СПбГЭТУ «ЛЭТИ»
ПЕРВЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ

Е.Ю.Белова

Разработка приложений в распределенной среде

Методические рекомендаций по практическим работам

СПбГЭТУ «ЛЭТИ», 2022 г.

1 РАЗРАБОТКА РАСПРЕДЕЛЕННОГО ПРИЛОЖЕНИЯ С ИСПОЛЬЗОВАНИЕМ ТЕХНОЛОГИИ УДАЛЕННОГО ВЫЗОВА МЕТОДОВ

Цель работы: изучение теоретического материала о Remote Method Invocation (RMI) и архитектуре распределенного приложения RMI, формирование практических навыков разработки распределенного медицинского приложения RMI на языке программирования Java.

1.1 Общие сведения

В распределенных средах часто необходимо осуществить вызов методов объектов с учетом условия, что данные объекты находятся на удаленных компьютерах. Изложенный принцип лежит в основе RMI, который обеспечивает платформо-независимый вызов методов [1], [2].

Приложение RMI, как правило, включает в себя две отдельные программы – сервер и клиент. Сервер создает несколько удаленных объектов, делает удаленные ссылки на них доступными и ожидает клиентов для того, чтобы вызвать методы данных объектов. Клиент получает удаленную ссылку на один или несколько удаленных объектов на сервере, вызывает методы и затем использует полученные результаты. RMI обеспечивает работу механизма, с помощью которого сервер и клиент поддерживают связь и обмениваются информацией друг с другом. Такое приложение иногда называют распределенным объектным приложением [1]–[3].

Распределенное приложение RMI содержит интерфейсы и классы. Методы объявляются в интерфейсах, а реализуются в классах [3], [4].

Описание архитектуры распределенного приложения RMI представлено ниже (Рис.1.1). Метод удаленного объекта не вызывается напрямую из объекта клиента. Для взаимодействия с интерфейсом сервера – скелетоном, – имею-щим доступ к реализации метода удаленного объекта, клиент использует заглушку – локального представителя удаленного объекта. Обмен данными фактически выполняется через уровень удаленной ссылки (и транспортный уровень), который сериализует параметры во время вызова и десериализует полученные результаты при возвращении методом значения или объекта [1]. RMI включает в себя несколько этапов [1], [4]:

1. Создание удаленного объекта на сервере. Для того чтобы объект являлся удаленным, необходимо выполнить его экспорт. Процесс экспорта подразумевает создание заглушки – удаленной ссылки на удаленный объект, – с информацией о местоположении объекта и методах, вызов которых возможен удаленно. Сам процесс происходит внутри сервера.

2. Сервер регистрирует заглушку с уникальным именем в реестре RMI. Сервер и реестр должны быть запущены на одной машине по соображениям безопасности.

3. Клиент отправляет запрос на поиск заглушки удаленного объекта в реестре RMI по уникальному имени. Реестр предоставляет заглушку клиенту.

4. Клиент вызывает метод удаленного объекта, используя заглушку. Заглушка является локальным представителем удаленного объекта и у клиента создается впечатление, что он вызывает метод объекта напрямую.

5. Заглушка связывается со скелетоном и пересылает ему вызов и все необходимые параметры. До пересылки выполняется сериализация параметров – маршаллинг.

6. Скелетон выполняет десериализацию параметров – демаршаллинг – и вызывает метод объекта.

7. Возвращаемое значение передается скелетону, который сериализует его и отправляет заглушке.

8. Заглушка выполняет десериализацию возвращаемого значения и отправляет его клиенту.

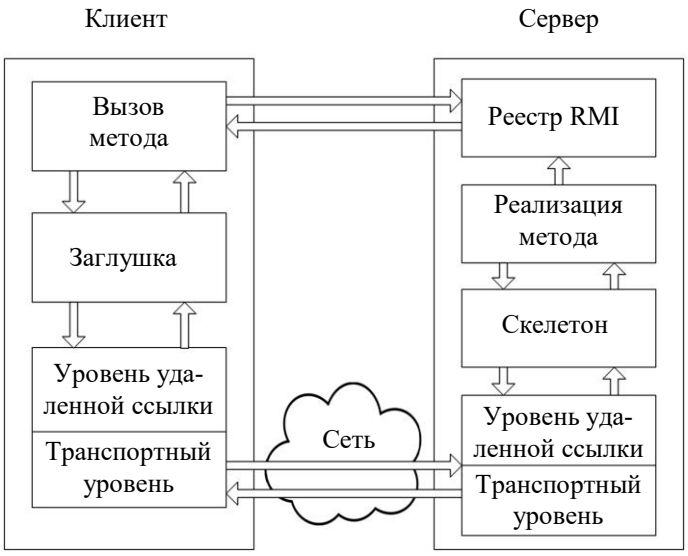


Рис. 1.1

Скелетон существовал только в первых версиях Java и был полностью удален в J2SE 1.2. Взаимодействие происходит напрямую через уровень удаленной ссылки. Тем не менее базовые принципы работы RMI остаются прежними [3].

Уровень удаленной ссылки скрывает описание протокола связи между заглушкой и скелетоном. Данный уровень использует транспортную службу [например, Transmission Control Protocol (TCP)], которая предоставляется транспортным уровнем для осуществления передачи данных [3].

1.2 Задание

Разработать распределенное медицинское приложение с использованием технологии RMI согласно номеру варианта (Таблица 1.1).

Таблица 1.1

№ варианта	*Раздел медицины*	*Название темы*
1	Педиатрия	Оценка физического развития ребенка мужского пола в возрасте до одного года
2	Акушерство	Определение предполагаемой массы и длины плода
3	Анестезиология	Расчет гемодинамических показателей
4	Реабилитология	Оценка результатов функциональных проб с дозированной физической нагрузкой
5	Спортивная медицина	Расчет индексов для оценки антропометрических данных
6	Фармакология	Расчет курсовой дозы элементарного железа для детей
7	Диетология	Оценка статуса питания
8	Терапия	Расчет объема суточной инфузионной терапии

Методы, которые могут быть вызваны клиентом, определить в интерфейсе сервера и реализовать в его классе. Величины, которые необходимо рассчитать, и данные для вывода в консоль представлены ниже (Таблица 1.2).

Таблица 1.2.

№ варианта	*Рассчитываемые величины*	*Данные для вывода*
1	Сигмальные отклонения антропометрических показателей (длины тела, массы тела, окружности головы и окружности груди)	Значения рассчитанных величин и уровень физического развития ребенка (очень низкий, низкий, ниже среднего, средний, выше среднего, высокий, очень высокий) по каждому из показателей
2	Масса плода по формулам Якубовой, Стройковой, Добровольского, средняя длина тела плода по формуле Гаазе	Значения рассчитанных величин
3	Гемодинамические показатели (среднее артериальное давление, индексы работы левого и правого желудочков, индексы общего периферического и легочного сопротивлений)	Значения рассчитанных величин и их сравнение с нормальными показателями центральной гемодинамики
4	Показатель качества реакции сердечно-сосудистой системы на нагрузку, индекс Руфье и индекс Гарвардского степ-теста	Значения рассчитанных величин и оценка результатов каждой из проб
5	Индекс массы тела, индекс Эрисмана, индекс Пирке и индекс Пинье	Значения рассчитанных величин и оценка величины каждого из индексов.
6	Общий дефицит железа, количество ампул для введения, курсовая доза железа и курсовое количество инъекций	Значения рассчитанных величин и длительность курсов лечения с учетом суточных доз железа для парентерального введения
7	Потеря массы тела, индекс массы тела, идеальная масса тела (для обоих полов), окружность мышц плеча.	Значения рассчитанных величин и длительность курсов лечения с учетом суточных доз железа для парентерального введения у детей.

Продолжение таблицы 1.2.

№ варианта	*Рассчитываемые величины*	*Данные для вывода*
8	Физиологические потребности в воде, патологические потери при лихорадке и отдышке, дефицит воды по величине гематокрита и концентрации натрия плазмы, объем суточной инфузии при дегидратации	Значения рассчитанных величин

Формулы для рассчитываемых величин с первого по восьмой варианты представлены в литературе [7-14] соответственно. При вводе переменных с клавиатуры предусмотреть проверку на корректность их типов данных и на попадание значений в допустимые диапазоны.

1.3 Порядок выполнения работы

1.3.1 Создание нового Java-проекта

Скачать Java SE Development Kit и выполнить его установку. Далее настроить переменную окружения JAVA_HOME, которая указывает на каталог установки Java SE Development Kit. Скачать Eclipse IDE for Java Developers с сайта и выполнить его установку.

В Eclipse выбрать пункт меню File=>New=>Java Project. Указать имя проекта – Lab_1_RMI, нажать на Next и затем на Finish. Нажать правой кнопкой мыши на src и выбрать New=>Package. Задать имя пакета – ru.eltech.rmi.

1.3.2 Создание RMI-сервера

Код сервера состоит из интерфейса и класса. Интерфейс определяет методы, которые могут быть вызваны с клиента. В сущности, интерфейс определяет клиентское представление удаленного объекта. Класс обеспечивает реализацию [5].

Для создания удаленного интерфейса необходимо щелкнуть правой кнопкой мыши на пакет ru.eltech.rmi, выбрать New=> Interface и задать имя

интерфейса – Server. Данный интерфейс обеспечивает подключение между клиентом и сервером:

```
package ru.eltech.rmi;
import java.rmi.*;
public interface Server extends Remote {
public static final int PORT = 1099;
public static final String NAME = "my_server";
public void check(String str) throws RemoteException;
}
```

Для реализации удаленного интерфейса необходимо щелкнуть правой кнопкой мыши на пакет ru.eltech.rmi, выбрать New=> Class и задать имя класса – ServerImpl. Для установки подлинности клиента RMI необходимо создать файл политики безопасности, который содержит все требуемые разрешения, по методике из [6]. Класс ServerImpl реализует интерфейс Server, тем самым проводя аналогичное действие с удаленным объектом. Кроме того, данный класс содержит остальную часть кода, представляющую собой программу сервера. Программа содержит метод main(), который создает экземпляр удаленного объекта, записывает его в регистр RMI и настраивает менеджер безопасности [5]:

```
package ru.eltech.rmi;
import java.rmi.*;
public class ServerImpl implements Server {
private final String HELLO = "Hello, ";
private final static String SERVER_WAIT = "Server is waiting for connection...";
@Override
public void check(String str) throws RemoteException {
System.out.println(HELLO + str);
}
public static void main(String[] args) throws Exception {
if (System.getSecurityManager() == null) {
System.setSecurityManager(new SecurityManager());
}
Registry reg = LocateRegistry.createRegistry(PORT);
```

```
Server srv = new ServerImpl();
Server stub = (Server) UnicastRemoteObject.exportObject(srv, PORT);
reg.rebind(NAME, stub);
System.out.println(SERVER_WAIT);
} }
```

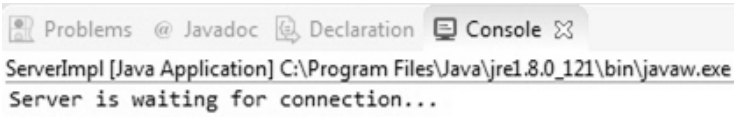
1.3.3 Создание программы клиента

Щелкнуть правой кнопкой мыши на пакет ru.eltech.rmi и выбрать New=> Class. Задать имя интерфейса – Client:

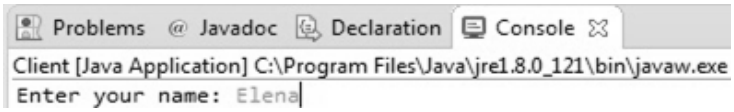
```
import java.rmi.*;
import java.util.Scanner;
public class Client{
private final String ENTER_NAME = "Enter your name: ";
private final static int CALL_ID = 0;
private Scanner in;
private void callRemote(int id){
try{
Registry reg = LocateRegistry.getRegistry(Server.PORT);
Server srv = (Server)reg.lookup(Server.NAME);
in = new Scanner(System.in);
System.out.print(ENTER_NAME);
String str = in.nextLine();
srv.check(str);
}catch(Exception ex){
ex.printStackTrace();
}
}
public static void main(String[] args) {
Client c = new Client();
c.callRemote(CALL_ID);
}
}
```

1.3.4 Компиляция и выполнение примера

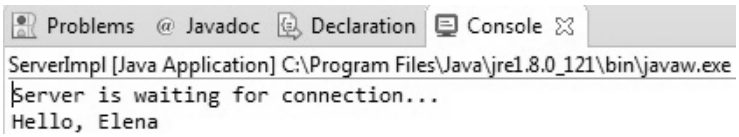
Классы ServerImpl и Client содержат метод main() и являются самостоятельными приложениями, которые могут быть запущены на отдельных компьютерах. В рассматриваемом случае роль клиента и сервера выполняет один компьютер. В Eclipse необходимо скомпилировать файлы Server.java, ServerImpl.java и Client.java. Затем запустить серверное и клиентское приложения, результаты работы которых представлены ниже (Рис. 1.2 а, б, в). На рис. 1.2. представлено окно консоли: а – сервера после запуска; б – клиента при вводе данных; в – сервера после ввода данных в клиентском приложении



а



б



в

Рис. 1.2.

Сервер находится в режиме ожидания подключения клиента. Запрашиваемая информация вводится в окно консоли клиента после его запуска. Результат отображается в окне консоли сервера при нажатии на кнопку Enter.

1.4 Контрольные вопросы

1. Каким образом происходит экспорт удаленного объекта?
2. Какие действия выполняются в серверной части приложения RMI?
3. Какая информация хранится в реестре RMI?
4. Какую роль выполняет заглушка в приложении RMI?

5. Продемонстрируйте реализацию методов, в которых выполняется расчет необходимых величин согласно варианту задания.

2 РАЗРАБОТКА КЛИЕНТ-СЕРВЕРНОГО ПРИЛОЖЕНИЯ С ПРИМЕНЕНИЕМ СОКЕТОВ

Цель работы: изучение теоретического материала о взаимодействии клиента и сервера с использованием сокетов, формирование практических навыков разработки клиент-серверного медицинского приложения на языке программирования Java.

2.1 Общие сведения

В клиент-серверном приложении сервер реализует некоторую службу, используемую клиентом. Как правило, взаимодействие между клиентом

и сервером осуществляется по протоколу TCP, который предоставляет канал связи типа «точка-точка» с гарантией доставки данных. Соединение в данном случае устанавливается по следующему принципу. Прежде всего клиент создает сокет на своем конце канала связи. Далее сервер получает первоначальный клиентский запрос на определенный номер порта и создает новый сокет на своем конце канала связи для осуществления взаимодействия с данным клиентом. Под взаимодействием понимается запись пересылаемых данных в сокет и чтение приходящих данных из него [2], [4], [15].

Сокет – это абстрактное понятие, которое используется для обозначения одной из двух конечных точек двунаправленного канала связи между двумя программами (сервером и клиентом), работающими в сети. Под конечной точкой понимают комбинацию IP-адреса и номера порта. Каждое TCP-соединение может быть однозначно определено посредством двух сокетов [1], [16]. Примитивы сокетов для стека протоколов Transmission Control Protocol / Internet Protocol (TCP/IP) приведены в табл. 2.1 [4], [17].

Схема взаимодействия клиента и сервера показана ниже (Рис. 2.1).

Сервер, как правило, последовательно выполняет первые четыре примитива (таблица 2.1). Процесс при вызове примитива Socket создает новый сокет для некоторого транспортного протокола. При этом локальная ОС резервирует ресурсы для размещения приходящих и отправляемых по данному протоколу сообщений.

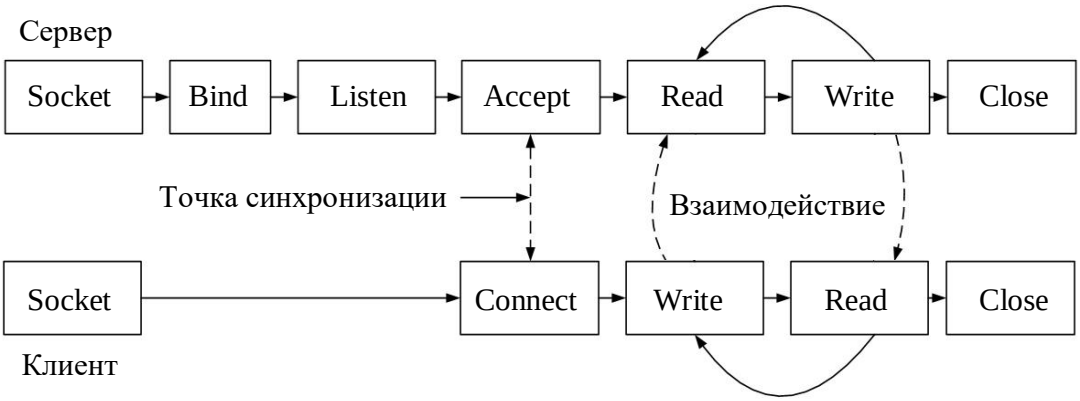


Рис. 2.1.

Примитив Bind осуществляет привязку локального адреса к созданному сокету. Примитив Listen требует от локальной ОС резерва буфера для максимального количества соединений, которые вызывающий процесс планирует поддерживать. Примитив Accept блокирует вызывающий процесс до прихода запроса на соединение. При получении запроса локальная ОС создает новый сокет, свойства которого аналогичны базовому, и возвращает его вызывающему процессу [17].

Таблица 2.1

Примитив	*Описание*
Socket	Создать сокет
Bind	Назначить сокету локальный адрес (IP-адрес и номер порта)
Listen	Обозначить готовность к установке соединения
Accept	Заблокировать вызывающую сторону до прибытия запроса о соединении
Connect	Совершить попытку установки соединения
Send	Послать данные через соединение (можно использовать Write)
Receive	Получить данные из соединения (можно использовать Read)
Close	Разорвать соединение

Со стороны клиента процесс описывается следующим образом. Сокет создается с помощью примитива Socket. Явная привязка сокета к локальному адресу не требуется, так как ОС динамически выделяет порт при установке соединения. Примитив Connect заставляет вызывающий процесс указать

адрес транспортного уровня, на который необходимо отправить запрос на соединение. Клиент блокируется до установки соединения. После установки соединения сервер и клиент обмениваются информацией с использованием примитивов Write и Read. При использовании сокетов возможно симметричное закрытие соединения, инициатором которого может являться как клиент, так и сервер. Оно выполняется путем вызова примитива Close [17].

2.2 Задание

Разработать консольное клиент-серверное приложение с использованием сокетов согласно номеру варианта, представленному в табл. 2.2. В данном приложении используются значения атрибутов из набора данных по заболеваниям сердца, который доступен для скачивания по ссылке https://www.kaggle.com/ronitf/heart-disease-uci. Два атрибута – возраст (годы) и пол (1 = мужской; 0 = женский) – являются общими для всех вариантов. Остальные атрибуты, а также данные для вывода в консоль представлены ниже (Таблица 2.2). Данными для ввода в консоль клиента для 1-6 вариантов являются возрастная группа (от 30 до 70 лет с шагом 10) и пол, для 7-8 вариантов – пол. В первом варианте необходимо предусмотреть ввод порядкового номера пациента из набора данных.

Таблица 2.2

№ варианта	*Атрибут*	*Данные для вывода в консоль*
1	Артериальное давление в состоянии покоя (в мм рт. ст. при поступлении в больницу)	Среднее значение артериального давления для возрастной группы определенного пола; категория артериального давления и степень артериальной гипертензии конкретного пациента
2	Холестерин в сыворотке крови (мг/дл)	Уровень холестерина в сыворотке крови для каждого пациента возрастной группы определенного пола; количество пациентов с одинаковым уровнем холестерина в сыворотке крови в возрастной группе

Продолжение таблицы 2.2

№ варианта	Атрибут	Данные для вывода в консоль
3	Максимальная частота сердечных сокращений	Наибольшее и наименьшее значения максимальной частоты сердечных сокращений в возрастной группе определенного пола; диапазоны значений пульса по пяти уровням интенсивности тренировок для найденных значений максимальной частоты сердечных сокращений
4	Депрессия сегмента ST, вызванная физическими упражнениями по сравнению с отдыхом	Процент пациентов с депрессией сегмента ST менее 1 мм, от 1 до 4 мм, более 4 мм в возрастной группе определенного пола; сравнение найденных значений со значениями из той же возрастной группы другого пола
5	Тип боли в грудной клетке (1 = типичная стенокардия; 2 = атипичная стенокардия; 3 = неангинальная боль; 4 = бессимптомная)	Процент пациентов с типичной и атипичной стенокардиями в возрастной группе определенного пола; сравнение найденных значений со значениями из соседних возрастных групп того же пола
6	Наклон сегмента ST при пиковой нагрузке (1 = восходящий; 2 = плоский; 3 = нисходящий)	Процент пациентов с восходящим и нисходящим наклоном сегмента ST при пиковой нагрузке в возрастной группе определенного пола; сравнение найденных значений со значениями из соседних возрастных групп того же пола
7	Результаты электрокардиографии в состоянии покоя (0 = нормальное; 1 = наличие ST-T; 2 = гипертрофия)	Количество пациентов с наличием ST-T и гипертрофией во всех возрастных группах определенного пола; средний возраст пациентов определенного пола с наличием ST-T и гипертрофии.

Окончание таблицы 2.2

№ варианта	Атрибут	Данные для вывода в консоль
8	Количество крупных сосудов (от 0 до 4), окрашенных флюороскопией	Количество пациентов с двумя и тремя крупными сосудами, окрашенными флюороскопией, во всех возрастных группах определенного пола; средний возраст пациентов определенного пола с наличием двух и трех крупных сосудов, окрашенных флюороскопией.

Класс сервера должен обеспечивать реализацию необходимых методов. При выполнении 1-4 вариантов задания рекомендуется использовать литературу [18-21] соответственно. При вводе чисел и символов с клавиатуры предусмотреть проверку на корректность их типов данных.

2.3 Порядок выполнения работы

В Eclipse выбрать пункт меню File>New>Java Project. Указать имя проекта – Lab_2_Socket, нажать на Next и затем на Finish. Нажать правой кнопкой мыши на src и выбрать New > Package. Задать имя нового пакета – ru.eltech.socket.

2.3.1 Создание программы сервера

Щелкнуть правой кнопкой мыши на пакет ru.eltech.socket и выбрать New > Class. Задать имя класса – Server:

```
import java.net.*;
import java.io.*;
public class Server {
private final static String SERVER_WAIT = "Server is waiting for connection...";
private final static String CLIENT_CONNECT = "Client connected to the server";
private final static String AND = " and ";
private final static String CLIENT_SEND = "Client sent these numbers: ";
```

```
private final static String SERVER_SEND = "Server is sending the maximum number back...";
private final static String SERVER_WAIT_NEW_DATA = "Server is waiting for new data...";
private final static String CLOSE = "Connection with the client is closed";
private final static int PORT = 6666;
public int max(int fn, int sn) {
return Math.max(fn, sn);
}
public static void main(String[] ar) {
Server s = new Server();
try {
ServerSocket ss = new ServerSocket(PORT);
System.out.println(SERVER_WAIT);
Socket socket = ss.accept();
System.out.println(CLIENT_CONNECT);
DataInputStream in = new DataInputStream(socket.getInputStream());
DataOutputStream out = new DataOutputStream(socket.getOutputStream());
int x = 0;
int y = 0;
x = in.readInt();
y = in.readInt();
System.out.println(CLIENT_SEND + x + AND + y);
System.out.println(SERVER_SEND);
out.writeInt(s.max(x, y));
out.flush();
System.out.println(SERVER_WAIT_NEW_DATA);
System.out.println();
in.close();
out.close();
ss.close();
socket.close();
System.out.println(CLOSE);
```

```
}
catch (Exception x) {
x.printStackTrace();
} }
}
```

2.3.2 Создание программы клиента

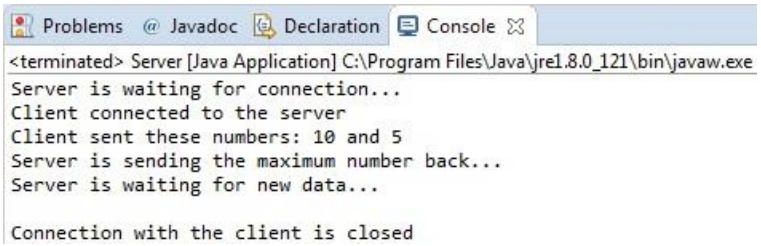
Щелкнуть правой кнопкой мыши на пакет ru.eltech.socket и выбрать New > Class. Задать имя класса – Client:

```
import java.net.*;
import java.io.*;
import java.util.Scanner;
public class Client {
private final static int PORT = 6666;
private final static String ADDRESS = "127.0.0.1";
private final static String CONNECTION_SUCCESSFUL = "Connection to the server was successful";
private final static String FIRST_NUMBER = "Type the first number: ";
private final static String SECOND_NUMBER = "Type the second number:";
private final static String MAXIMUM_NUMBER = "Maximum number: ";
private final static String CONNECTION_CLOSE = "Connection with the server is closed";
private static Scanner id;
public static void main(String[] ar) {
int serverPort = PORT;
String address = ADDRESS;
try {
InetAddress ipAddress = InetAddress.getByName(address);
Socket socket = new Socket(ipAddress, serverPort);
System.out.println(CONNECTION_SUCCESSFUL);
DataInputStream in = new DataInputStream(socket.getInputStream());
DataOutputStream out = new DataOutputStream(socket.getOutputStream());
int x = 0;
```

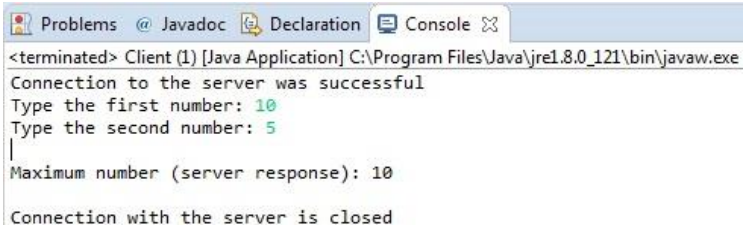
```
int y = 0;
int z = 0;
id = new Scanner(System.in);
System.out.print(FIRST_NUMBER);
x = id.nextInt();
System.out.print(SECOND_NUMBER);
y = id.nextInt();
System.out.println();
out.writeInt(x);
out.writeInt(y);
out.flush();
z = in.readInt();
System.out.println(MAXIMUM_NUMBER + z);
System.out.println();
in.close();
out.close();
socket.close();
System.out.println(CONNECTION_CLOSE);
} catch (Exception x) {
x.printStackTrace();
} } }
```

2.3.3 Компиляция и запуск примера

Классы Server и Client содержат метод main( ) и являются самостоятельными приложениями, которые могут быть запущены на отдельных компьютерах. В рассматриваемом случае роль клиента и сервера выполняет один компьютер. В Eclipse необходимо скомпилировать файлы Server.java и Client.java. Затем запустить серверное и клиентское приложения, результаты работы которых представлены ниже (Рис. 2.2 а, б).



а



б

Рис. 2.2.

Сервер находится в режиме ожидания подключения клиента. Когда подключение произошло, в окнах консоли сервера и клиента отображаются соответствующие сообщения. Затем пользователь последовательно вводит два целых числа в окне консоли клиента. Данные числа отсылаются на сервер. Сервер выбирает максимальное число и отсылает его клиенту. Затем работа серверного и клиентского приложений завершается.

2.4 Контрольные вопросы

1. На основе какого протокола чаще всего осуществляется клиент-серверное взаимодействие?
2. Что такое сокет?
3. Для чего используется примитив Listen?
4. С помощью каких примитивов осуществляется обмен данными между клиентом и сервером?
5. Необходима ли явная привязка сокета к локальному адресу на стороне клиента?

3 РАЗРАБОТКА РАСПРЕДЕЛЕННОГО ПРИЛОЖЕНИЯ С ДОСТУПОМ К БАЗЕ ДАННЫХ

Цель работы: изучение теоретического материала о платформе JavaFX и встраиваемой библиотеке SQLite, формирование практических навыков разработки распределенного Graphical User Interface (GUI) приложения с использованием сокетов на языке программирования Java для доступа к базе данных (БД).

3.1 Общие сведения

JavaFX – это платформа на базе Java, которая обеспечивает создание мощного GUI для кросс-платформенных многофункциональных клиентских приложений. Она содержит богатый набор элементов управления, графический и медийный Application Programming Interface (API) с аппаратным ускорением графики. Последней версией JavaFX является JavaFX 8 [23], [24].

JavaFX API содержит более 30 пакетов: javafx.application, javafx.stage, javafx.scene и др. Основная метафора, воплощаемая JavaFX, называется подмостками (stage). По аналогии с театром на подмостках ставится сцена (scene). Подмостки определяют некоторое пространство, а сцена – то, что происходит в данном пространстве. Таким образом, подмостки являются контейнером для сцен, а сцена – контейнером для узлов, которые составляют данную сцену. Под узлами понимаются элементы GUI, например элементы управления или текст. Набор всех узлов в сцене создает граф сцены, который является деревом. Любое JavaFX-приложение имеет как минимум одни подмостки и одну сцену, включенную в данные подмостки. Подмостки и сцена инкапсулированы в JavaFX API посредством классов Stage и Scene соответственно. Любое JavaFX-приложение при запуске автоматически получает доступ к одним подмосткам. После создания сцены и добавления в нее элементов она устанавливается на подмостки [24], [25].

JavaFX-приложение должно наследоваться от класса Application, который упакован в javafx.application. Класс Application определяет три метода, которые могут быть переопределены, – init( ), start( ) и stop( ). Метод init( ) вызывается, когда приложение начинает выполняться и используется для осуществления различных инициализаций. Метод start( )

вызывается после метода init( ) и используется для создания и установки сцены. Когда приложение завершает работу, вызывается метод stop( ) [24], [25].

CSS – это язык, который используется для описания представления элементов GUI в приложении. Внешний вид JavaFX-приложений настраивается с использованием CSS, который предоставляет синтаксис для написания правил, устанавливающих визуальные свойства. Правило состоит из селектора и набора пар «свойство»-«значение». Селектор – это строка, которая идентифицирует элементы GUI, к которому применяются правила. Пара «свойство»-«значение» состоит из имени свойства и его значения, которые разделены двоеточием (:). Две пары «свойство»-«значение» разделены точкой с запятой (;). Набор пар «свойство»-«значение» заключается в фигурные скобки ({}), которым предшествует селектор [24], [26].

SQLite – это встраиваемая библиотека, реализующая автономную, безсерверную, автоматически конфигурируемую, реляционную БД SQL-типа. Поскольку данная библиотека является встраиваемой, она не имеет отдельного процесса сервера. Иначе говоря, SQLite находится внутри приложения, которое обслуживает, – ее код встроен в программу. Таким образом, клиент и сервер работают вместе в одном и том же процессе. Преимуществами такого подхода являются уменьшение накладных расходов, связанных с сетевыми вызовами; облегчение администрирования БД и упрощение развертывания приложения. SQLite производит запись и чтение напрямую в файл на диске. БД в полном объеме, включая таблицы, индексы, триггеры и представления, помещается в один файл кроссплатформенного формата [27]–[29].

SQLite имеет компактный размер, а именно 500 Кб, с учетом всех активированных функций. Если дополнительные функции не используются, то размер будет ниже 300 Кб. SQLite работает в случаях минимального пространства стека (4 Кб) и очень маленькой кучи (100 Кб). Данный факт предоставляет SQLite преимущество, когда речь идет о выборе БД для устройств с ограниченным объемом памяти, например мобильных телефонах [27].

SQLite предоставляет широкий спектр функций, к которым относят следующие. Во-первых, описанные в международном стандарте ANSI SQL-92 (большое подмножество, но не все): транзакции, представления, проверочные ограничения, внешние ключи, коррелированные подзапросы, сложные запросы и так далее. Во-вторых, имеющиеся в реляционных БД: триггеры, индексы, автоинкрементируемые столбцы и операторы LIMIT/OFFSET. В-третьих, считающиеся редкими или уникальными: БД, размещенные в памяти; динамическая типизация и разрешение конфликтов [28].

Код SQLite находится в открытом доступе, распространяется без лицензии и может быть использован как в частных, так и в коммерческих целях [27]–[29].

3.2 Задание

Разработать клиент-серверное GUI-приложение с использованием сокетов для работы с БД согласно номеру задания, представленному в таблице.

БД создается с использованием визуального инструмента DB Browser for SQLite. Она содержит три таблицы, каждая из которых имеет столбец с ограничением PRIMARY KEY. Таблицы связаны друг с другом. Количество столбцов в каждой таблице должно быть не менее пяти, количество используемых типов данных – не менее трех.

В GUI серверного приложения осуществляется просмотр содержимого таблиц из БД. Каждая таблица представлена на отдельной вкладке. В GUI клиентского приложения осуществляется три различных варианта поиска информации из данных таблиц согласно введенному пользователем значению (или значениям). Ввод значения производится в отдельном поле (если в поле разрешено вводить только цифры, необходимо предусмотреть соответствующую проверку). Вывод таблицы выполняется при нажатии на кнопку "Поиск". Сформированная таблица должна содержать поля из двух исходных таблиц как минимум (например, в задании 1 одним из вариантов поиска может быть вывод данных о поликлиниках с имеющимся у них количеством ампул конкретной вакцины).

Если значение в таблице не найдено, в клиентском приложении должно появляться сообщение об этом. Все запросы к БД осуществляются в серверном приложении.

Таблица

№ задания	*Заголовки таблиц*
1	Вакцины, поставщики вакцин, поликлиники
2	Льготные лекарственные препараты, рецепты, аптеки
3	Медицинское оборудование, отделения больницы, обслуживающий (оборудование) персонал
4	Клинический анализ крови, вакуумные пробирки, медсестры
5	МРТ, томографы, рентгенологи
6	Физиотерапевтические процедуры, филиалы медицинского центра, расписание
7	УЗИ-аппараты, плановое техническое обслуживание, сервисные центры
8	Травмы, ортезы, магазины средств реабилитации
9	Телемедицинская консультация, врач, лист назначений
10	Медицинское ПО, лицензии, отделения поликлиники
11	Районные станции скорой медицинской помощи, вызовы, автомобили скорой медицинской помощи
12	Профилактические осмотры, сотрудники, предприятия
13	Операции, квоты, медицинское учреждение
14	Диспансеризация, пациенты, осмотры
15	Сеть санаториев, виды услуг, санаторно-курортные карты

Кроме того, необходимо учитывать, что при закрытии серверного приложения в клиентском приложении должно появляться уведомление о возникшей ситуации. Затем клиентское приложение должно закрываться. Аналогичные действия необходимо выполнить при закрытии клиентского приложения. Если клиентское приложение запущено раньше серверного приложения, то следует вывести сообщение об ошибке подключения к серверу. Затем клиентское приложение должно закрываться.

3.3 Порядок выполнения работы

Установить плагин JavaFX для Eclipse, выполнив следующие действия. Запустить Eclipse, в раскрывшемся списке меню Help выбрать Install New Software. В открывшемся окне нажать на Add. В поле Name ввести

e(fx)clipse, в поле Location – http://download.eclipse.org/efxclipse/updates-nightly/site/ (текущей версией плагина является Eclipse e(fx)clipse 3.8.0, ссылка на сайте http://projects.eclipse.org/projects/technology.efxclipse/downloads). Нажать на OK. В дереве устанавливаемых компонент отметить только элементы, соответствующие вашей версии: e(fx)clipse-IDE. Пройти через мастера установки и перезагрузить Eclipse после запроса [30].

Перед запуском JavaFX-приложения необходимо подключить JavaFX библиотеки времени исполнения. В Eclipse в раскрывающемся списке меню Window выбрать Preferences и затем в появившемся окне – Java=>Installed JREs. Нажать на Add. В JRE type выбрать Standard VM и нажать на Next.

В поле JRE home указать путь к домашней директории Java (например, C:\Program Files\Java\jdk18.0.1) и нажать на Finish. Сделать данный JDK по умолчанию, поставив галочку на нем. Аналогичное действие выполнить в Compatible JREs, выбрав в дереве слева Java=>Installed JREs=>Execution Environments и затем JavaSE-18 в области справа от дерева.

3.3.1 Создание GUI-сервера

Скачать последнюю версию DB Browser for SQLite с сайта http://sqlitebrowser.org/. В Eclipse выбрать пункт меню File=>New=>Project=>JavaFX=>JavaFX Project. Указать имя проекта – Lab_3_DB, нажать на Next и затем на Finish. Нажать правой кнопкой мыши на src и выбрать New=>Package. Задать имя нового пакета – ru.eltech.db.

Скачать последнюю версию SQLite JDBC Driver и поместить данный файл в папку проекта C:\Users\<User Name>\workspace\Lab_3_ DB.

Добавить библиотеку (JAR-файл) к пути к классу проекта. В Eclipse щелкнуть правой кнопкой мыши на имени проекта и из раскрывающегося списка выбрать Build Path=>Configure Build Path. В открывшемся окне справа перейти на вкладку Libraries и нажать на Add External JARs. Выбрать ранее скачанный файл и нажать на OK.

Для создания GUI-сервера с целью просмотра содержимого таблицы из БД необходимо выполнить следующие действия. Щелкнуть правой кнопкой мыши на пакет ru.eltech.db и выбрать New > Class. Задать имя класса – Server. В данном классе представлена реализация методов для подключения к БД, получения содержимого таблицы из БД, поиска содержимого одного из

столбцов таблицы по известному значению из другого столбца данной таблицы:

```
package ru.eltech.db;
import java.sql.*;
import javafx.*;
public class Server extends Application {
private final static String URL = "jdbc:sqlite:D:/SQLite/db/huawei.db";
private final String SERVER_TITLE = "Server";
private final static String SQL_SELECT_SERVERS = "SELECT * from
FusionServers";
private final static String TAB_SERVERS =      "Fusion Servers";
private TableView tableViewServers;
private ObservableList<ObservableList> data;
public static Connection connect() {
Connection conn = null;
try {
conn = DriverManager.getConnection(URL);
} catch (SQLException e) {
e.printStackTrace();
}
return conn;
}
public static void main(String[] args) {
launch(args);
}
private void initialize(Stage stage) {
stage.setTitle(SERVER_TITLE);
Group root = new Group();
Scene scene = new Scene(root);
TabPane tabPane = new TabPane();
tabPane.setTabClosingPolicy(TabClosingPolicy.UNAVAILABLE);
BorderPane mainPane = new BorderPane();
Tab tabServers = new Tab();
tabServers.setText(TAB_SERVERS);
```

```
VBox vBoxServers = new VBox();
tableViewServers = new TableView();
buildData(SQL_SELECT_SERVERS, tableViewServers);
vBoxServers.getChildren().add(tableViewServers);
tabServers.setContent(vBoxServers);
tabPane.getTabs().add(tabServers);
mainPane.setCenter(tabPane);
mainPane.prefHeightProperty().bind(scene.heightProperty());
mainPane.prefWidthProperty().bind(scene.widthProperty());
root.getChildren().add(mainPane);
stage.setScene(scene);
stage.show();
}
public void buildData(String sql, TableView tableView) {
Connection c;
data = FXCollections.observableArrayList();
ResultSet rs = null;
try {
c = connect();
rs = c.createStatement().executeQuery(sql);
for (int i=0; i<rs.getMetaData().getColumnCount(); i++) {
final int j = i;
TableColumn col = new TableColumn(rs.getMetaData().
getColumnName(i+1));
col.setCellValueFactory(new Callback<CellDataFeatures
<ObservableList,String>,ObservableValue<String>>() {
public ObservableValue<String> call(CellDataFeatures
<ObservableList, String> param) {
return new SimpleStringProperty(param.getValue().get(j).toString());
} });
tableView.getColumns().addAll(col);
}
while(rs.next()) {
ObservableList<String> row = FXCollections.observableArrayList();
```

```
for(int i=1 ; i<=rs.getMetaData().getColumnCount(); i++) {
row.add(rs.getString(i));
}
data.add(row);
}
tableView.setItems(data);
}
catch(Exception e) {
e.printStackTrace();
 System.out.println("Error on Building Data");
}
}
public String selectProcessors(String model){
String sql = "select Processors from FusionServers where Model=\""+ model
+ "\"";
String processors = null;
try (Connection conn = Test.connect();
Statement stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery(sql)) {
while (rs.next()) {
processors = rs.getString("Processors");
}
} catch (SQLException e) {
e.printStackTrace();
}
return processors;
}
@Override
public void start(Stage stage) {
initialize(stage);
}
}
```

3.3.2 Создание css-файла

В Eclipse щелкнуть правой кнопкой мыши на пакет ru.eltech.javafx и выбрать New > File. Задать имя файла – style.css. Файл фона необходимо предварительно сохранить в C:\Users\<User Name>\workspace\Lab_3_DB \bin\ru\eltech\javafx. Установка фонового изображения, стили надписей и кнопок представлены в файле style.css:

```
.root {
-fx-background-image: url("blue_background.jpg");
}
.label {
-fx-font-size: 12px;
-fx-font-weight: bold;
-fx-text-fill: #333333;
-fx-effect: dropshadow( gaussian , rgba(255,255,255,0.5) , 0,0,0,1 );
}
.button {
-fx-text-fill: #333333;
-fx-font-weight: bold;
-fx-background-color: linear-gradient(#a0e6f2, #0ad0ea);
-fx-effect: dropshadow( three-pass-box , rgba(0,0,0,0.6) , 5, 0.0 , 0 , 1 );
}
```

3.4 Содержание отчета

Отчет должен содержать: титульный лист; указание цели работы; основные теоретические положения; вариант задания; краткое описание выполнения работы, сопровождаемое рисунками; полученные результаты; выводы.

3.5 Контрольные вопросы

1. Что такое узел в терминах JavaFX?
2. Дайте описание методам init(), start() и stop().
3. Что такое CSS?
4. Что представляет собой SQLite?
5. Перечислить несколько функций, которые имеются в реляционных БД и поддерживаются SQLite.

СПИСОК ЛИТЕРАТУРЫ

1. Ciubotaru B., Muntean G. M. Advanced network programming – principles and techniques: network application programming with Java. UK, London: Springer Science & Business Media, 2013.

1. Graba, J. An Introduction to network programming with Java. USA, New York: Springer, 2006.

2. Remote Method Invocation Specification. Introduction and Java Distributed Object Model // Oracle and/or its affiliates. 2022. URL: https://docs.oracle.com/en/java/javase/17/docs/specs/rmi/index.html (дата обращения: 01.07.2022).

3. Sharan K. Beginning Java 8 APIs, extensions and libraries: Swing, JavaFX, JavaScript, JDBC and network programming APIs. USA, New York: Apress, 2014.

6. RMI System Overview // Oracle and/or its affiliates. 2022. URL: https://docs.oracle.com/en/java/javase/17/docs/specs/rmi/arch.html (дата обращения: 01.07.2022).

7. Дубаков А. А. Сетевое программирование: учеб. пособие. СПб: НИУ ИТМО, 2013.

8. Учебно-методическое пособие по оценке физического развития детей: учеб.-метод. пособие / под ред. проф. Е.М. Булатовой. СПб.: СПбГПМУ, 2019.

9. Практические навыки по акушерству и гинекологии: учеб. пособие; 2-е изд., перераб. и доп., с элементами симуляционного обучения / Л. И. Трубникова [и др.]; под ред. проф. Л. И. Трубниковой. Ульяновск: УлГУ, 2015.

10. Анестезиология и интенсивная терапия: Практическое руководство / Б.Р. Гельфанд [и др.]; под общ. ред. Б.Р. Гельфанда. М.: Литтерра, 2006. - 576 с.

11. Буйкова О. М., Булнаева Г. И. Функциональные пробы в лечебной и массовой физической культуре: учеб. пособие. Иркутск: ИГМУ, 2017.

12. Черноземов В.Г., Афанасенкова Н.В., И.А. Варенцова И.А. Методы физиологического исследования человека: учеб.-метод. пособие.

Архангельск: Северный (Арктический) федеральный ун-т им. М. В. Ломоносова, 2017.

13. Федеральные клинические рекомендации по диагностике и лечению железодефицитной анемии / А.Г. Румянцев [и др.]. М.: Стандартинформ, 2014.

14. Адаменко Е. И., Силивончик Н. Н. Оценка статуса питания: учеб.-метод. пособие. Минск: БГМУ, 2009.

15. Соколенко Г. В., Базлов С. Б. Расчет и назначение инфузионно-трансфузионной терапии: учеб.-метод. пособие. Краснодар: КубГМУ, 2013.

16. Lesson: all about sockets // Oracle and/or its affiliates. 20221. URL: https://docs.oracle.com/javase/tutorial/networking/sockets/index.html (дата обращения: 01.07.2022).

17. What is a socket? // Oracle and/or its affiliates. 2022. URL: https://docs.oracle.com/javase/tutorial/networking/sockets/definition.html (дата обращения: 01.07.2022).

18. Таненбаум Э., ван Стеен М. Распределенные системы. Принципы и парадигмы. СПб.: Питер, 2003.

19. 2018 ЕОК/ЕОАГ Рекомендации по лечению больных с артериальной гипертензией/ Williams B. [и др.]. // Российский кардиологический журнал. – 2018. – №12. – с. 143-228.

20. Биохимический практикум: учеб.-метод. пособие / Ф.Х. Камилов [и др.]. Уфа: Изд-во ГБОУ ВПО БГМУ Минздрава России, 2014.

21. Бодин О.Н., Полосин В. Г., Балахонова С. А. Прогнозирование максимальной частоты сердечных сокращений для расчета интенсивности физических нагрузок // Измерение. Мониторинг. Управление. Контроль. – 2013. – №1. – с. 50-54.

22. Тавровская Т.В. Велоэргометрия. Пособие для врачей. Санкт-Петербург: Инкарт, 2007.

23. JavaFX overview // Oracle and/or its affiliates. 2015. URL: http://docs.oracle.com/javase/8/javafx/get-started-tutorial/jfxoverview.htm#JFXS T784 (дата обращения: 01.07.2022).

24. Sharan K. Learn JavaFX 8: Building user experience and interfaces with Java 8. USA, New York: Apress, 2015.

25. Schildt H. Introducing JavaFX 8 programming. USA, New York: McGraw-Hill Education, 2015.

26. DiMarzio J. F. JavaFX. A beginners guide. USA, New York: McGraw Hill Professional, 2011.

27. E(fx)clipse. JavaFX tooling and runtime for Eclipse and OSGi // The Eclipse Foundation. 2017. URL: http://www.eclipse.org/efxclipse/install.html#for-the-ambitious (дата обращения: 01.07.2022).

28. About SQLite. URL: http://sqlite.org/about.html (дата обращения: 01.07.2022).

29. Owens M., Allen G. The definitive guide to SQLite, Second edition. USA, New York: Apress, 2010.

30. Kreibich J. Using SQLite. USA, CA, Sebastopol: O'Reilly Media, Inc., 2010.

ОГЛАВЛЕНИЕ