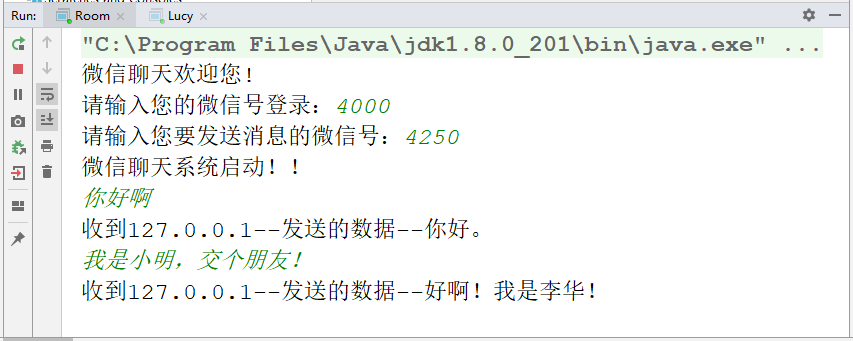
【案例9-2】 模拟微信聊天

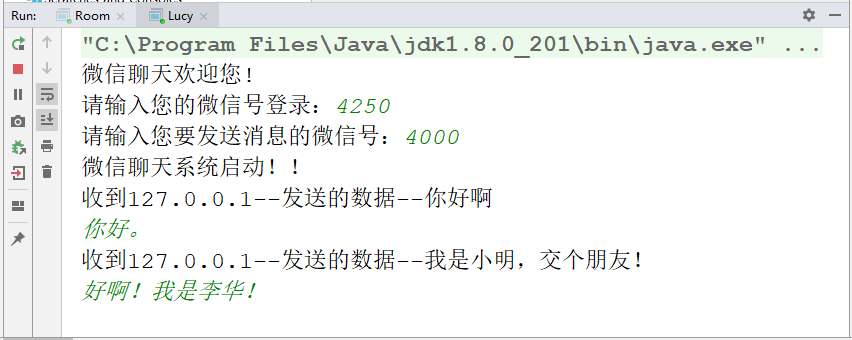
**【案例介绍】**

**1.案例描述**

在如今，微信聊天已经人们生活中必不可少的重要组成部分，人们的交流很多都是通过微信来进行的。本案例要求：将多线程与UDP通信相关知识结合，模拟实现微信聊天小程序。通过监听指定的端口号、目标IP地址和目标端口号，实现消息的发送和接收功能，并显示聊天的内容。

**2.运行结果**





运行结果

**【案例目标】**

* 学会分析“模拟微信聊天”任务的实现思路。
* 根据思路独立完成“模拟微信聊天”任务的源代码编写、编译及运行。
* 掌握网络通信中UDP协议的编程原理。
* 掌握UDP网络通信DatagramPacket和DatagramSocket的使用。

**【案例分析】**

（1）第一要知道用什么技术实现，通过上述任务描述可知此任务是使用多线程与UDP通信相关知识实现的。要实现图中的聊天窗口界面。首先需要定义一个实现微信聊天功能的类，类中需要定义访问微信聊天的输出语句，从而获取输入的发送端端口号、接收端端口号以及实现发送和接收功能的方法。

（2）实现发送数据的功能。该功能通过一个实现了Runnable接口的类实现，类中需要定义获取发送数据的端口号，并在实现run()的方法中，编写发送数据的方法。

（3）实现接收数据的功能。该功能通过一个实现了Runnable接口的类实现，类中需要定义获取接收数据的端口号，并在实现run()的方法中，编写显示接收到的数据的方法。

（4）创建完所有的类与方法后，运行两次程序，同时开启两个窗口来实现聊天功能。

**【案例实现】**

（1）创建微信聊天程序，其代码具体如下所示。

Room.java

import java.util.Scanner;

public class Room {

public static void main(String[] args) {

System.out.println("微信聊天欢迎您!");

Scanner sc = new Scanner(System.in);

System.out.print("请输入您的微信号登录：");

int sendPort = sc.nextInt();

System.out.print("请输入您要发送消息的微信号：");

int receivePort = sc.nextInt();

System.out.println("微信聊天系统启动！！");

//发送操作

new Thread(new SendTask(sendPort), "发送端任务").start();

//接收操作

new Thread(new ReceiveTask(receivePort), "接收端任务").start();

}

}

上述代码中，第12行代码用多线程实现发送端口号以及实现发送端功能的方法。第14行代码用多线程实现接收端口号以及实现接收端功能的方法。

（2）创建发送数据的任务类，其代码如下所示。

SendTask.java

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;

import java.util.Scanner;

public class SendTask implements Runnable {

private int sendPort; // 发数据的端口号

// 构造方法

public SendTask(int sendPort) {

this.sendPort = sendPort;

}

@Override

public void run() {

try {

// 1. 创建DatagramSocket对象

DatagramSocket ds = new DatagramSocket();

// 2.输入要发送的数据

Scanner sc = new Scanner(System.in);

while (true) {

String data = sc.nextLine();// 获取键盘输入的数据

// 3.封装数据到 DatagramPacket对象中

byte[] buf = data.getBytes();

DatagramPacket dp = new DatagramPacket(buf, buf.length,

InetAddress.getByName("127.0.0.255"),sendPort);

// 4.发送数据

ds.send(dp);

}

} catch (Exception e) {

e.printStackTrace();

}

}

}

上述代码中，第6~10行代码声明了一个名称为sendPort的变量表示发送数据的端口号，并通过该类的构造方法获取所输入的端口号。第12~29行代码，实现run()方法，在run()方法中，首先创建了DatagramSocket对象，然后通过Scanner对象和循环方法获取键盘输入的数据，并将获取的数据封装到了DatagramPacket对象中，最后通过DatagramSocket对象的send()方法发送数据。在循环方法中，由于发送数据时可能多次连续发送，所以将循环条件设置为true，表示不断循环获取和发送数据。在创建DatagramPacket对象时，为了让当前局域网的所有人都接收到消息，所以将IP地址设置为127.0.0.255。

（3）创建接收数据的任务类，其代码如下所示。

import java.net.DatagramPacket;

import java.net.DatagramSocket;

public class ReceiveTask implements Runnable{

private int receivePort;// 接收数据的端口号

public ReceiveTask(int receivePort) {

this.receivePort = receivePort;

}

@Override

public void run() {

try {

// 1.DatagramSocket对象

DatagramSocket ds = new DatagramSocket(receivePort);

// 2.创建DatagramPacket对象

byte[] buf = new byte[1024];

DatagramPacket dp = new DatagramPacket(buf, buf.length);

// 3.接收数据

while (true) {

ds.receive(dp);

// 4.显示接收到的数据

String str = new String(dp.getData(), 0,

dp.getLength());

System.out.println("收到" +

dp.getAddress().getHostAddress()

+ "--发送的数据--" + str);

}

} catch (Exception e) {

e.printStackTrace();

}

}

}

上述代码中，第4行代码声明了一个名称为receivePort的变量来表示接收数据的端口号，第5~7行代码定义一个该类的构造方法，并通过该类的构造方法获取所输入的端口号，第9~28行代码在run()方法中，首先创建了DatagramSocket对象，创建该对象时需要传入接收数据的端口号，然后创建用于接收数据的DatagramPacket对象，接下来通过循环的方法来接收数据，最后通过输出语句来显示接收到的数据。