package com.company;  
  
import java.util.Scanner;  
  
public class Main {  
  
 public static void main(String[] args) {  
 Scanner scanner=new Scanner(System.*in*);  
 System.*out*.println("Выберите режим работы программы (S - сервер, С - клиент):");  
 String mode =scanner.nextLine();*// Или короче String mode=new Scanner(System.in).nextLine();* switch (mode){  
 case "S":new ChatServer();break;*// Если пользователь выбрал S, то создаем экземпляр класса чат сервер.* case "C":new ChatClient();break;  
 } }}

package com.company;  
  
import java.io.\*;  
import java.net.Socket;  
  
public class Connection {  
 private String login;  
 private final Thread thread;  
 private final Socket socket;  
 private final BufferedReader in;  
 private final BufferedWriter out;  
 private ConnectionListener connectionListener;  
  
 public Connection(Socket socket,ConnectionListener connectionListener) throws IOException {  
 this.socket=socket;  
 this.connectionListener=connectionListener;  
 *//Связываем входной поток с данными, входящими по сети* in=new BufferedReader(new InputStreamReader(socket.getInputStream()));*// Связываем входной поток сокета со входным потоком сети* out=new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));*// Связываем входной поток сокета со входным потоком сети* thread=new Thread(new Runnable() {*// Создаем новый поток одним из 2-х способов. В данном случае путем реализации интерфейса Runnable.* @Override  
 public void run() {  
 try { login=in.readLine();*// Обращаемся к входному потоку, используем метод readLine, чтобы считать введенные данные пользователем.* connectionListener.onConnect(Connection.this);*// Если подключение сорвалость, то надо подстраховаться и передать ссылку на текущее подключение  
 // Организуем цикл, чтобы можно было отправлять данные сколько угодно, пока не прервется связь* while (!thread.isInterrupted()){*// Проверяем прерван поток или нет. Т.е. пока поток не прерван.* connectionListener.onRecieveString(Connection.this,in.readLine());}*// Получаем сообщения* } catch (IOException e) {  
 System.*out*.println("Ошибка сетевого подключения!");e.printStackTrace();  
 }finally { connectionListener.onDisconnect(Connection.this);*// Заканчиваем работу потока, чтобы освободить ресурсы . Мы вызываем текущий класс Connection и у него используем поле this.* } } } );  
 thread.start(); }

public void sendMessage(String message){  
 try {  
 out.write(message+"\n");*// Обращаемся к выходному потоку, вызываем у него метод write* out.flush();*// Позволяет очистить поток от остатков данных* } catch (IOException e) {  
 e.printStackTrace();  
 disconnect(); *// Отключаемся.* } }  
  
 public void disconnect(){  
 thread.interrupt();*//прерываем поток* try { socket.close();*// Отключаемся от сервера. Освобождаем сокет,закрывая его.* } catch (IOException e) { e.printStackTrace(); } }  
  
 public String getLogin() {return login; }}

package com.company;  
  
import java.io.IOException;  
import java.net.ServerSocket;  
import java.util.ArrayList;  
  
public class ChatServer implements ConnectionListener{  
  
 private ArrayList<Connection> connections;*// Объявляем список, в к-ый будем добавлять или удалять пользователей.* ServerSocket serverSocket;  
  
 public ChatServer() {  
  
 connections=new ArrayList<>();  
  
 try { serverSocket=new ServerSocket(Constans.*PORT*);}*// Создаем серверный сокет.* catch (IOException e) {  
 System.*out*.println("Ошибка запуска сервера!");  
 return;  
 *//e.printStackTrace();* }  
 System.*out*.println("Сервис успешно запущен!");  
  
 while (true){  
 try {  
 new Connection(serverSocket.accept(),this);*//Создаем подключение, говорим, что все события, описанные в connectionListener, будет выполнять текущий экземпляр класса ChatServer(serverSocket)  
 // Метод accept выполняет двойную функцию. 1-я это принимает входящее подключение, 2-я это возвращает сокет(т.е. берется ip адрес компа, на к-ом сервер запущен, добавляется порт и получается сокет, к-й возвращается по запросу для текущего подключения.* } catch (IOException e) {  
 System.*out*.println("Ошибка создания подключения!");  
 *// e.printStackTrace();* } } }  
 @Override  
 public synchronized void onConnect(Connection connection) {  
connections.add(connection);*// Добавляем тек. пользователя в список подключений.*String message="Подключение пользователя "+connection.getLogin();*//Выводим на экран логин тек. подключения* System.*out*.println(message);  
 sendMessage(connection,message);*// Посылаем всем пользователям оповещение, что подключился новый поль-ль.* }  
 @Override  
 public synchronized void onDisconnect(Connection connection) {  
 connections.remove(connection);*// Удаляем подключение из списка.* String message ="Отключение пользователя"+connection.getLogin();  
 System.*out*.println(message);  
 sendMessage(connection,message);*// Отправляем смс всем пользователям* }  
 @Override*// Чтобы оповестить всех пользователей о вновь подключившимся* public synchronized void onRecieveString(Connection connection, String message) {  
 sendMessage(connection, connection.getLogin()+" : "+message); }  
  
 private synchronized void sendMessage(Connection connection,String message) {  
 for (Connection item:connections){  
 if (!item.equals(connection))*// Если текущий пользователь не является отправителем смс* item.sendMessage(message);*//всем подключенным пересылаем смс , к-ое пришло в качестве параметра* } }}

package com.company;  
  
import java.io.BufferedReader;  
import java.io.IOException;  
import java.io.InputStreamReader;  
import java.io.PrintWriter;  
import java.net.Socket;  
import java.util.Scanner;  
  
public class ChatClient {  
 private BufferedReader in;  
 private PrintWriter out;  
 private Socket socket;  
 public ChatClient(){  
 Scanner scanner=new Scanner(System.*in*);*// Мы могли бы запросить у пользователя IP адрес сервера и порт, к-му хотим подключиться, считать его с клавиатуры* String ip="127.0.0.1";*// А пока тестируем программу на одном и том же компе, будем использовать наш локалхост.  
 // Это адрес обратной петли, когда клиент и сервер на одном компе, надо протестировать работу, как бы они работали по сети.* try {  
 socket=new Socket(ip,Constans.*PORT*);*// В качестве порта, д.б. одно и тоже число, как для клиента, так и для сервера.* in=new BufferedReader(new InputStreamReader(socket.getInputStream()));*// Создаем входной поток и связываем с входным потоком сокета* out = new PrintWriter(socket.getOutputStream(),true);*// Обращаемся к выходному потоку сокета и обрабатываем с помощью PrintWriter. True означает, что мы подключаем выталкивание данных из буфера автоматически сразу.* System.*out*.println("Введите имя пользователя: ");  
 out.println(scanner.nextLine());*// Выводим в выходной поток нашего клиента, то что ввел пользователь  
 // Объект, к-ый будет перенаправлять смс от сервера в нашу консоль отдельным потоком* Resender resender=new Resender(in);  
 resender.start();  
  
 String message="";  
 while (!message.equals(Constans.*EXIT\_CODE*)){*// Будем пересылать смс, пока пользовать не введет кодовое значение, и тогда выйдем из чата* message=scanner.nextLine();  
 out.println(message);  
 }  
 resender.setStop();*// Как только выходим из чата, пересылку останавливаем.* }  
 catch (IOException e)  
 {  
 *// e.printStackTrace();* System.*out*.println("Ошибка подключения к серверу!");  
 } finally {  
 try {  
 if(in!=null) in.close();  
 if(out!=null )out.close();  
 if (socket!=null) socket.close(); }  
 catch (IOException e){  
 *// e.printStackTrace();* System.*out*.println("Ошибка при очистке потоков!");  
 } }  
 } }

package com.company;  
  
import java.io.BufferedReader;  
import java.io.IOException;  
  
import static java.lang.System.*out*;  
*// Создаем отдельный поток для того, ч. передавать смс, пока другой поток прослушивает подключения.*public class Resender extends Thread{  
 private BufferedReader bufferedReader;  
  
 private boolean stoped;*// Нужна логическая переменная, к-ая будет показывать работает ли сейчас тек. поток.  
 //Получается, ч. остановить пересылку смс, мы д. значение этой переменной изменить.* public Resender (BufferedReader bufferedReader){  
 this.bufferedReader=bufferedReader;  
}  
 public void setStop(){  
 stoped=true;}  
 @Override  
 public void run() {*// Поскольку наследуемся от класса thread, мы д. переопределить метод run. Этот один из 2х способов создания потока. Другой через runable.* String message = "";  
 while (!stoped) {*// Будем отправлять смс пока это значение ложь.* try {  
 message=bufferedReader.readLine();  
 *out*.println(message);  
 *//message = bufferedReader.readLine();// Считываем значения из входного потока.* } catch (IOException e) {  
 System.*out*.println("Произошло отключение от сервера!");  
 e.printStackTrace();  
 } } }}

package com.company;  
  
public class Constans {  
 public static int *PORT*=2302;  
 public static final String *EXIT\_CODE*="/exit";  
}

package com.company;  
  
public interface ConnectionListener {  
 void onConnect(Connection connection);*// После установления подключения возникнет событие* void onDisconnect(Connection connection);*// Событие, когда мы отключаемся от сервера* void onRecieveString(Connection connection,String message);*// Метод отправки сообщения*}