МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

*Лабораторная работа №5 по дисциплине «Технология программирования»*

*Вариант 2*

Группа: АВТ-610

Студенты: Князев Р.И., Дунаев Н.Ю.

Преподаватель: Васюткина И. А.

Новосибирск 2018

**Темы:**

***Сетевые «клиент-серверные» приложения***

**Задание:**

1. Изучить особенности реализации сетевых приложений в Java.
2. Доработать программу, созданную в лабораторной работе № 5:
3. создать отдельное консольное приложение, которое будет играть роль TCP-сервера. Клиентом будет приложение, которое было создано в предыдущих работах;
4. TCP-сервер должен ожидать подключения клиентов и выдавать вновь подключенному клиенту список уже подключенных. Необходимо также внести изменения в интерфейс клиента, так чтобы в панели управления отображался список всех подключенных к серверу клиентов. При отсоединении клиентов или при подключении новых список должен обновляться;
5. запрограммировать специальное взаимодействие по TCP с другими клиентами через сервер по варианту;

***Вариант 2***

Реализовать возможность отправить N случайных объектов из текущей симуляции другому подключенному клиенту.

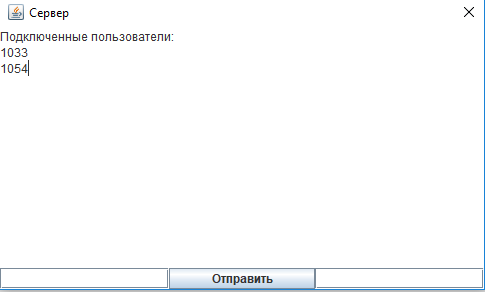
***Описание структуры классов:***

Для реализации сервера были созданы классы:

* Main – сервер
* OneJabber – класс, описывающий взаимодействие с конкретным клиентом

В программу из предыдущих лабораторных работ были добавлены:

* ClientThread – класс, реализующий обработку информации, полученной от сервера.
* Кнопка «Сервер», открывающая окно для взаимодействия с сервером.



Окно содержит текстовую область, содержащую информацию о подключенных пользователях, поля для ввода ID пользователя и кол-ва пчел для передачи, а также кнопку «Отправить».

Кнопка «Отправить» имеет слушателя, в теле которого реализована отправка пчел другому пользователю.

***Описание функционала программы:***

**Клиент:**

// отправка команды, айди, кол-ва пчел, затем типа пчелы и всех ее свойств.

**wind**.**sendBut**.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **try** {  
 **out**.writeInt(1);  
 **out**.writeInt(**idToSend**);  
 **out**.writeInt(**numBee**);  
 **for** (**int** i = 0; i < **numBee**; ++i)  
 {  
 Bee bb = **mas**.get(i);  
 **if** (bb **instanceof** BeeWorker)  
 **out**.writeInt(1);  
 **else  
 out**.writeInt(2);  
 **out**.writeDouble(bb.**x**);  
 **out**.writeDouble(bb.**y**);  
 **out**.writeInt(bb.**uniqueKey**);  
 **out**.writeDouble(bb.**liveTime**);  
 **out**.writeDouble(bb.**bornTime**);  
 **if** (bb **instanceof** BeeWorker) {  
 BeeWorker bw = (BeeWorker)bb;  
 **out**.writeDouble(bw.**startX**);  
 **out**.writeDouble(bw.**startY**);  
 }  
 }  
 } **catch** (IOException e1) {  
 e1.printStackTrace();  
 }  
 }  
});  
  
**wind**.**servIdToSend**.addFocusListener(**new** FocusAdapter() {  
 @Override  
 **public void** focusLost(FocusEvent e) {  
 **idToSend** = Integer.*parseInt*(**wind**.**servIdToSend**.getText());  
 }  
});  
  
**wind**.**servSendBeeCount**.addFocusListener(**new** FocusAdapter() {  
 @Override  
 **public void** focusLost(FocusEvent e) {  
 **numBee** = Integer.*parseInt*(**wind**.**servSendBeeCount**.getText());  
 }  
});

**public class** ClientThread **extends** Thread {  
 DataInputStream **in**;  
 Window **w**;  
 **boolean isEnd** = **false**;  
 **int size**;  
 ArrayList<Bee> **mas**;  
  
 ClientThread(DataInputStream inputStream, Window wind, ArrayList<Bee> maas)  
 {  
 **mas** = maas;  
 **w** = wind;  
 **in** = inputStream;  
 start();  
 }  
  
  
 @Override  
 **public void** run()  
 {

**while**(**true**)  
 {  
 **try** {  
 **if**(!**isEnd**) {  
 **int** k = **in**.readInt();  
 **switch** (k) {  
 **case** 1:  
 **size** = **in**.readInt();  
 String str = **""**;  
 **for** (**int** i = 0; i < **size**; i++) {  
 str += String.*valueOf*(**in**.readInt());  
 str += **"\n"**;  
 }  
 **w**.**servTxtArea**.setText(**"Подключенные пользователи: \n"** + str);  
 **break**;  
  
 **case** 2:  
 **size** = **in**.readInt();  
 System.***out***.println(**size**);  
 **for** (**int** i = 0; i < **size**; ++i)  
 {  
  
 Bee bb;  
 System.***out***.println(**"1"**);  
 **int** type = **in**.readInt();  
 System.***out***.println(**"2"**);  
 **double** x = **in**.readDouble();  
 System.***out***.println(**"1"**);  
 **double** y = **in**.readDouble();  
 **int** key = **in**.readInt();  
 System.***out***.println(**"2"**);  
 **double** liveTime = **in**.readDouble();  
 **double** bornTime = **in**.readDouble();  
  
 **if** (type == 1) {  
 **double** startX = **in**.readDouble();  
 **double** startY = **in**.readDouble();  
 bb = **new** BeeWorker(x, y, key, liveTime, bornTime);  
 BeeWorker bw = (BeeWorker)bb;  
 bw.**startX** = startX;  
 bw.**startY** = startY;  
 }  
 **else** bb = **new** BeeTruten(x, y, key, liveTime, bornTime);  
 **mas**.add(bb);  
  
 }  
 **break**;  
 }  
 }  
 **else  
 return**;  
 }**catch** (IOException e){  
  
 }  
 }  
 }  
}

**Сервер:**

**public class** Main {  
 **static** Vector<OneJabber> *mas* = **new** Vector<>();  
 **static final int *PORT*** = 6996;  
 **static** HashSet<Integer> *idMas* = **new** HashSet<>();  
 **public static void** main(String[] args) **throws** IOException  
 {  
 ServerSocket s = **new** ServerSocket(***PORT***);  
 System.***out***.println(**"Server Started"**);  
 **int** ID = 0;  
 **try** {  
 **while** (**true**) {  
 Socket socket = s.accept();  
 Random rand = **new** Random();  
 **boolean** added = **false**;  
 **while** (!added) {  
 ID = rand.nextInt(150) + 1000;  
 **if** (!*idMas*.contains(ID))  
 added = *idMas*.add(ID);  
 }  
 **try** {  
 OneJabber oneJabber = **new** OneJabber(socket, ID, *mas*);  
 *mas*.add(oneJabber);  
 **for**(**int** i = 0; i < *mas*.size(); i++)  
 *mas*.get(i).sendActive(*idMas*);  
 }  
 **catch** (IOException e) {  
 socket.close();  
 }  
 }  
 }  
 **finally** {  
 s.close();  
 }  
 }  
  
 **public static void** tellClientInfo(**int** ID)  
 {  
 *idMas*.remove(ID);  
 **for** (**int** i = 0; i < *mas*.size(); i++)  
 {  
 **if**(*mas*.get(i).**ID** == ID) *mas*.remove(i);  
 }  
 **for** (**int** i = 0; i < *mas*.size(); ++i)  
 {  
 *mas*.get(i).sendActive(*idMas*);  
 }  
 }  
}

**public class** OneJabber **extends** Thread {  
 **public** DataInputStream **in**;  
 **public** DataOutputStream **out**;  
 **private** Socket **socket**;  
 **public int ID**;  
 Vector<OneJabber> **mas**;  
  
 **public** OneJabber(Socket s, **int** ID, Vector<OneJabber> vector) **throws** IOException {  
 **this**.**ID** = ID;  
 **mas** = vector;  
 **socket** = s;  
 **out** = **new** DataOutputStream(**socket**.getOutputStream());  
 **out**.flush();  
 **out**.writeInt(ID);  
 **out**.flush();  
 **in** = **new** DataInputStream(**socket**.getInputStream());  
 System.***out***.println(**"New connection, ID = "** + ID);  
 start();  
 }  
  
 **public void** sendActive(HashSet<Integer> ss)  
 {  
 **try** {  
 **out**.writeInt(1);  
 **out**.writeInt(ss.size() - 1);  
 Iterator<Integer> it = ss.iterator();  
 **while** (it.hasNext()) {  
 **int** id = it.next();  
 **if** (id != **ID**)  
 **out**.writeInt(id);  
 }  
  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
  
 }  
  
 @Override  
 **public void** run() {  
  
 **while** (**true**) {  
 **try** {  
 **int** command = **in**.readInt();  
 **if**(command == 0)  
 {  
 **socket**.close();  
 **in**.close();  
 **out**.close();  
 Main.*tellClientInfo*(**ID**);  
 **break**;  
 }  
 **if**(command == 1)  
 {  
 System.***out***.println(**"ЕДИНИЧКА"**);  
 **int** idToSend = **in**.readInt();  
 **int** kolvo = **in**.readInt();  
 System.***out***.println(kolvo);  
 **for**(**int** i = 0; i < **mas**.size(); i++)  
 {  
 **if**(**mas**.get(i).**ID** == idToSend)  
 {  
 OneJabber stre = **mas**.get(i);  
 **try** {  
 stre.**out**.writeInt(2);  
 stre.**out**.writeInt(kolvo);  
 **for** (**int** j = 0; j < kolvo; ++j)  
 {  
 **int** type = **in**.readInt();  
 stre.**out**.writeInt(type);  
 stre.**out**.writeDouble(**in**.readDouble());  
 stre.**out**.writeDouble(**in**.readDouble());  
 stre.**out**.writeInt(**in**.readInt());  
 stre.**out**.writeDouble(**in**.readDouble());  
 stre.**out**.writeDouble(**in**.readDouble());  
 **if** (type == 1)  
 {  
 stre.**out**.writeDouble(**in**.readDouble());  
 stre.**out**.writeDouble(**in**.readDouble());  
 }  
  
 }  
  
 }  
 **catch** (Exception e) {  
  
 }  
  
  
 }  
 }  
  
 }  
  
  
 } **catch** (Exception ex) {  
 }  
 }  
 }

**Вывод:**

В ходе выполнения лабораторной работы мы изучили такие понятия, как сокет, клиент, сервер, реализовали TCP соединение между клиентами через сервер.