**ЛАБОРАТОРНАЯ РАБОТА №2**. **Работа на основе сокетных соедиений**

**Цель работы.** Познакомиться с техникой использования сокетных классов Java.

**Краткое теоретическое содержание**. В теоретическом материале представлены сведения по работе с сокетными классами на базе протоколов TCP и UDP. В разрабатываемом ниже приложении «клиент-сервер» используются два параллельно выполняющихся потока: **поток-сервер** и **поток-клиент**.

Для установления связи между клиентом и сервером нужно создать сокетное соединение как на серверной, так и на клиентской стороне. На стороне клиента это делается так

class clientThread extends Thread{

DataInputStream dis=null;

Socket s=null;

public clientThread()

{ try{

s=new Socket("127.0.0.1",3001);

. . . и т.д.

Здесь объявлена переменная типа Socket, которая создается в команде

s=new Socket("127.0.0.1",3001);

На стороне сервера указанные действия выполняются в главном методе потока run:

ServerSocket server;

String amountstring;

static int amount=200;

public void run(){

try

{

server= new ServerSocket(3001);

}

catch(Exception e)

{ System.out.println("ERRSOCK+"+e);

}

Подключение клиента к серверу:

s=server.accept();

В данной лабораторной работе сервер передает клиенту данные о банковском счете, представленные случайными целыми числами. На стороне сервера это делается таким образом

class Account extends Thread{

ServerSocket server;

String amountstring;

static int amount=200;

public void run(){

try

{

server= new ServerSocket(3001); //Номер сокета

}

catch(Exception e)

{ System.out.println("Ошибка соединения+"+e);

}

while(true)

{ Socket s=null;

try{

s=server.accept(); //ожидание соединения с клиентом

.0

catch(Exception e)

{System.out.println("Ошибка"+e);}

try

{PrintStream ps=new PrintStream(s.getOutputStream()); //PrintStream предназначен для текстового вывода

int amountcur=((int)(Math.random()\*1000)); //отрицательный вклад −  снятие части денег со счета

if (Math.random()>0.5)

amount-=amountcur;

else

amount+=amountcur;

Integer x=new Integer(amount);

amountstring=x.toString();

ps.println("Account:"+amountstring);

//передача строки клиенту

ps.flush();

s.close();//Сокетное соединение закрывается

}

catch(Exception e)

{System.out.println("Ошибка "+e);

}

}

Поток вывода для сервера реализуется через переменную ps:

PrintStream

ps=new PrintStream(s.getOutputStream());

Объектная переменная ps предоставляет методы вывода, например

ps.println("Account:"+amountstring);

В команде

amount=((int)(Math.random()\*1000));

на счет клиента добавляется случайная величина с помощью метода

random, генерирующего случайные числа от 0 до 1.

Изменение суммы в переменной amount выполняется путем внесения или снятия случайной величины со счета:

int amountcur=((int)(Math.random()\*1000));

if (Math.random()>0.5)

amount-=amountcur;

else

amount+=amountcur;

Серверная часть реализована как приложение Java на основе формы c главным методом main :

public static void main(String args[])

{

serv f=new serv();

f.resize(400,400);

f.show();

new Account().start();

}

Базовым классом сервера является класс serv:

public class serv extends Frame{

public boolean handleEvent(Event evt)//Используется обработчик //событий ранних версий Java

{

if (evt.id==Event.WINDOW\_DESTROY)//Закрыть приложение

{System.exit(0);}

return super.handleEvent(evt);

}

public boolean mouseDown(Event evt,int x,int y)//Обработчик //события от мыши

{

new clientThread().start();//Запуск потока клиента

return(true);

}

public static void main(String args[])

{

serv f=new serv();

f.resize(400,400);

f.show();

new Account().start();

}

}

Главный метод запускает поток-сервера

new Account().start();

Базовый класс приложения реализует обработку события закрытия окна:

public boolean handleEvent(Event evt)

{

if (evt.id==Event.WINDOW\_DESTROY)

{System.exit(0);}

return super.handleEvent(evt);

}

Тип события проверяется командой

if (evt.id==Event.WINDOW\_DESTROY)

Второй метод

public boolean mouseDown(Event evt,int x,int y)

{

new clientThread().start();

return(true);

}

обрабатывает щелчок мышью в окне сервера (по щелчку в окне запускается клиент):

new clientThread().start();

И сервер, и клиент реализованы как отдельные потоки. Поток клиента такой

class clientThread extends Thread{

DataInputStream dis=null;

Socket s=null;

public clientThread()

{ try{

s=new Socket("127.0.0.1",2525);

dis= new DataInputStream(s.getInputStream());

}

catch(Exception e)

{ System.out.println("Ошибка: "+e);

}}

public void run()

{ while (true)

{

try

{sleep(100);

}

catch(Exception er)

{System.out.println("Ошибка "+er);

}

try{

String msg=dis.readLine();

if(msg==null)

break;

System.out.println(msg);

}

catch(Exception e)

{System.out.println("ERRORR+"+e);

}

}

}

}

Клиент инициализируется таким образом

DataInputStream dis=null;

Socket s=null;

public clientThread()

{ try{

s=new Socket("127.0.0.1",2525);

dis= new DataInputStream(s.getInputStream());

}

Клиент пытается прочитать данные из сокета Socket("127.0.0.1",3001); Именно туда пишет данные сервер. Заметим, что клиент должен указать сетевой адрес компьютера, где расположен сервер (этот адрес называется IP-адресом). Клиент использует объектную переменную dis для чтения данных из сокета в методе run:

public void run()

{ while (true)

{

try

{sleep(100); //клиент выполняет попытку чтения из сокета каждые //100 милисекунд

}

catch(Exception er)

{System.out.println("Ошибка "+er);

}

try{

String msg=dis.readLine();// Клиент пытается прочитать строку //из сокета

if(msg==null)

break;

System.out.println(msg); //прочитанная строка выводится на //консоль

}

### Задание (общее)

1. Завершите приложение работоспособной программой, в которой используются описанные поток сервера и поток клиента.
2. Создайте свой вариант программы на основании следующих данных.
   1. Написать приложение для двух клиентов, которые работают с одним счетом. Каждому клиенту соответствует свой поток.
   2. Cоздать приложение клиент-сервер с некоторым визуальным интерфейсом, который выполняет запуск клиента по нажатию кнопки и отражает состояние счета в текстовом поле.

***Порядок выполнения работы***

1. Изучить теоретическую часть.
2. Получить индивидуальное задание у преподавателя из прилагаемого списка.
3. Создать и отладить приложение .
4. Написать отчет.
5. Защитить работу.

***Индивидуальные задания***

**Вариант 1.**

Запустить клиент и сервер в разных проектах. Потоки не использовать.

**Вариант 2.**

Создать 2 клиента через потоки. Один клиент только добавляет и снимает деньги со счета, второй только снимает.

**Вариант 3.**

Имеется один клиент, но 2 банковских счета. Снимаются деньги с того счета, где текущая сумма больше.

**Контрольные вопросы**.

1. Объясните назначение протоколов TCP, UDP. В чем их отличие?
2. С помощью каких объектов и каких методов данные отсылаются по сети?
3. Как реализовать пересылку данных в обратном направлении (от клиента к серверу)?
4. Как определить сетевой адрес вашего компьютера.