Міністерство освіти і науки України

Харківський національний університет ім. В. Н. Каразіна

Факультет комп’ютерних наук

Кафедра безпеки інформаційних систем і технологій

Лабораторна робота №11

*з навчальної дисципліни*

«Спеціалізовані мови програмування та проектування електронних елементів і систем»

Виконав:

Студент групи КБ-41

Кравченко Є.М.

Перевірив:

Олійников Р.В.

Харків – 2020 р.

1.Разработать серверное приложение, принимающее текстовые запросы по сети, и реализующее функциональность простого калькулятора.

По сети сервер принимает запрос, содержащий текстовую строку вида “77/11”, “77 / 11” и т.п. (обрабатываются операции сложения, умножения, деления и вычитания). Выход - по команде “exit”, в случае деления на 0 или переполнения - вывод соответствующего сообщения.

Сетевой ввод-вывод дублируется на локальную консоль сервера (System.out), ошибки дублируются - на System.err.

Проверка функциональности: подключиться через telnet на соответствующий порт (telnet IP порт, например, telnet localhost 9876), ввести данные, увидеть результат на экране.

**public class** Task1 {  
 **private int port**;  
 **private boolean work** = **true**;  
  
 **public** Task1(**int** port) {  
 **this**.**port** = port;  
 }  
  
 **public void** stop() {  
 **work** = **false**;  
 }  
  
 **public void** start() **throws** IOException {  
 **while** (**work**) {  
 ServerSocket serverSocket = **new** ServerSocket(**port**);  
 Socket socket = serverSocket.accept();  
 System.***out***.println(**"Have request!"**);  
  
 **try** (BufferedReader br = **new** BufferedReader(**new** InputStreamReader(socket.getInputStream()));  
 PrintWriter pw = **new** PrintWriter(socket.getOutputStream(), **true**)) {  
 pw.write(**"Welcome to simple calculator"** + System.*lineSeparator*());  
 pw.flush();  
 **while** (**work**) {  
 pw.write(**"Enter your task:"** + System.*lineSeparator*());  
 pw.flush();  
 String request = br.readLine();  
 **if** (**"exit"**.equals(request)) {  
 **work** = **false**;  
 **break**;  
 }  
 request = request.replace(**" "**, **""**);  
 **if** (request.matches(**"(-?\\d+\\.?\\d\*)\\+(-?\\d+\\.?\\d\*)"**)) {  
 String[] added = request.split(**"\\+"**);  
 pw.write(Double.*parseDouble*(added[0])  
 + Double.*parseDouble*(added[1])  
 + System.*lineSeparator*());  
 } **else if** (request.matches(**"(-?\\d+\\.?\\d\*)-(-?\\d+\\.?\\d\*)"**)) {  
 String[] added = request.split(**"-"**);  
 pw.write(Double.*parseDouble*(added[0])  
 - Double.*parseDouble*(added[1])  
 + System.*lineSeparator*());  
 } **else if** (request.matches(**"(-?\\d+\\.?\\d\*)\\\*-?(\\d+\\.?\\d\*)"**)) {  
 String[] added = request.split(**"\\\*"**);  
 pw.write(Double.*parseDouble*(added[0])  
 \* Double.*parseDouble*(added[1])  
 + System.*lineSeparator*());  
 } **else if** (request.matches(**"(-?\\d+\\.?\\d\*)/(-?\\d+\\.?\\d\*)"**)) {  
 String[] added = request.split(**"/"**);  
 pw.write(Double.*parseDouble*(added[0])  
 / Double.*parseDouble*(added[1])  
 + System.*lineSeparator*());  
 } **else** {  
 pw.write(**"Incorrect input"** + System.*lineSeparator*());  
 }  
 pw.flush();  
 }  
 } **catch** (Exception e) {  
  
 **work** = **false**;  
 }  
  
 }  
 }  
}

2.Сделать серверное приложение многопоточным, когда на каждый клиентский запрос создается отдельный поток исполнения и каждый клиент работает независимо.

**public class** Task2 {  
 **private int port**;  
 **private boolean work** = **true**;  
  
 **public** Task2(**int** port) {  
 **this**.**port** = port;  
 }  
  
 **public void** exit() {  
 **work** = **false**;  
 }  
  
 **public void** start() {  
 **try** {  
 ServerSocket serverSocket = **new** ServerSocket(**port**);  
 System.***out***.println(**"Server started"**);  
 **while** (**work**) {  
 Socket socket = serverSocket.accept();  
 System.***out***.println(**"Have new connection: "** + socket);  
 Thread handler = **new** Thread(**new** Handler(socket));  
 handler.start();  
 }  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 }  
}

**public class** Handler **implements** Runnable {  
 **private** Socket **socket**;  
  
 **public** Handler(Socket socket) {  
 **this**.**socket** = socket;  
 }  
  
 @Override  
 **public void** run() {  
 **boolean** work = **true**;  
 **try** (BufferedReader br = **new** BufferedReader(**new** InputStreamReader(**socket**.getInputStream()));  
 PrintWriter pw = **new** PrintWriter(**socket**.getOutputStream(), **true**)) {  
 pw.write(**"Welcome to calculator"** + System.*lineSeparator*());  
 **while** (work) {  
 **try** {  
 pw.write(**"Input task:"** + System.*lineSeparator*());  
 pw.flush();  
 String request = br.readLine().replace(**" "**, **""**);  
 **if** (**"exit"**.equals(request)) {  
 work = **false**;  
 **break**;  
 }  
 pw.write(request + **" = "**);  
 **if** (request.matches(**"(-?\\d+\\.?\\d\*)\\+(-?\\d+\\.?\\d\*)"**)) {  
 String[] added = request.split(**"\\+"**);  
 pw.write(Double.*parseDouble*(added[0])  
 + Double.*parseDouble*(added[1])  
 + System.*lineSeparator*());  
 } **else if** (request.matches(**"(-?\\d+\\.?\\d\*)-(-?\\d+\\.?\\d\*)"**)) {  
 String[] added = request.split(**"-"**);  
 pw.write(Double.*parseDouble*(added[0])  
 - Double.*parseDouble*(added[1])  
 + System.*lineSeparator*());  
 } **else if** (request.matches(**"(-?\\d+\\.?\\d\*)\\\*-?(\\d+\\.?\\d\*)"**)) {  
 String[] added = request.split(**"\\\*"**);  
 pw.write(Double.*parseDouble*(added[0])  
 \* Double.*parseDouble*(added[1])  
 + System.*lineSeparator*());  
 } **else if** (request.matches(**"(-?\\d+\\.?\\d\*)/(-?\\d+\\.?\\d\*)"**)) {  
 String[] added = request.split(**"/"**);  
 pw.write(Double.*parseDouble*(added[0])  
 / Double.*parseDouble*(added[1])  
 + System.*lineSeparator*());  
 } **else** {  
 pw.write(**"Incorrect input"** + System.*lineSeparator*());  
 }  
 pw.flush();  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 work = **false**;  
 }  
 }  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 }  
}

3.Разработать клиента для серверного калькулятора, который выполняет 5 запросов на расчеты через случайные интервалы времени (длиной от 3 до 7 секунд). Данные на расчетов тоже генерируются случайно (запрос с делением на 0 допускается). Данные, отправляемые и принимаемые по сети, дублируются на локальную консоль клиента (System.out).

**public class** Task3 **implements** Runnable {  
 **private final int port**;  
 **private final** String **host**;  
 **private boolean work** = **true**;  
  
 **public** Task3(String host, **int** port) {  
 **this**.**host** = host;  
 **this**.**port** = port;  
 }  
  
 **public void** exit() {  
 **work** = **false**;  
 }  
  
 @Override  
 **public void** run() {  
 **try** {  
 Socket socket = **new** Socket(**host**, **port**);  
 System.***out***.println(**"Client started"**);  
 **try** (BufferedReader br = **new** BufferedReader(**new** InputStreamReader(socket.getInputStream()));  
 PrintWriter pw = **new** PrintWriter(socket.getOutputStream(), **true**)) {  
 **while** (**work**) {  
 Thread.*sleep*(2000);  
 String sign = **""**;  
 **double** rand = Math.*random*();  
 **if** (rand <= 0.25) {  
 sign = **"+"**;  
 } **else if** (rand > 0.25 && rand <= 0.5) {  
 sign = **"/"**;  
 } **else if** (rand > 0.5 && rand <= 0.75) {  
 sign = **"\*"**;  
 } **else if** (rand > 0.75 && rand <= 1) {  
 sign = **"-"**;  
 }  
 pw.write((String.*format*(**"%.2f"**, Math.*random*() \* 100) + sign + String.*format*(**"%.2f"**, Math.*random*() \* 100) + System.*lineSeparator*()));  
 pw.flush();  
 System.***out***.println(br.readLine());  
 }  
 }  
 socket.close();  
 } **catch** (IOException | InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
}

4.Создать java-приложение, параллельно запускающее 3 независимых клиента для серверного калькулятора (клиенты работают в рамках одного приложения).

**new** Thread(App::*task3*).start();  
**new** Thread(App::*task3*).start();  
**new** Thread(App::*task3*).start();

**private static void** task3() {  
 **new** Thread(**new** Task3(**"localhost"**, 9999)).start();  
}

5.Разработать java-приложение, выполяющее подключение к Twitter и выполяющее:

- поиск по ключевому слову;

- вывод списка сообщений пользователя (timeline).