Веб- технологии. Лабораторная работа №1/2. Клиент-сервер.

Часть 1. Теоретическая.

Модель OSI		Модель TCP/IP	
Прикладной уровень (application layer)	7	4	Прикладной уровень (application layer)
Уровень представления (presentation layer)	6		
Сеансовый уровень (session layer)	5		
Транспортный уровень (transport layer)	4	3	Транспортный уровень (transport layer)
Сетевой уровень (network layer)	3	2	Межсетевой уровень (internet layer)
Канальный уровень (data link layer)	2	1	Канальный уровень (link layer)
Физический уровень (physical layer)	1		

Рисунок 1. Модели передачи данных.

Тема: *Динамические веб-сайты – серверное программирование*

https://developer.mozilla.org/ru/docs/Learn/Server-side

Модель OSI&TCP/IP:

https://medium.com/@yangmuxizi/tcp-ip-vs-osi-какие-различия-у-этих-двух-моделей-7f6e6c7c12ce

Межсетевое взаимодействие:

https://kartaslov.ru/карта-знаний/Межсетевое+взаимодействие

Клиент сервер:

https://habr.com/ru/post/495698/ (для тех кто не в теме)

https://developer.mozilla.org/ru/docs/Learn/Server-side/First_steps/Client-Server_overview (для тех кто в теме)

Протоколы:

https://otus.ru/nest/post/1919/ (коротко и по теме)

Веб- технологии. Лабораторная работа №1/2. Клиент-сервер.

Сокеты:

https://ru.hexlet.io/blog/posts/chto-takoe-websocket-i-kak-oni-voobsche-rabotayut (Общая информация)

https://lecturesnet.readthedocs.io/net/low-level/ipc/socket/intro.html (для C++ программистов)

https://habr.com/ru/post/330676/ (для Java программистов)

…остальные типы программистов проигнорированы как незначительные

Часть 2. Практическая.

2.1 Рассмотрим простую программу:

```java
package iit.mal.wt23.lab01;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.Inet4Address;
import java.net.Socket;

public class socketExample {

    public static void main(String[] args) throws IOException {
//        http - 80
//        https - 443
        var inetAddress : InetAddress = Inet4Address.getByName("google.com");
        try (var socket = new Socket(inetAddress, port: 80);
             var outputStream = new DataOutputStream(socket.getOutputStream());
             var inputStream = new DataInputStream(socket.getInputStream())) {
            outputStream.writeUTF( str: "Hello world!");
            var response : byte[] = inputStream.readAllBytes();
            System.out.println(response.length);
        }

    }
```

Результат выполнения программы:

```
179

Process finished with exit code 0
```

```java
package iit.mal.wt23.lab01;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.Inet4Address;
import java.net.Socket;

public class socketExample {

    public static void main(String[] args) throws IOException {
//        http - 80
//        https - 443
        var inetAddress : InetAddress = Inet4Address.getByName("google.com");
        try (var socket = new Socket(inetAddress, port: 443);
             var outputStream = new DataOutputStream(socket.getOutputStream());
             var inputStream = new DataInputStream(socket.getInputStream())) {
            outputStream.writeUTF( str: "Hello world!");
            var response : byte[] = inputStream.readAllBytes();
            System.out.println(response.length);
        }

    }
```

Результат выполнения программы:

```
7

Process finished with exit code 0
```

Мы получили разную длину ответа от сервера (179 и 7). Почему полученные цифры разные и что содержится в ответе?

Задание: необходимо модифицировать программу:

- таким образом чтобы пользователь мог читать ответ от сервера в удобочитаемом текстовом виде;

- работать с любыми другими веб серверами;

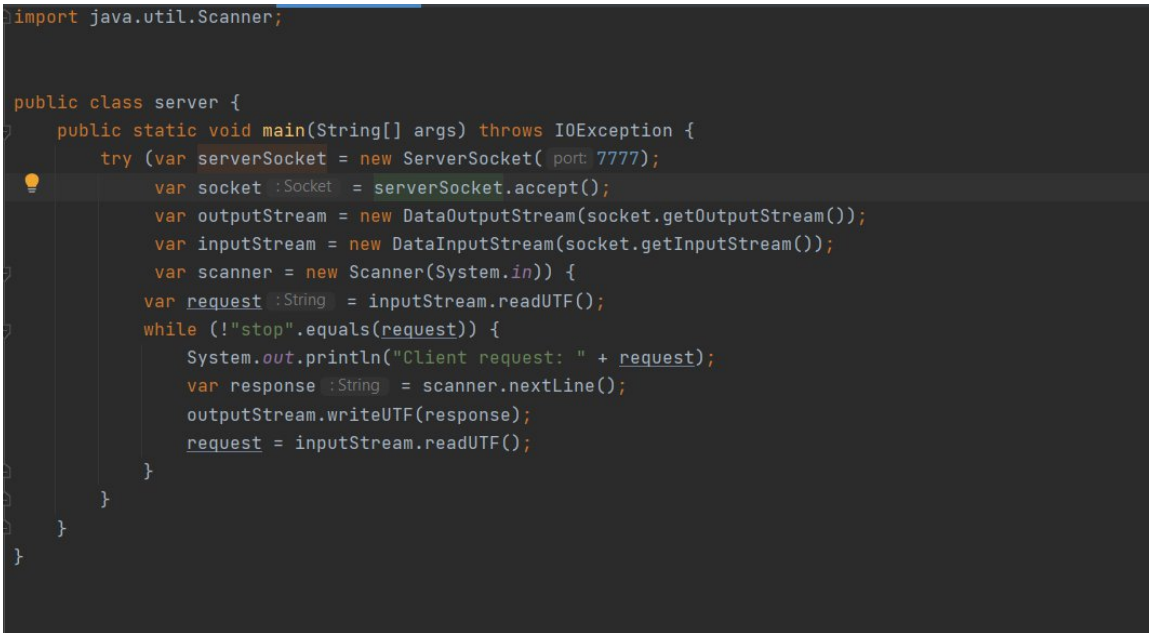
- работать с другими протоколами, например telnet или ssh;

- корректно обрабатывать time connection error если сервер не поддерживает указанный протокол и порт или разорвал соединение;

- корректно обрабатывать ошибку несуществующего сервера протокола или порта.

2.2 Напишем простое клиент-серверное приложение:

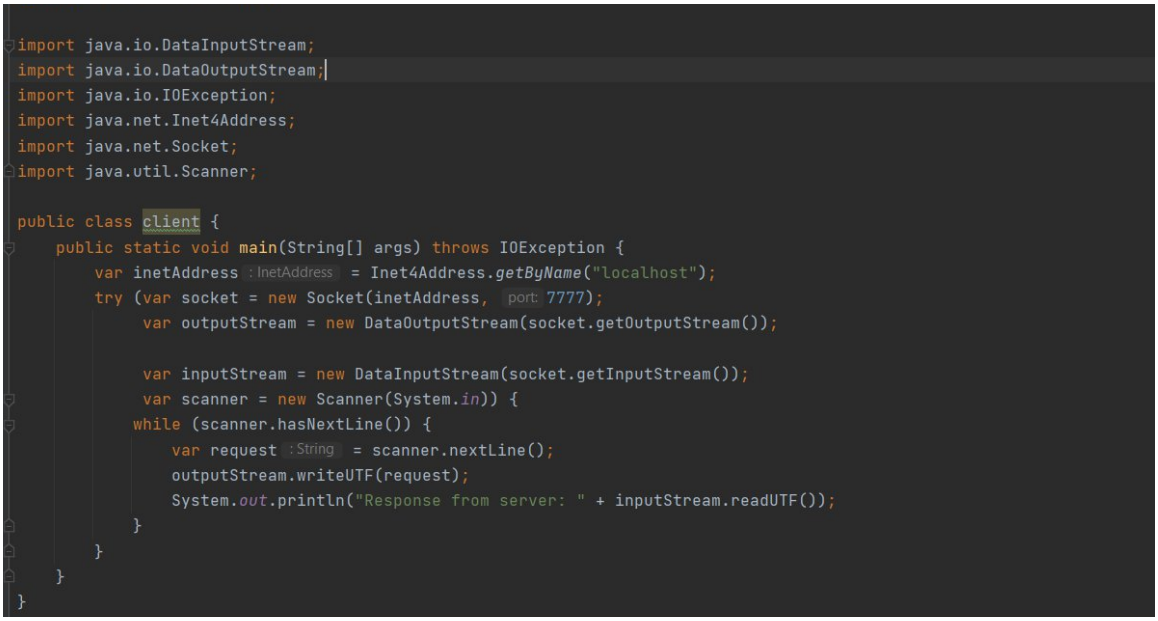
Сервер:



```
import java.util.Scanner;


public class server {
    public static void main(String[] args) throws IOException {
        try (var serverSocket = new ServerSocket( port: 7777);
             var socket :Socket = serverSocket.accept();
             var outputStream = new DataOutputStream(socket.getOutputStream());
             var inputStream = new DataInputStream(socket.getInputStream());
             var scanner = new Scanner(System.in)) {
            var request :String = inputStream.readUTF();
            while (!"stop".equals(request)) {
                System.out.println("Client request: " + request);
                var response :String = scanner.nextLine();
                outputStream.writeUTF(response);
                request = inputStream.readUTF();
            }
        }
    }
}
```

И клиент:



```
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.Inet4Address;
import java.net.Socket;
import java.util.Scanner;

public class client {
    public static void main(String[] args) throws IOException {
        var inetAddress :InetAddress = Inet4Address.getByName("localhost");
        try (var socket = new Socket(inetAddress, port: 7777);
             var outputStream = new DataOutputStream(socket.getOutputStream());

             var inputStream = new DataInputStream(socket.getInputStream());
             var scanner = new Scanner(System.in)) {
            while (scanner.hasNextLine()) {
                var request :String = scanner.nextLine();
                outputStream.writeUTF(request);
                System.out.println("Response from server: " + inputStream.readUTF());
            }
        }
    }
}
```

Запустим сначала сервер, потом клиент.

Передадим на клиенте строку «Hello world»

```
"Hello world"
```

и посмотрим результат на сервере.

Веб- технологии. Лабораторная работа №1/2. Клиент-сервер.

Задание: необходимо модифицировать программу таким образом:

Клиент должен:

- - отобразить время обработки запроса сервером;
- - пользователь мог выбирать адрес сервера и порт;
- - посылать на сервер несколько различных команд;
- - получать ответы с сервера согласно поданным командам;
- - отправлять команду «конца связи» которая корректно завершает программу. (пример: «end», «bye», «stop», «exit»)

Сервер должен:

- - уметь работать с несколькими (2 и более) клиентами;
- - иметь несколько режимов (состояний работы):
  - готов, пауза, остановлен;
- - уметь реагировать на различные команды от различных клиентов;
- - изменять своё состояние (ready, pause, stopped в зависимости от полученной команды;
- - реагировать на несуществующие команды;
- - корректно завершать работу по команде завершения работы от клиента.

Задание со звёздочкой

* вести лог своего состояния, времени обработки запроса, подключённых клиентов и команд, поступивших от них в текстовом файле.

2.3 Сервер дейтаграмм UDP:

```java
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;

public class datagramServer {

    public static void main(String[] args) throws IOException {
        try (var datagramServer = new DatagramSocket( port: 7777)) {
            byte[] buffer = new byte[512];
            DatagramPacket packet = new DatagramPacket(buffer, buffer.length);
            datagramServer.receive(packet);

            System.out.println(new String(buffer));
        }
    }
}
```

Клиент:

```java
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;

public class datagramClient {
    public static void main(String[] args) throws IOException {
        var inetAddress : InetAddress = InetAddress.getByName("localhost");
        try (var datagramSocket = new DatagramSocket()) {
            var bytes : byte[] = "Hello from UDP client".getBytes();
            DatagramPacket packet = new DatagramPacket(bytes, bytes.length, inetAddress, port: 7777);
            datagramSocket.send(packet);
        }
    }
}
```

Результат выполнения программы:

```
Hello from UDP client

Process finished with exit code 0
```

Задание: необходимо модифицировать программу таким образом:

- добавить возможность отобразить время обработки запроса;

- задавать размер и тип передаваемых данных.

По результату изменения программ сравнить время выполнения запросов и объяснить полученные результаты.

Часть 3. Контрольные вопросы.

По результатам выполнения лабораторной работы преподаватель надеется, что студент получил достаточные теоретические и практические знания и в состоянии ответить на следующие вопросы:

1.Стек протокола TCP/IP (количество уровней, в чем отличие от классической модели OSI?)

2.Основные различия между протоколами TCP и UDP.

3. Что такое IP - адрес? Из чего он состоит и как формируется?

4. Для чего нужна маска сети? Что можно определить с помощью маски?

5. Что такое MAC адрес? Как он записывается, для чего нужен? Какие отличия MAC адреса от IP?

6. Клиент-серверная архитектура. Функции сервера и клиента. Какие типы клиентов бывают?

7. Что такое сокеты? Где используются сокеты? Для чего они нужны и как с ними работать? В чём отличие сокета от порта?

Для защиты лабораторной работы необходимо:

- Прочитать и понять теоретическую часть.
- **Индивидуально** выполнить все задания (теоретические и практические) согласно варианта - если такой присутствует. Принимается первый уникальный код, все остальные признаются плагиатом или репликами первого и не учитываются.
- Понимать, что делает каждая строка написанного кода.
- Оформить отчёт, который должен быть распечатан или написан вручную на листах формата А4 (цвет листов не важен) и содержать титульный лист и ход выполнения работы. Выводы о результатах проделанной работы приветствуются.

  * использование репозитория (git или аналоги) с открытым доступом приветствуется.

  P.S. Для тех, кто имеет навык чтения и желает получить более глубокие знания по теме, к лабораторной работе приложены книги. Находятся на сервере в папке с лабораторной работой.