Java Pro 4.3

TEL-RAN
by Starta Institute

Сетевое программирование. JSON

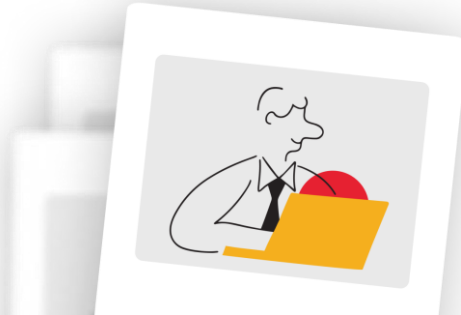
ПРЕПОДАВАТЕЛЬ

TEL-RAN
by Starta Institute

Юрий Костяной

Java/Kotlin backend-разработчик

- 3+ года опыта в коммерческой разработке
- 2+ года опыта в преподавании
- Проекты по интеграции сторонних платформ, CRM
- Проблемно-ориентированный подход в преподавании



ВАЖНО:

- Камера должна быть включена на протяжении всего занятия.
- Если у Вас возник вопрос в процессе занятия, пожалуйста, поднимите руку и дождитесь, пока преподаватель закончит мысль и спросит Вас, также можно задать вопрос в чате или когда преподаватель скажет, что начался блок вопросов.
- Организационные вопросы по обучению решаются с кураторами, а не на тематических занятиях.
- Вести себя уважительно и этично по отношению к остальным участникам занятия.
- Во время занятия будут интерактивные задания, будьте готовы включить камеру или демонстрацию экрана по просьбе преподавателя.

TEL-RAN
by Starta Institute

ПЛАН ЗАНЯТИЯ

1. Повторение
2. Основной блок
3. Вопросы по основному блоку
4. Домашняя работа

TEL-RAN
by Starta Institute

1 ПОВТОРЕНИЕ ИЗУЧЕННОГО

Повторение

TEL-RAN
by Starta Institute

1. Какие существуют виды потоков ввода/вывода?
2. Назовите основные предки потоков ввода/вывода.
3. Что общего и чем отличаются следующие потоки: InputStream, OutputStream, Reader, Writer?
9. Какой класс-надстройка позволяет ускорить чтение/запись за счет использования буфера?
11. Какой класс предназначен для работы с элементами файловой системы?
15. Что такое сериализация?
16. Какие условия “благополучной” сериализации объекта?

Ответы: https://javastudy.ru/interview/input-output/

Повторение

TEL-RAN
by Starta Institute

Исправьте ошибку в коде

```
import java.nio.file.Path;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(Path.of("file.dat"));
        String input = scanner.nextLine();
    }
}
```

Повторение

TEL-RAN
by Starta Institute

Исправьте ошибку в коде

```
import java.io.IOException;
import java.nio.file.Path;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        try (Scanner scanner = new Scanner(Path.of("file.dat"))){
            String input = scanner.nextLine();
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
}
```

Конструктор Scanner, принимающий путь к файлу бросает IOException, которое компилятор требует отловить или указать в сигнатуре метода. Т.к. объект Scanner всё равно требуется закрывать, то лучше использовать конструкцию try-with-resources, которая позволяет делать и то, и другое.

Повторение

Исправьте ошибку в коде

```
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectOutputStream;

public class Main2 {
    public static void main(String[] args) {
        try (ObjectOutputStream oos = new
ObjectOutputStream(new FileOutputStream("object.dat"))) {
            MyClass obj = new MyClass();
            oos.writeObject(obj);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private static class MyClass {
        private int myField;
    }
}
```

Повторение

TEL-RAN
by Starta Institute

Исправьте ошибку в коде

```
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectOutputStream;

public class Main2 {
    public static void main(String[] args) {
        try (ObjectOutputStream oos = new
ObjectOutputStream(new FileOutputStream("object.dat"))) {
            MyClass obj = new MyClass();
            oos.writeObject(obj);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private static class MyClass {
        private int myField;
    }
}
```

Класс *MyClass* должен реализовывать интерфейс *Serializable*, иначе будет выброшено исключение *NotSerializableException*.

Повторение

TEL-RAN
by Starta Institute

Исправьте ошибку в коде

```
try {
    PrintWriter writer = new PrintWriter("output.txt");
    writer.print("Hello, world!");
} catch (IOException e) {
    e.printStackTrace();
}
```

Повторение

TEL-RAN
by Starta Institute

Исправьте ошибку в коде

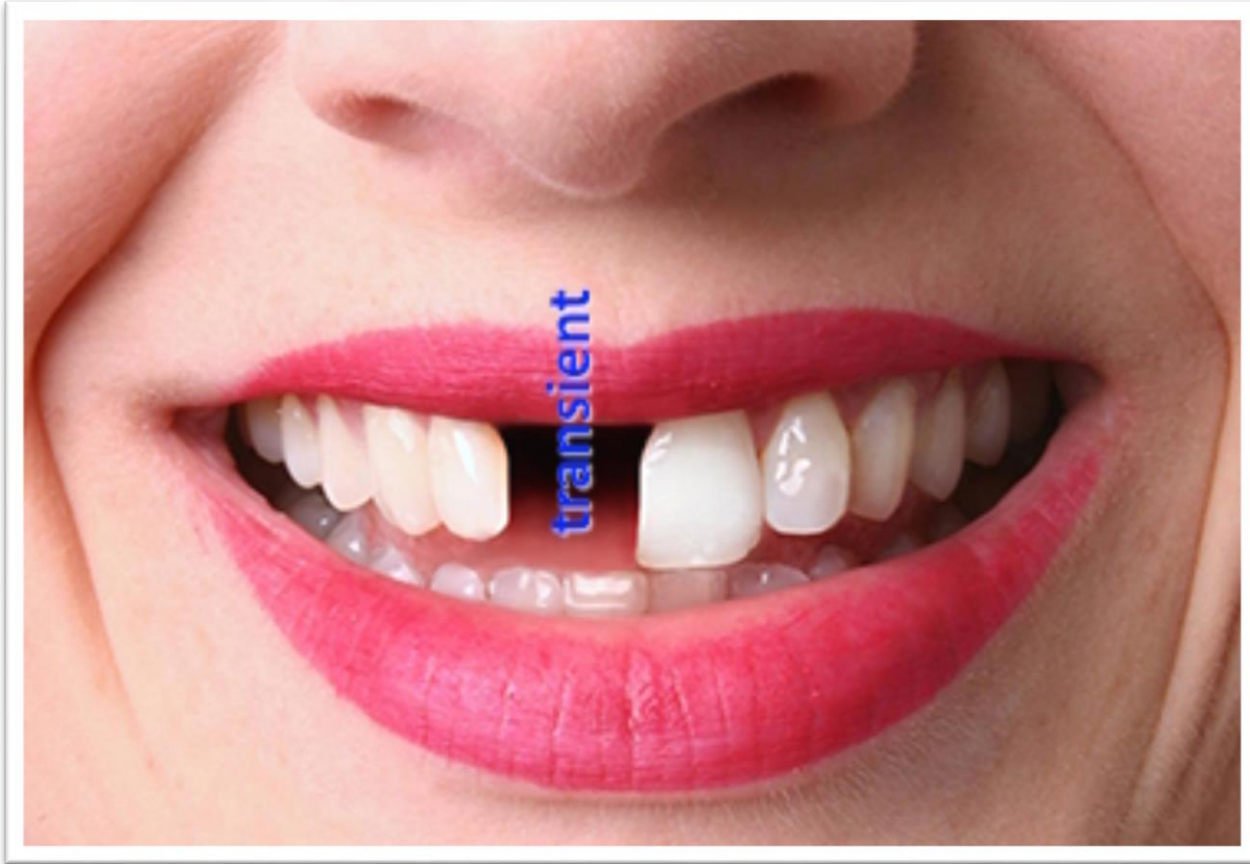
```
try (PrintWriter writer = new PrintWriter("output.txt")) {
    writer.print("Hello, world!");
} catch (IOException e) {
    e.printStackTrace();
}
```

Необходимо закрывать PrintWriter.
Необходимо добавить блок finally или использовать try-with-resources

Повторение

TEL-RAN
by Starta Institute

В чём прикол мема?



TEL-RAN
by Starta Institute

2 ОСНОВНОЙ БЛОК

Введение

TEL-RAN
by Starta Institute

- В сетях
- Ссылочки
- Наметить
- В честь Стэтхэма

Проблема

TEL-RAN
by Starta Institute

Мы уже говорили о том, что механизм потоков ввода-вывода позволяет работать не только с консолью и файловой системой.

Но как работать с сетью? Есть ли принципиальные отличия?

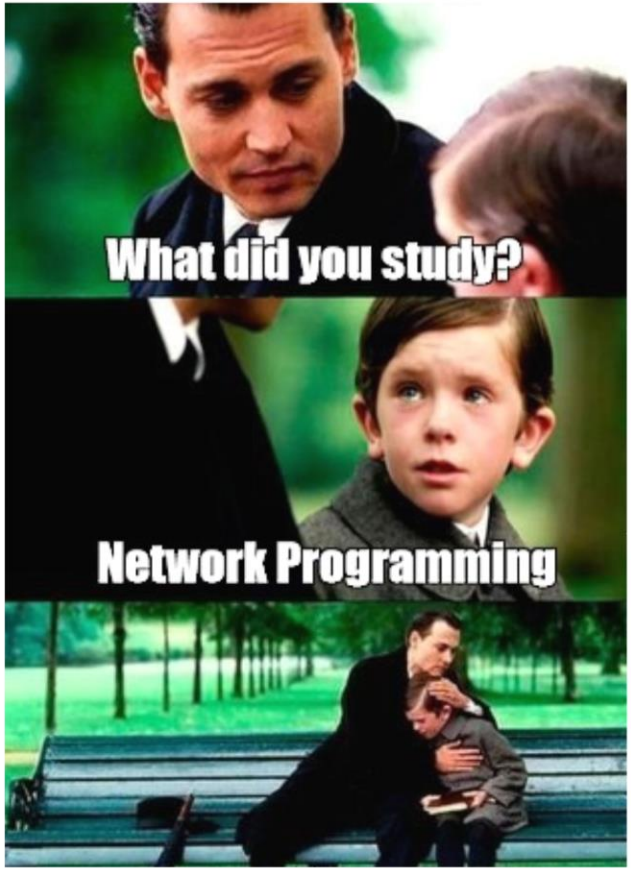
В сетях

Сетевое программирование

TEL-RAN
by Starta Institute

Термин **сетевое программирование** относится к написанию программ, которые выполняются на нескольких устройствах (компьютерах), в которых все устройства подключены друг к другу с помощью сети.

Пакет *java.net* набора интерфейсов прикладного программирования *J2SE* содержит набор классов и интерфейсов, которые предоставляют подробные сведения о низкоуровневом взаимодействии, что позволяет не изобретать велосипед и писать программы, ориентированные на решение существующей проблемы.

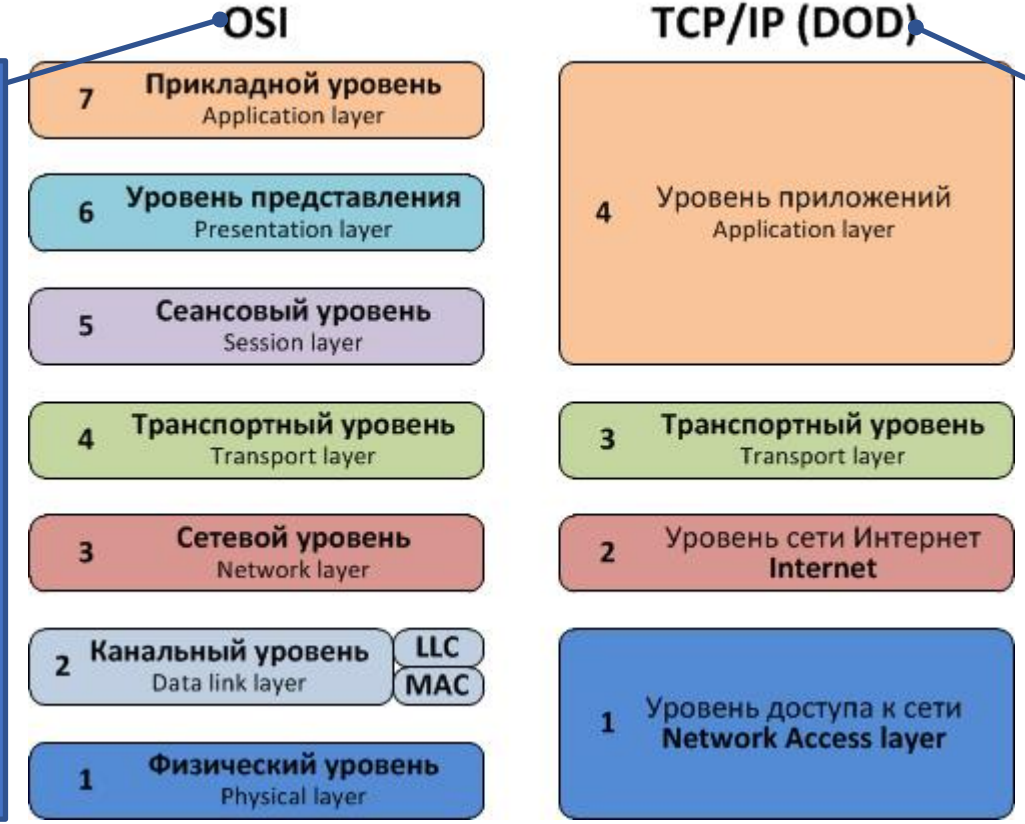


В сетях

Модель OSI и стек TCP/IP

TEL-RAN
by Starta Institute

Модель OSI – это идеализированное представление, как должен осуществляться сетевой обмен данными. В ней выделены 7 основных уровней сетевого взаимодействия



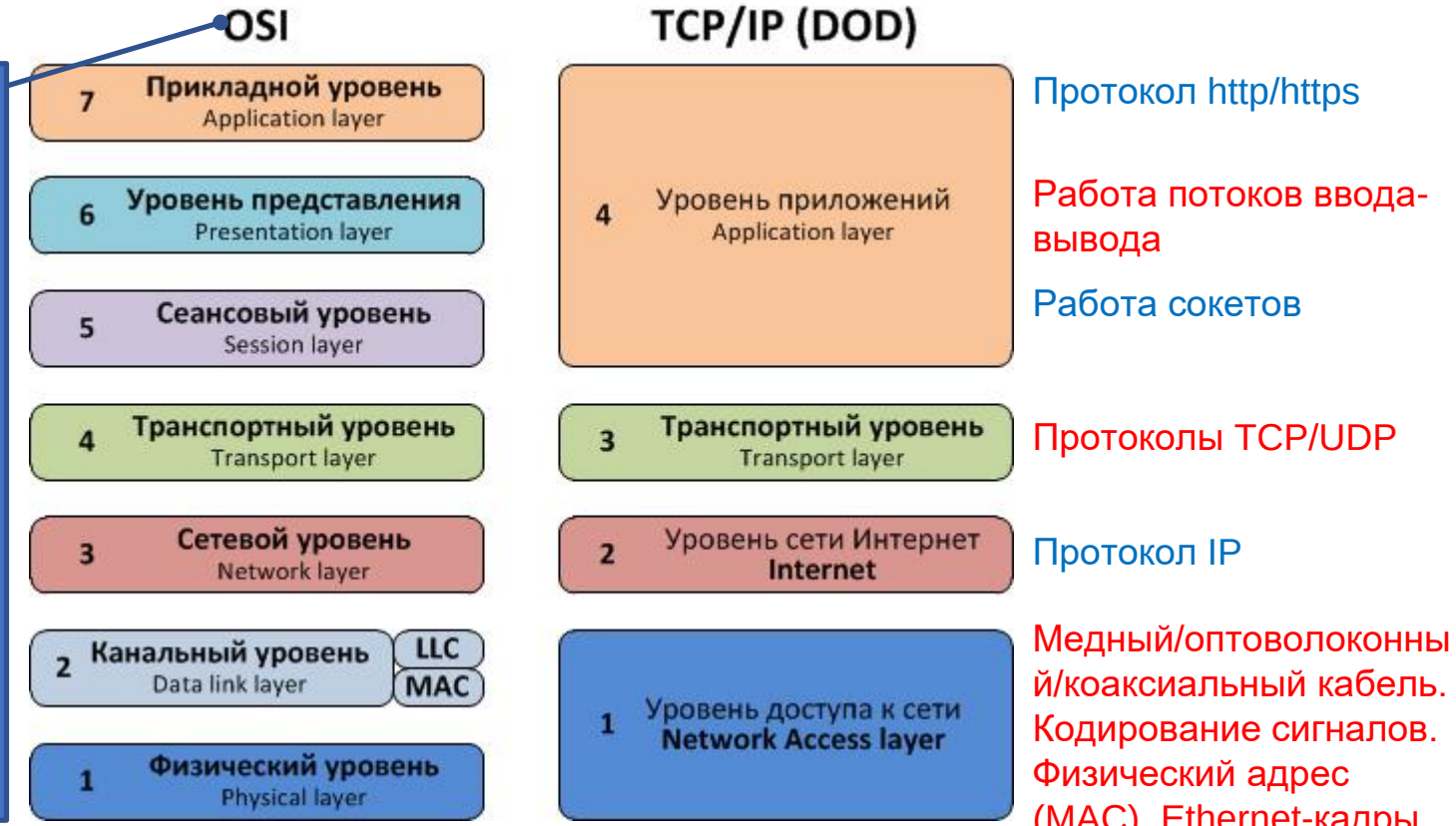
Реальные технологии (стеки протоколов) чаще всего отличаются от модели OSI в сторону упрощения и объединения уровней.

В сетях

Модель OSI и стек TCP/IP

TEL-RAN by Starta Institute

Модель OSI – это идеализированное представление, как должен осуществляться сетевой обмен данными. В ней выделены 7 основных уровней сетевого взаимодействия



В сетях

Протоколы TCP, UDP и HTTP

TEL-RAN
by Starta Institute

Пакет *java.net* обеспечивает поддержку двух общих сетевых протоколов:

TCP – это протокол управления передачей, который обеспечивает надежную связь между двумя приложениями. В Java *TCP* обычно используется через Интернет-протокол, который называется *TCP/IP*.

UDP – это протокол пользовательских дейтаграмм, протокол без установления соединения, который позволяет передавать пакеты данных между приложениями.

HTTP – это протокол для передачи гипертекстовых документов (то есть документов, которые могут содержать ссылки, позволяющие организовать переход к другим документам). Обычно речь идёт про передачу *html* и *json*. *HTTP* предполагает использование клиент-серверной структуры передачи данных. Клиентское приложение формирует запрос и отправляет его на сервер, после чего серверное программное обеспечение обрабатывает данный запрос, формирует ответ и передаёт его обратно клиенту. Данный протокол подробнее разберём при изучении *Spring Framework*.

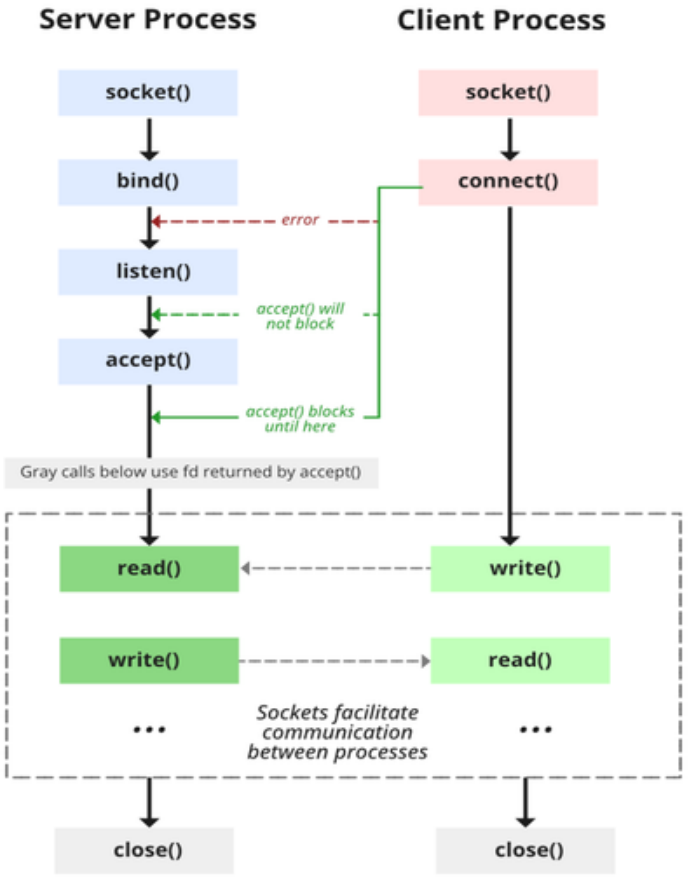
В сетях

Программирование сокетов

TEL-RAN
by Starta Institute

В Java **сокеты** обеспечивают механизм связи между двумя компьютерами, использующими TCP. Клиентская программа создает сокет на своем конце связи и пытается подключить этот сокет к серверу.
Когда соединение установлено, сервер создает объект сокета на своем конце связи. Клиент и сервер теперь могут общаться, записывая и считывая данные с сокета.

Класс **java.net.Socket** представляет собой сокет, а класс **java.net.ServerSocket** предоставляет механизм серверной программы для прослушивания клиентов и установления соединений с ними.



В сетях

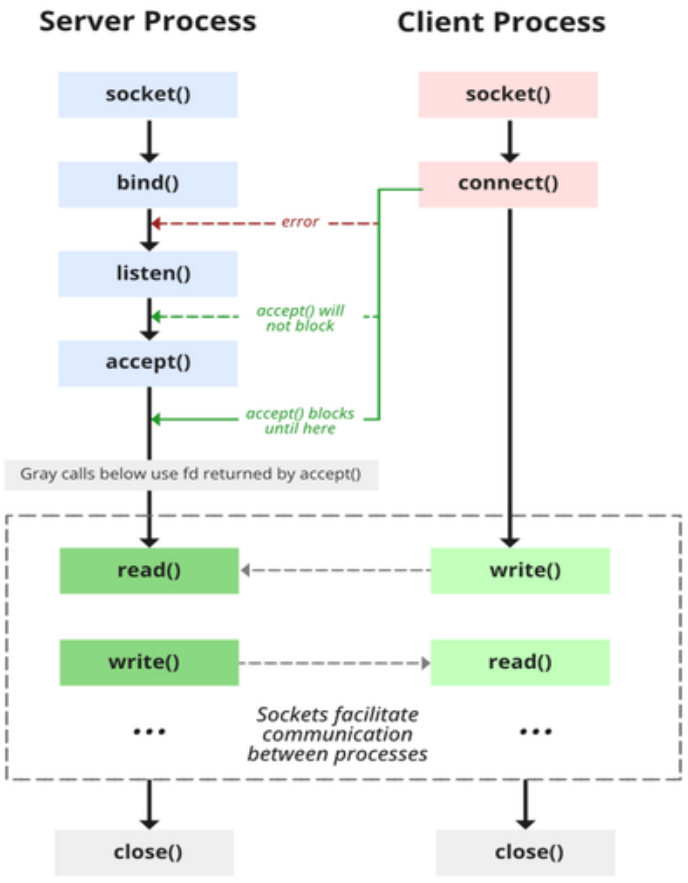
Программирование сокетов

1 Сервер создает экземпляр объекта *ServerSocket*, определяющий, по какому номеру порта должна происходить связь.

2 Сервер вызывает метод accept() класса ServerSocket. Этот метод ожидает, пока клиент не подключится к серверу по указанному порту.

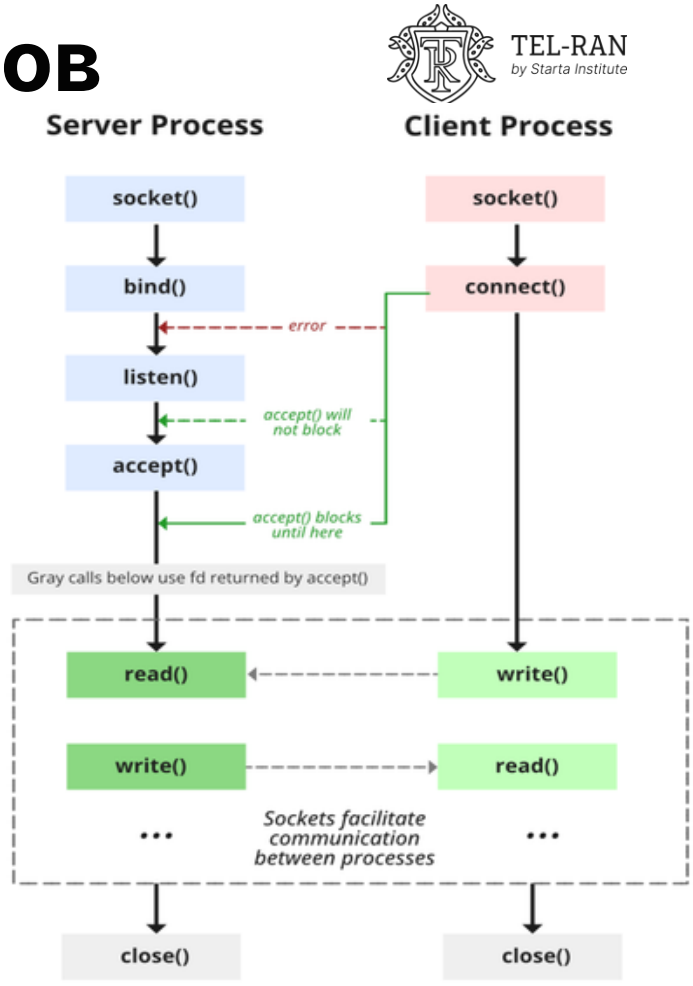
3 По завершению ожидания сервера клиент создает экземпляр объекта сокета, указывая имя сервера и номер порта подключения.

4 Конструктор класса *Socket* осуществляет попытку подключить клиента к указанному серверу и номеру порта. Если связь установлена, у клиента теперь есть объект Socket, способный связываться с сервером.



В сетях

Программирование сокетов



5 На стороне сервера метод accept() возвращает ссылку к новому сокету на сервере, который подключен к клиентскому сокету.

6 После того, как соединения установлены, связь может происходить с использованием потоков входных/выходных данных. Каждый сокет имеет и *OutputStream* (поток выходных данных), и *InputStream* (поток входных данных). *OutputStream* клиента подключен к *InputStream* сервера, а *InputStream* клиента подключен к *OutputStream* сервера.

TCP является двусторонним протоколом связи, поэтому данные могут передаваться по обоим потокам одновременно.

В сетях

Конструкторы ServerSocket

TEL-RAN
by Starta Institute

public ServerSocket(int port) throws IOException – попытки создания серверного сокета, связанного с указанным портом. Исключение происходит, если порт уже связан другим приложением.

public ServerSocket(int port, int backlog) throws IOException - как и в предыдущем конструкторе, параметр *backlog* указывает, сколько входящих клиентов нужно сохранить в очереди ожидания.

public ServerSocket(int port, int backlog, InetAddress address) throws IOException – как и в предыдущем конструкторе, параметр *InetAddress* указывает локальный IP-адрес для осуществления привязки. *InetAddress* используется в серверах, которые могут иметь несколько IP-адресов, что позволяет серверу указывать IP-адрес приема запросов клиентов.

public ServerSocket() throws IOException – создает непривязанный сокет сервера. При использовании этого конструктора используйте метод привязки *bind()*, когда будете готовы привязать сокет сервера.

В сетях

Методы ServerSocket

TEL-RAN
by Starta Institute

public int getLocalPort() – возвращает порт, который прослушивает сокет сервера. Этот метод полезен, если вы передали 0 в качестве номера порта в конструкторе и позволили серверу найти порт.

public Socket accept() throws IOException – ожидает входящего клиента. Этот метод блокируется до тех пор, пока клиент не подключится к серверу на указанном порту или не истечет время ожидания сокета, при условии, что значение времени ожидания было установлено с помощью метода *setSoTimeout()*. В противном случае этот метод блокируется на неопределенный срок.

public void setSoTimeout(int timeout) – устанавливает значение времени ожидания клиента сокетом сервера во время accept().

public void bind (SocketAddress address, int backlog) – привязывает сокет к указанному серверу и порту в объекте *SocketAddress*. *И*используйте этот метод, если вы создали *ServerSocket* с помощью конструктора без аргументов.

В сетях

Методы ServerSocket

Когда *ServerSocket* вызывает *accept()*, метод не возвращается, пока клиент не подключится. После того, как клиент все-таки подключится, *ServerSocket* создает новый сокет для неуказанного порта и возвращает ссылку на этот новый сокет. Теперь между клиентом и сервером существует TCP-соединение, и связь может установиться.

В сетях

Демонстрация ServerSocket

TEL-RAN
by Starta Institute

ServerExample.zip

В сетях

Конструкторы Socket

TEL-RAN
by Starta Institute

public Socket(String host, int port) throws UnknownHostException, IOException – конструктор предпринимает попытку подключения к указанному серверу через указанный порт. Если этот конструктор не выдает исключение, то соединение установлено успешно.

public Socket(InetAddress host, int port) throws IOException – этот метод идентичен предыдущему конструктору, за исключением того, что хост обозначается объектом InetAddress.

public Socket(String host, int port, InetAddress localAddress, int localPort) throws IOException – подключается к указанному хосту и порту, создавая сокет на локальном хосте по указанному адресу и порту.

public Socket(InetAddress host, int port, InetAddress localAddress, int localPort) throws IOException – этот метод идентичен предыдущему конструктору, за исключением того, что хост обозначается объектом *InetAddress* вместо строки адреса.

public Socket() – создает неподключенный сокет. Используйте метод *connect()* для подключения такого сокета к серверу.

В сетях

Методы Socket

TEL-RAN
by Starta Institute

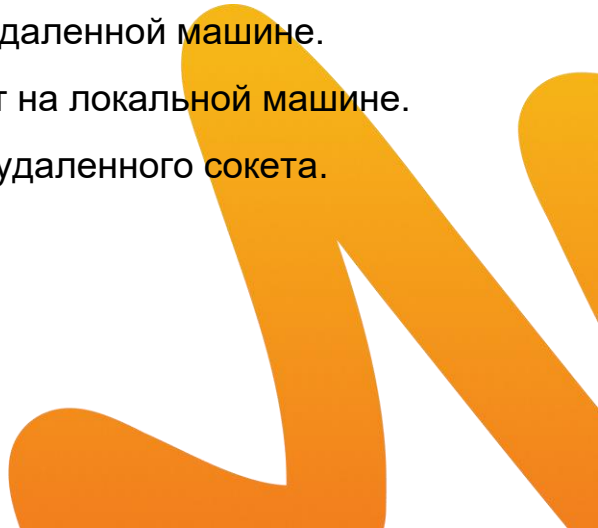
public void connect(SocketAddress host, int timeout) throws IOException – этот метод подключает сокет к указанному хосту. Этот метод необходим только при создании экземпляра сокета с помощью конструктора без аргументов.

public InetAddress getInetAddress() – этот метод возвращает адрес другого компьютера, к которому подключен этот сокет.

public int getPort() – возвращает порт, к которому привязан сокет на удаленной машине.

public int getLocalPort() – возвращает порт, к которому привязан сокет на локальной машине.

public SocketAddress getRemoteSocketAddress() – возвращает адрес удаленного сокета.



В сетях

Методы Socket

TEL-RAN
by Starta Institute

public InputStream getInputStream() throws IOException – возвращает поток входных данных сокета. Поток входных данных подключен к потоку выходных данных удаленного сокета.

public OutputStream getOutputStream() throws IOException – возвращает поток выходных данных сокета. Поток выходных данных подключен к потоку входных данных удаленного сокета.

public void close() throws IOException – закрывает сокет, что делает данный объект сокета не способным снова подключаться к любому серверу.

В сетях

Методы InetAddress

TEL-RAN
by Starta Institute

static InetAddress getByAddress(byte[] addr) – возвращает объект *InetAddress* с учетом необработанного IP-адреса.

static InetAddress getByAddress(String host, byte[] addr) – создает *InetAddress* на основе предоставленного имени хоста и IP-адреса.

static InetAddress getByName(String host) – определяет IP-адрес хоста, учитывая имя хоста.

String getHostAddress() – возвращает строку IP-адреса в текстовой форме.

String getHostName() – получает имя хоста для данного IP-адреса.

static InetAddress InetAddress getLocalHost() – возвращает локальный хост.

String toString() – конвертирует этот IP-адрес в адресную строку.

В сетях

Демонстрация Socket

TEL-RAN
by Starta Institute

ClientExample.zip

Задание

TEL-RAN
by Starta Institute

Напишите простое клиент-серверное приложение, которое сообщает клиенту текущую дату и время при подключении.

Ссылочки
URL

TEL-RAN
by Starta Institute

Для работы с ресурсами в интернете в Java есть специальный класс – *URL*.

Пример получения странички из интернета:

```
// download
URL url1 = new URL("https://www.google.com"); // Создает объект URL с путем к странице
InputStream input = url1.openStream(); // Получает InputStream у интернет-объекта
byte[] buffer = input.readAllBytes(); // Читает все байты и возвращает массив байт
String str = new String(buffer); // Преобразуем массив в строку
System.out.println(str); // Выводим строку на экран
input.close();
```

Ссылочки
URL

TEL-RAN
by Starta Institute

Пример загрузки данных в интернет:

```
// upload
URL url2 = new URL("https://www.google.com"); // Создаем объект URL с путем к странице
URLConnection connection = url2.openConnection();

// получили поток для отправки данных
OutputStream output = connection.getOutputStream(); // Получаем поток вывода
output.write(1); // отправляем данные

// получили поток для чтения данных
InputStream input2 = connection.getInputStream();
int data = input2.read(); // читаем данные
```

Задание

TEL-RAN
by Starta Institute

Через поисковик найдите в интернете любую картинку. По ссылке сохраните её на компьютер.

Решение

TEL-RAN
by Starta Institute

Пример получения данных и сохранения в файл.

Закрытие потоков опущено для простоты.

```
public static void main(String[] args) throws IOException {
    String image = "https://www.google.com/images/branding/googlelogo/1x/googlelogo_color_272x92dp.png";
    downloadAndSave(image);
}

private static void downloadAndSave(String urlStr) throws IOException {
    URL url = new URL(urlStr);
    InputStream input = url.openStream();

    Path path = Path.of("c:\\GoogleLogo.png");
    Files.copy(input, path);
}
```

Проблема

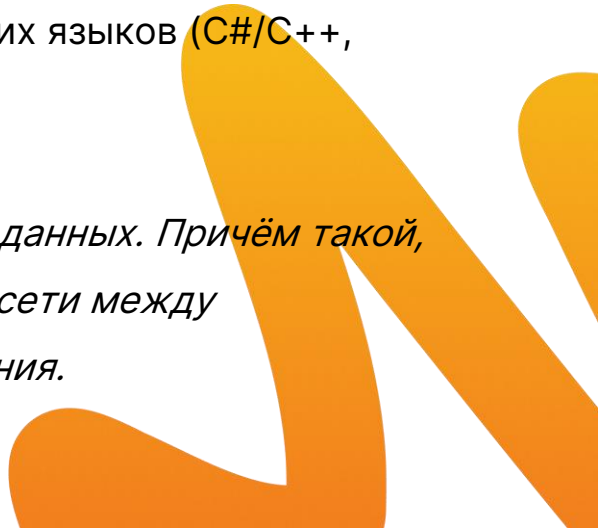
TEL-RAN
by Starta Institute

Сериализация представляет данные для передачи в виде набора байт.

Но у этого подхода есть две существенные проблемы:

1. Сериализованный объект не читаем для человека, т.е. разработчик не может легко оценить содержимое объекта без десериализации;
2. Механизм сериализации работает только в JVM. Для других языков (C#/C++, JavaScript) он может отличаться.

Поэтому нужны общие способ и формат упаковки и распаковки данных. Причём такой, который было бы легко читать человеку и легко передавать по сети между приложениями, написанными на разных языках программирования.



Наметить

Языки разметки

TEL-RAN
by Starta Institute

Языки разметки – это семейство языков представления данных.

Некоторые из этих языков используются для формирования документов (*Markdown*), файлов настроек (*Yaml*), другие – для отображения интернет-страниц в браузере (*html*), третьи – для обмена данными между клиентским и серверным приложениями или между backend-сервисами (*JSON*, *XML*).

XML часто применяется и для других целей. Например, в pom-файлах maven.

В честь Стэтхэма
JSON

JSON – *JavaScript Object Notation* – текстовый формат обмена данными, созданный для *JavaScript*. Но при этом формат независим от *JavaScript* и может использоваться в любом языке программирования.

В честь Стэтхэма
MIME

MIME-тип JSON’a – application/json

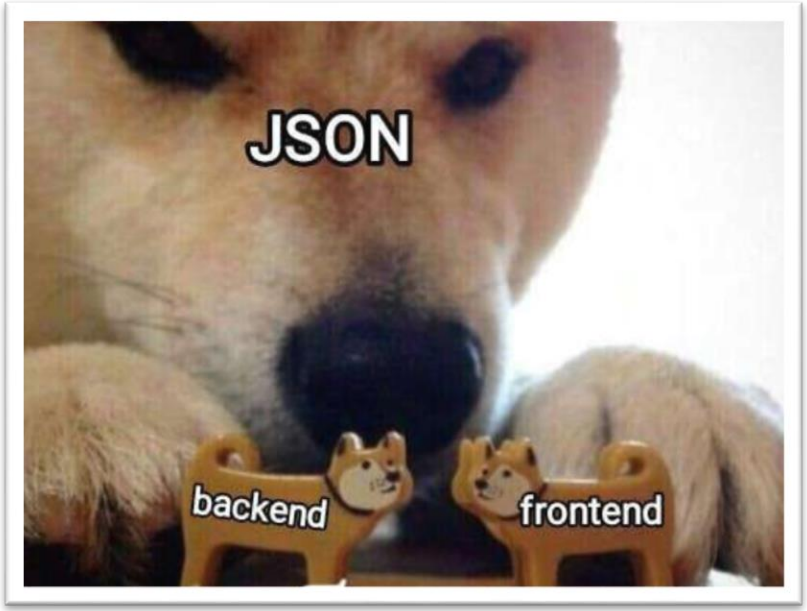
MIME – Multipurpose Internet Mail Extensions – стандарт, описывающий передачу различных типов данных по Интернету.

Если Вы хотите подсказать получателю сообщения, какой тип данных Вы ему передаёте и какой-тип данных от него ожидаете в ответ, то MIME передаётся в соответствующих заголовках Вашего запроса.

В честь Стэтхэма

Применение JSON

За счёт своей лаконичности по сравнению с *XML* формат *JSON* может быть более подходящим для сериализации сложных структур. Применяется в веб-приложениях как для обмена данными между браузером и сервером, так и между серверами.



XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<person>
 <name>Иван</name>
 <age>37</age>
 <mother>
  <name>Ольга</name>
  <age>58</age>
 </mother>
 <children>
  <child>Маша</child>
  <child>Игорь</child>
  <child>Таня</child>
 </children>
 <married>true</married>
 <dog null="true" />
</person>
```

VS

JSON

```
{
 "person":{
  "name":"Иван",
  "age":37,
  "mother":{
   "name":"Ольга",
   "age":58
  },
  "children":[
   "Маша",
   "Игорь",
   "Таня"
  ],
  "married":true,
  "dog":null
 }
```

В честь Стэтхэма

Структура JSON

Структура *JSON* может содержать три типа элементов:

- Пара “ключ”: значение. Ключом может быть только строка. Как правило регистр строки учитывается. Повторяющиеся имена ключей допустимы, но не рекомендуются стандартом; обработка таких ситуаций происходит на усмотрение программного обеспечения.
- Упорядоченный набор значений. Представляет собой массив данных в квадратных скобках. Например, “ключ”: [1, 2, 3, 4];
- Объект. Представляет собой структуру в фигурных скобках {}. Внутри скобок могут быть пары, массивы и другие объекты.

```
{
  "firstName": "Иван",
  "lastName": "Иванов",
  “isMale": true,
  "address": {
    "streetAddress": “Гая ул., 1, кв.10",
    "city": "Ленинград",
    "postalCode": 101101
  },
  "phoneNumbers": [
    "812 123-1234",
    "916 123-4567"
  ]
}
```

В честь Стэтхэма

Библиотеки для работы с JSON

Simple Json – это самый примитивный способ.

Использует два основных класса:

JSONArray – может включать в себя несколько объектов *JSONObject*, его можно обходить циклом, на каждой итерации получая объект *JSONObject*.

JSONObject - объект, из которого можно доставать его отдельные свойства.

```
// Считываем json
        Object obj=new JSONParser().parse(jsonString);
// Object obj = new JSONParser().parse(new FileReader("JSONExample.json"));

// Кастим obj в JSONObject
        JSONObject jo=(JSONObject)obj;
// Достаём firstName and lastName
        String firstName=(String)jo.get("firstName");
        String lastName=(String)jo.get("lastName");
        System.out.println("fio: "+firstName+" "+lastName);
// Достаем массив номеров
        JSONArray phoneNumbersArr=(JSONArray)jo.get("phoneNumbers");
        Iterator phonesItr=phoneNumbersArr.iterator();
        System.out.println("phoneNumbers:");
// Выводим в цикле данные массива
        while(phonesItr.hasNext()){
        JSONObject test=(JSONObject)phonesItr.next();
        System.out.println("- type: "+test.get("type")+", phone: "+test.get("number"));
        }
```

В честь Стэтхэма

Библиотеки для работы с JSON

GSON – это аналог *Simple Json* от Google. т.е. использует те же *JSONObject* и *JSONArray*, но имеет более мощный инструмент парсинга. Достаточно создать классы, которые повторяют структуру Json'a.

```
{
  "firstName": "Иван",
  "lastName": "Иванов",
  “isMale": true,
  "address": {
    "streetAddress": “Гая ул., 1, кв.10",
    "city": "Ленинград",
    "postalCode": 101101
  },
  "phoneNumbers": [
    "812 123-1234",
    "916 123-4567"
  ]
}
```

```
class Person {
    public String firstName;
    public String lastName;
    public boolean isMale;
    public Address address;
    public List<String> phoneNumbers;
}

class Address {
    public String streetAddress;
    public String city;
    public String state;
    public int postalCode;
}
```

В коде получаем объект из Json (можно также распарсить в мапу):

```
Gson g = new Gson();
Person person =
g.fromJson(jsonString, Person.class);
```

В честь Стэтхэма

Библиотеки для работы с JSON

Jackson – это аналог *Gson*. Является основным парсером в *Spring*. Тоже требует классы, которые повторяют структуру Json'а, либо позволяет распарсить в мапу.

Позволяет указывать дополнительные требования за счет аннотаций, например: не парсить указанные поля, использовать кастомный конструктор класса, поменять имя переменной (например не *firstName*, а *first_name*), а также создавать уникальные классы для сериализации и десериализации каких-либо классов.

Подробнее здесь https://www.baeldung.com/jackson

Краткий мануал https://www.baeldung.com/jackson-object-mapper-tutorial

Maven central https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-databind

У класса должен быть конструктор по умолчанию. Основной класс в *Jackson* – это *ObjectMapper*.

```
ObjectMapper mapper = new ObjectMapper();
Person person = mapper.readValue(jsonString, Person.class);
```

В честь Стэтхэма

Библиотеки для работы с JSON

JsonPath – библиотека относится к так называемым *XPath* библиотекам. Её суть аналогична *xpath* в *xml*, то есть позволяет легко получать часть информации из json'а, по указанному пути, не распознавая его целиком. Результат позволяет фильтровать по заданному условию.

```
// Выведет все фамилии друзей
List<String> friendsLastnames = JsonPath.read(jsonString, "$.friends[*].lastName");
for (String lastname : friendsLastnames) {
    System.out.println(lastname);
}

// Поиск друга, которому больше 22 лет
List<String> friendsWithAges = JsonPath
        .using(Configuration.defaultConfiguration())
        .parse(jsonString)
        .read("$.friends[?(@.age > 22)].lastName", List.class);
```

В честь Стэтхэма

Библиотеки для работы с JSON

Существует множество других библиотек для парсинга и создания Json из объектов.

Задание

TEL-RAN
by Starta Institute

1 Конвертировать данные из Json в объект:

Example0.json

2 Создать объект с 3-4 полями разных типов (в т.ч. ссылочных) и записать объект в файл в виде Json.

3 Конвертировать объект из одного класса в другой с помощью метода convertValue класса ObjectMapper. Классы не должны быть связаны общей иерархией, но имеют одинаковый набор полей.

TEL-RAN
by Starta Institute

3 Домашнее задание

Домашнее задание

TEL-RAN
by Starta Institute

1 Конвертировать данные из Json в объект:


Example2.json

2 Создать объект с 3-4 полями разных типов (в т.ч. ссылочных) и записать объект в файл в виде Json.

3 Прочитать значение поля age из файла ниже с помощью JsonPath

https://mvnrepository.com/artifact/com.jayway.jsonpath/json-path

Вывести результат в консоль.


Example1.json

4 Напишите код для скачивания пяти картинок с сайта https://doodles.google/

ЗАКЛЮЧЕНИЕ

TEL-RAN
by Starta Institute