBuilder();  
 while ((answerLine = inStream.readLine()) != null)  
 serverAnswer.append(answerLine).append("\n");  
 System.out.println("Получен ответ от сервера: " + serverAnswer);  
  
 } finally {  
 System.out.println("Клиент: Остановка");  
 socket.close();  
 inStream.close();  
 outStream.close();  
 }  
 } catch (IOException e) {  
 System.err.println(e);  
 }  
 }  
}

# 5 Результат работы

Изображение выглядит как текст

Автоматически созданное описание

**Рисунок 1.Данные, отправленные на сервер.**

**Изображение выглядит как текст

Автоматически созданное описание**

**Рисунок 2. Данные, полученные сервером.**

# 6 Вывод

В данной практической работе мы ознакомились со средой разработки IntellijIDEA и написали простейшее серверное приложение на языке программирования Java, которое потом может быть использовано в качестве основы для дальнейших практических работ.

# Список использованных источников

1. https://tproger.ru/translations/java-intro-for-beginners/
2. https://www.sqlitetutorial.net/
3. https://www.jetbrains.com/ru-ru/idea/