实验7 多线程与网络编程

1 实验目的

熟悉Java中的多线程与网络编程技术。

2 实验环境

开发环境：JDK 8.0（或更高版本，高版本要下载独立JavaFX）
开发工具：Eclipse
设计工具：StarUML（或PlantUML）

2 基础知识

2.1 多线程

线程实现包含两种方式，一种是定义继承于Thread类的自定义类，并重写run()函数；另一种是直接创建一个Thread对象，并将实现Runnable接口的任务对象传递给Thread对象。具体实现如下：

```
public class Test {
	public static void main(String[] args) {
		Thread t1 = new MyTask();
		t1.start();
		Thread t2 = new Thread(() -> {
			System.out.println("任务2");
		});
		t2.start();
	}
}
class MyTask extends Thread{
	@Override
	public void run() {
		System.out.println("任务1");
	}
}
```

2.2 网络编程

套接字是对网络中不同主机上的应用进程之间进行双向通信的端点的抽象。JDK中的Socket类表示客户端，ServerSocket表示服务端。Socket类通过connect()函数连接服务端，ServerSocket类通过accept()监听并获取连接到本服务端程序的套接字。通过Socket类获取InputStream和OutputStream对象，实现从套接字获取数据或向套接字发送数据。

4 实验内容

4.1 简单的聊天工具

需求描述

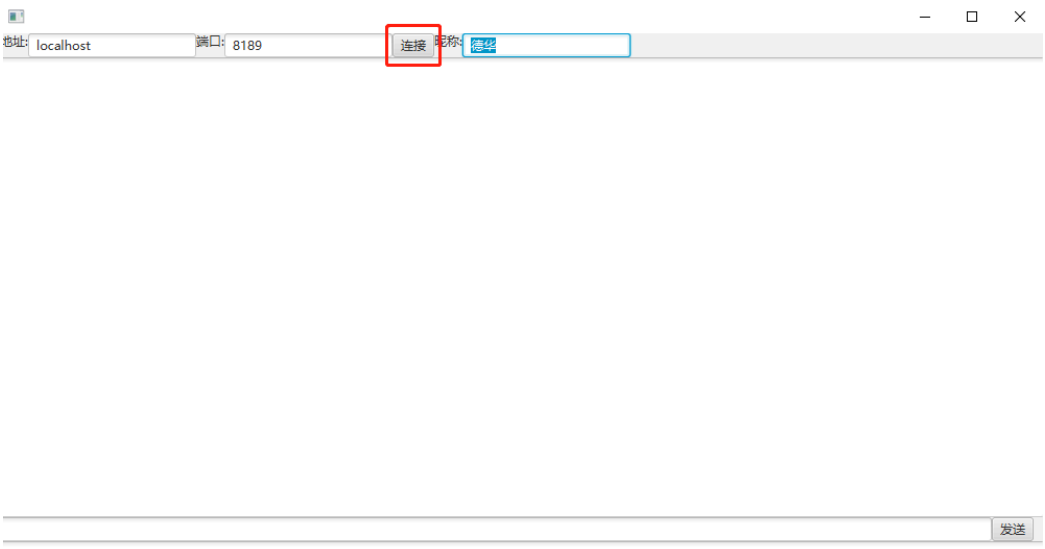
现有一个简单的聊天工具程序代码，请按步骤调试并运行，分析程序的实现原理。

实验步骤

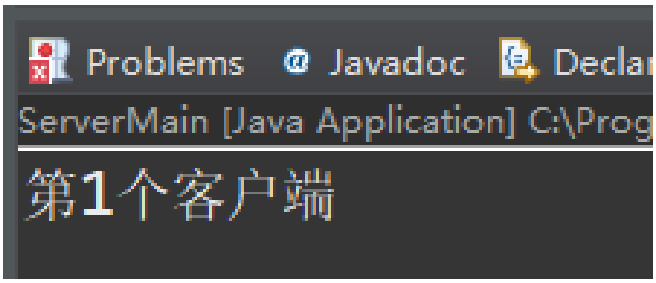
1、在ServerMain源代码中设置服务器端口，默认为8189

```
public class ServerMain{
        public static void main(String[] args) {
                try (ChatServer server = new ChatServer(8189)) {
                        server.start();
                }
                catch (IOException e) {
                        e.printStackTrace();
                }
        }
}
```

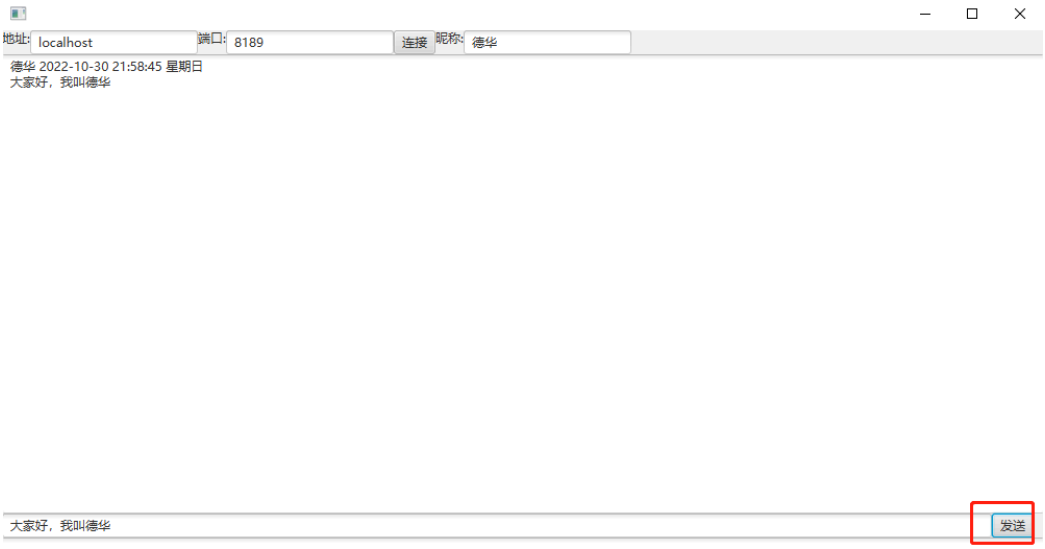
2、运行服务器主程序（ServerMain）
3、本地第一个客户端连接服务器
（1）在当前计算机运行1个客户端程序（ClientMain），设置地址为localhost和端口8189，点击连接



（2）观察服务器有客户端连接的提示

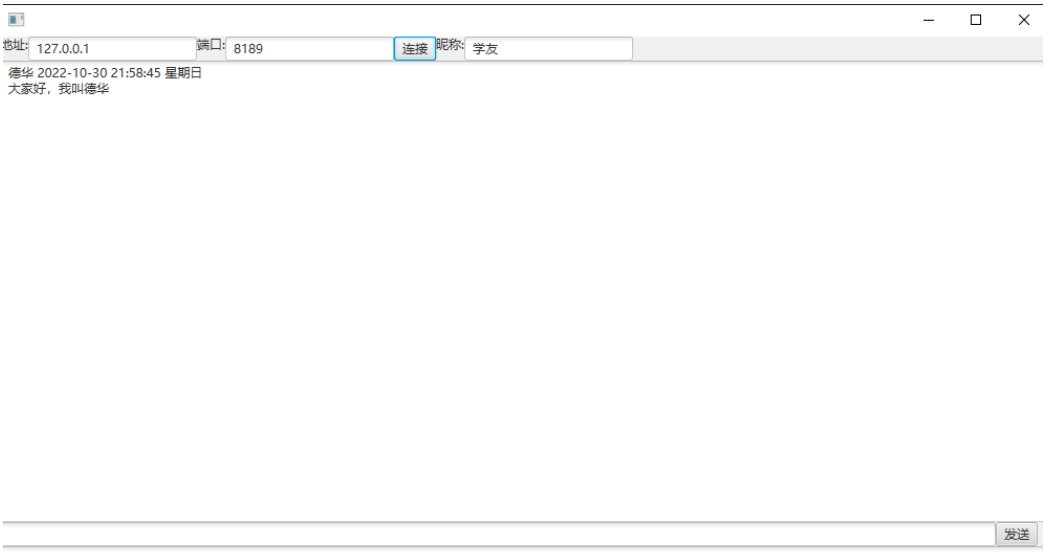


（3）在当前聊天室发送消息



4、本地第二个客户端链接服务器

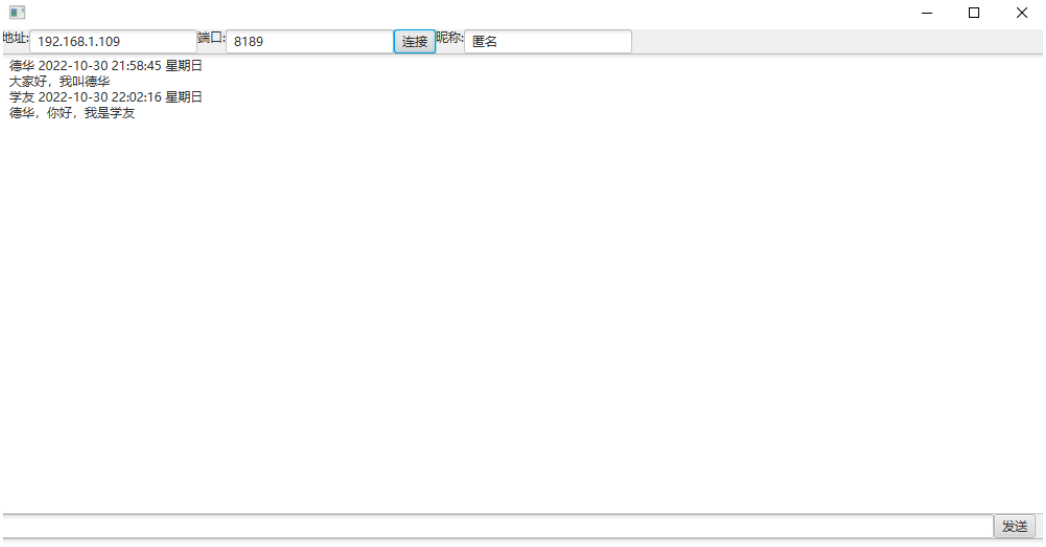
（1）在当前计算机运行另一个客户端程序（ClientMain），设置地址为127.0.0.1和端口8189，点击连接。连接到服务器后会显示当前其他用户的聊天记录。



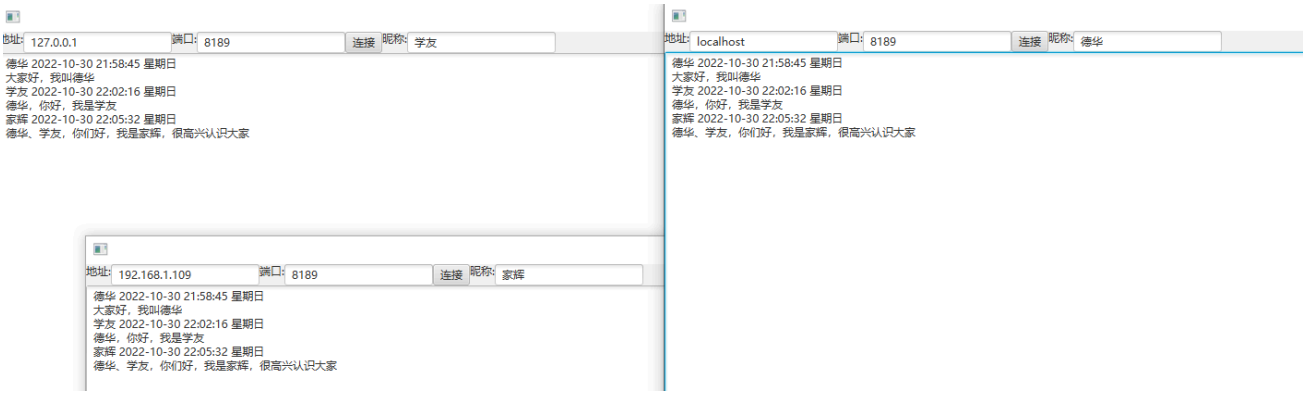
（2）在当前聊天室发送消息，观察到两个客户端的聊天记录刷新。

5、局域网第三个客户端链接服务器

（1）在当前局域网的另一台机器运行客户端程序（ClientMain），设置服务器地址和端口8189，点击连接。连接到服务器后会显示当前其他用户的聊天记录。



（2）在当前聊天室发送消息，观察到三个客户端的聊天记录刷新。



6、完成上述步骤后，尝试阅读代码分析聊天程序的实现原理。

思考问题：

1、套接字（Socket）是什么？
2、服务器程序是如何通过多线程支持与多个客户端通信的？
3、如果需要将聊天记录持久化，是否需要数据库？如何实现？
4、客户端发送一个get命令有什么作用？

5 实验要求

5.1 实验评价

1、完善实验程序，调试、运行并提交指导老师检查（100%）