作业一

基于模块三的作业四展开，ReadList类和WriteList类参考了网上的做法，并对其做了优化。

1. 创建两个异常类AgeException和IdException，这两个类分别继承Exception类。
    - 生成两种构造方法：有参构造和无参构造
    - 生成序列号
2. 在Student类的`setId`和`setAge`两个方法中调用这两个异常

```
public void setId(int id) throws IdException {
  if(id <= 0){
    throw new IdException("ID小于等于0! ");
  }else{
    this.id = id;
  }
}
```

```
public void setId(int id) throws IdException {
  if(age <= 0){
    throw new AgeException("年龄小于等于0! ");
  }else{
    this.age = age;
  }
}
```

3. 创建ReadList类，其中有一个静态方法用于读取文件的内容

```
public static List<Student> readList(String path) {
  ObjectInputStream in = null;
  List<Student> listStudents =null;
  ...
}
```

    - 使用对象输入流，把读取到的数据存到集合中

```
try{
   in = new ObjectInputStream(new FileInputStream(path));
   Object obj = in.readObject();
   listStudents = (List<Student>)obj;
}catch(){
  ...
}finally{
  ...
}
```

- 最后返回取到的数据

```
return listStudents;
```

4. 创建WriteList类，其中有一个静态方法用于把内容写到文件中

```
public static void writeList(List<Student> listStudents, String path) {
  ObjectOutputStream out = null;
  ...
}
```

- 使用对象输出流，把数据写到文件当中

```
try{
  out = new ObjectOutputStream(new FileOutputStream(path));
  out.writeObject(listStudents);
}catch(){
  ...
}finally{
  ...
}
```

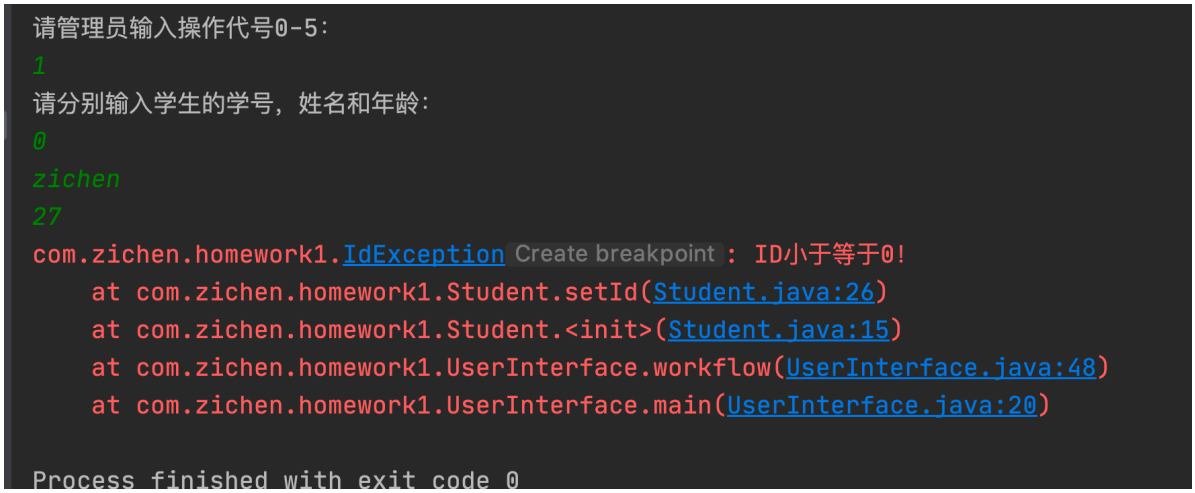
5. 对UserInterface类进行修改
   - 对main进行修改

```
List<Student> studentList = null;
String path = "./src/com/zichen/homework1/studentList.txt";
File txt = new File(path);
if (txt.exists()) {
  studentList = ReadList.readList(path);
}else{
  studentList = new LinkedList<>();
}
try{
  workflow(studentList, path);
}catch(){
 ...
}
```

- 对case0进行修改，添加

```
WriteList.writeList(ms.returnStudentList(), path);
```

6. 运行结果截图
   - Id异常验证



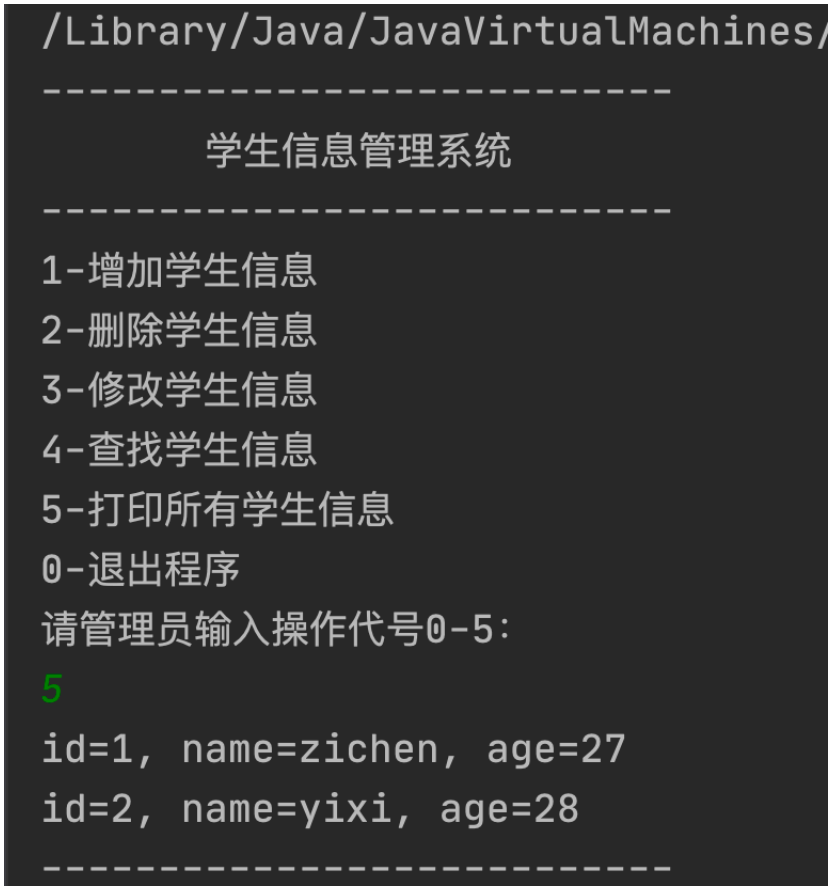
   - 年龄异常验证

```
请管理员输入操作代号0-5:
1
请分别输入学生的学号，姓名和年龄:
1
zichen
0
com.zichen.homework1.AgeException Create breakpoint : 年龄小于等于0!
    at com.zichen.homework1.Student.setAge(Student.java:46)
    at com.zichen.homework1.Student.<init>(Student.java:17)
    at com.zichen.homework1.UserInterface.workflow(UserInterface.java:48)
    at com.zichen.homework1.UserInterface.main(UserInterface.java:20)

Process finished with exit code 0
```

- 文件读取验证

```
----------------------------
1-增加学生信息
2-删除学生信息
3-修改学生信息
4-查找学生信息
5-打印所有学生信息
0-退出程序
请管理员输入操作代号0-5:
0
程序已退出!
----------------------------
```



作业二

利用递归算法从里到外删除文件

1. 创建一个方法用于创建目录

```
public static void createFolder(String path) throws IOException {
  File file = new File(path);
  if(!file.exists()){
    file.mkdirs();
    System.out.println("目录创建成功! ");
  }else{
    System.out.println("目录已存在! ");
  }
}
```

2. 创建一个方法用于创建文件

```
public static void createFile(String path) throws IOException {
  File file = new File(path);
  if(!file.exists()){
    file.createNewFile();
    System.out.println("文件创建成功！");
  }else{
    System.out.println("文件已存在！");
  }
}
```

3. 创建一个方法用于删除所有文件，注意最后要删除空目录。

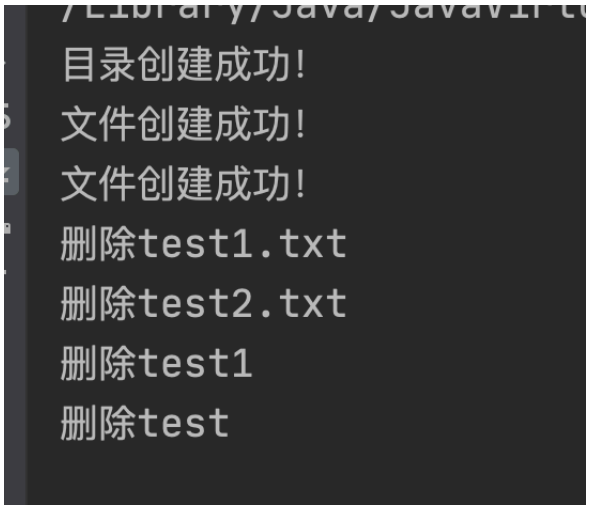
```
public static void deleteAllFiles(File rootFile){
  if(!rootFile.exists()){
    return;
  }
  File[] files = rootFile.listFiles();
  for(File file : files){
    if(file.isFile()){
      System.out.println("删除"+file.getName());
      file.delete();
    }else if(file.isDirectory()){
      deleteAllFiles(file);
    }
  }
  System.out.println("删除"+rootFile.getName());
  rootFile.delete();
}
```

4. main方法

```
public static void main(String[] args) {
  String rootPath = "./src/com/zichen/homework2/test";
  String folderPath = "./src/com/zichen/homework2/test/test1";
  String filePath1 = "./src/com/zichen/homework2/test/test1/test2.txt";
  String filePath2 = "./src/com/zichen/homework2/test/test1.txt";
  try {
    createFolder(folderPath);
    createFile(filePath1);
    createFile(filePath2);
   } catch (IOException e) {
    e.printStackTrace();
   }
   deleteAllFiles(new File(rootPath));
```

```
}
```

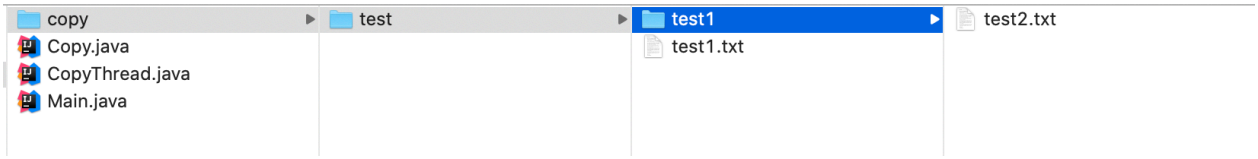
5. 运行结果截图



作业三

将copy目录下的所有内容拷贝到paste目录下

1. 手动创建了copy目录



2. 创建Copy类，里面包含两个方法，一个是复制文件，一个是复制目录
   - 创建一个copyFile方法用于复制文件，此方法与老师上课讲的方法一致，除了参数部分

```
public void copyFile(String oldPath, String newPath) {
  ...
}
```

   - 创建一个copyFolder方法用于复制目录

```
public synchronized void copyFolder(String oldPath, String newPath){

}
```

     - 首先根据路径创建File类的对象

```
File oldFolder = new File(oldPath);
File newFolder = new File(newPath);
```

- 对文件目录进行判断

```
if(!oldFolder.exists()){
  System.out.println("原文件夹不存在！");
  return;
}

if(!newFolder.exists()){
  newFolder.mkdirs();
  System.out.println("复制目录"+newFolder.getName());
}
```

- 获取当前目录下的文件信息，如果是文件直接复制，如果是目录，递归调用copyFolder方法。这里需要注意参数的传递。

```
for(File f : files){
  if(f.isFile()){
    copyFile(f.getAbsolutePath(),
newFolder.getAbsolutePath()+"/"+f.getName());
    System.out.println("复制文件"+f.getName());
  }else if(f.isDirectory()){
    copyFolder(f.getAbsolutePath(),
newFolder.getAbsolutePath()+"/"+f.getName());
  }
}
```

3. 创建一个CopyThread类，用于多线程的实现，此类继承Runnable接口，重写run方法

```
public class CopyThread implements Runnable{
  ...
}
```

- 创建构造方法

```
public CopyThread(String oldPath, String newPath){
  this.oldPath = oldPath;
  this.newPath = newPath;
}
```

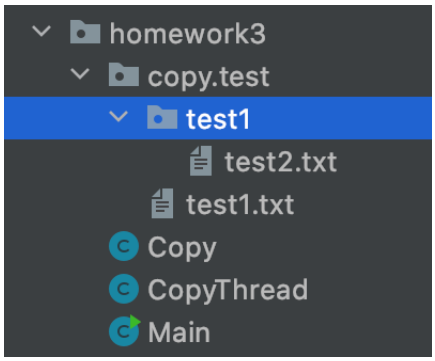
- 重写run方法

```
@Override
public void run() {
  Copy copy = new Copy();
  copy.copyFolder(oldPath, newPath);
}
```

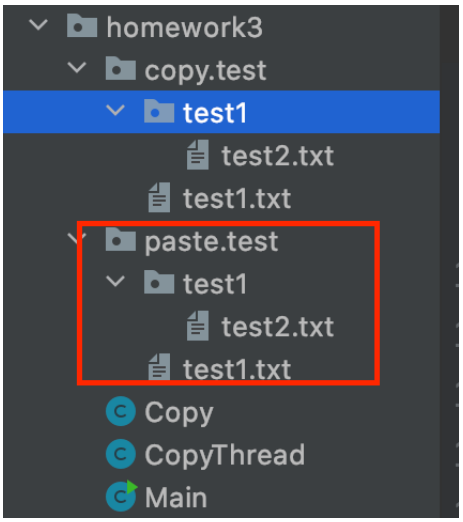
4. 创建Main类，用于测试。

```
public class Main {
  public static void main(String[] args) {
    String oldPath = "./src/com/zichen/homework3/copy";
    String newPath = "./src/com/zichen/homework3/paste";
    ExecutorService executorService = Executors.newFixedThreadPool(10);
    executorService.execute(new CopyThread(oldPath, newPath));
    executorService.shutdown();
  }
}
```

5. 运行结果截图
   - 运行前



   - 运行后



作业四

是4.4.6和4.4.7的结合，按照老师讲解的思路即可

1. 根据题意创建User类
2. 根据题意创建UserMessage类
3. 创建Server类，并生成main方法

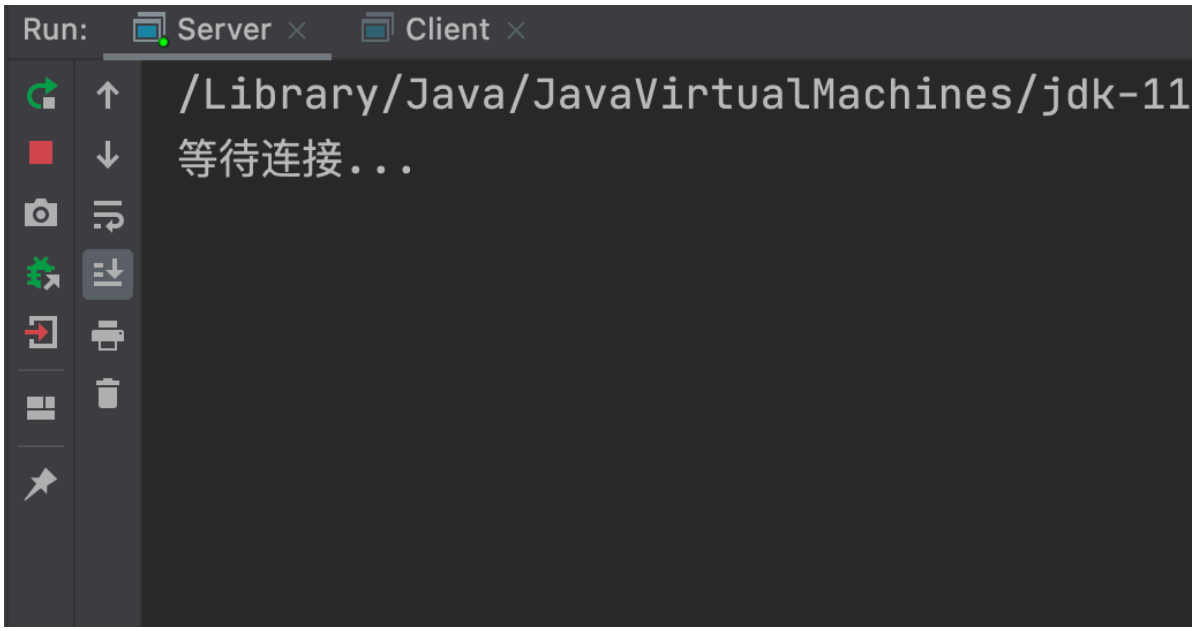
```
try{
  serverSocket = new ServerSocket(8888);
  System.out.println("等待连接...");
  socket = serverSocket.accept();
  objectInputStream = new ObjectInputStream(socket.getInputStream());
  UserMessage obj = (UserMessage)objectInputStream.readObject();
  if(obj.getUser().getUsername().equals("admin")){
    obj.setType("success");
  }else{
```

```
    obj.setType("fail");
  }
  objectOutputStream = new ObjectOutputStream(socket.getOutputStream());
  objectOutputStream.writeObject(obj);
}
```

4. 创建Client类，并生成main方法

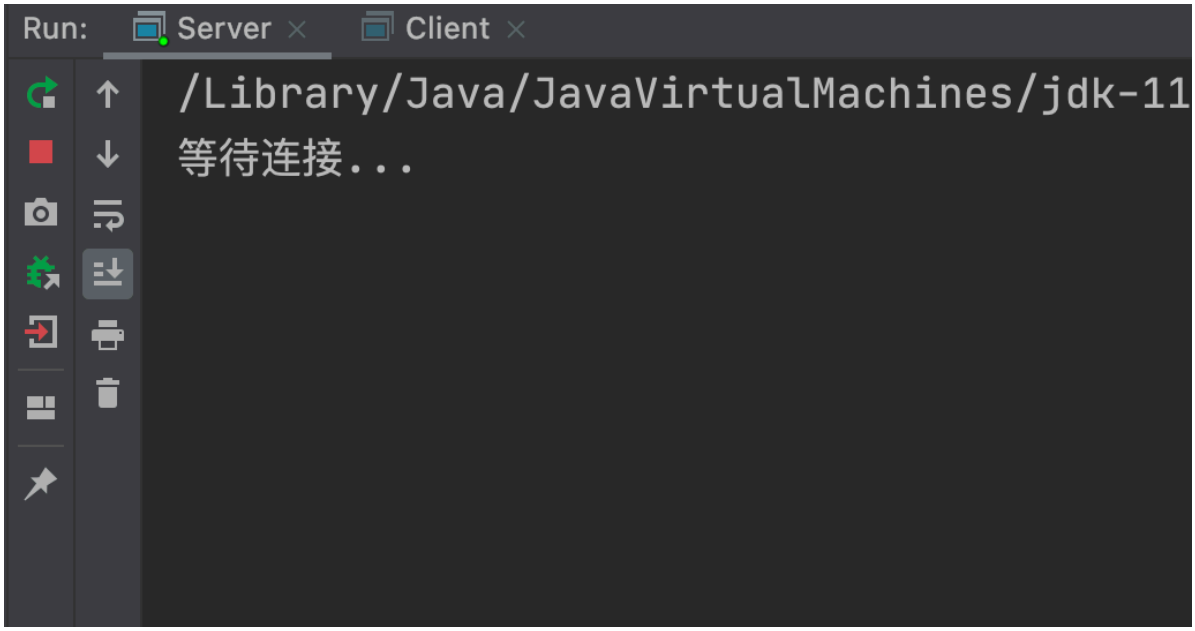
```
try {
  socket = new Socket("127.0.0.1", 8888);
  System.out.println("连接成功! ");
  UserMessage userMessage = new UserMessage("check", new
User("admin","123456"));
  objectOutputStream = new ObjectOutputStream(socket.getOutputStream());
  objectOutputStream.writeObject(userMessage);
  objectInputStream = new ObjectInputStream(socket.getInputStream());
  UserMessage obj = (UserMessage)objectInputStream.readObject();
  if(obj.getType().equals("success")){
    System.out.println("登陆成功! ");
  }else{
    System.out.println("登陆失败! ");
  }
}
```

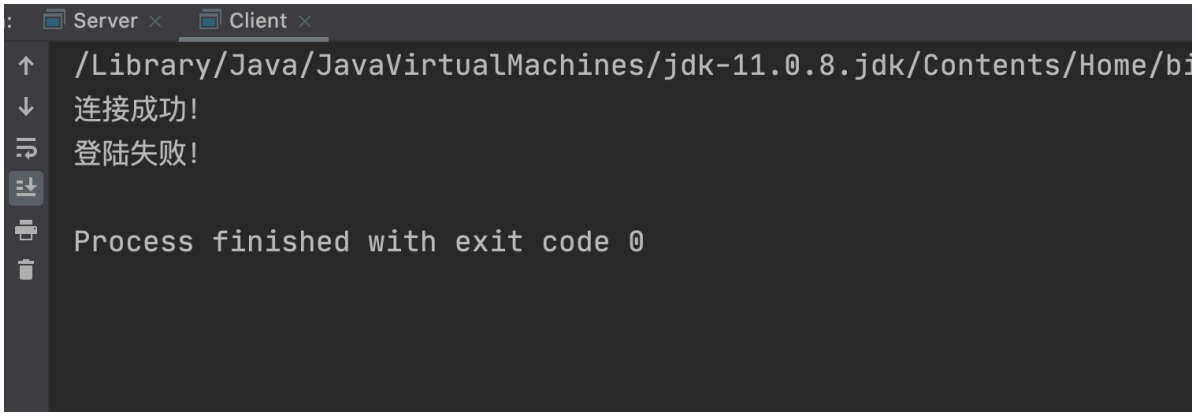
5. 运行结果截图
   - 成功



```
Connected to the target VM, address: '127.0.0.1:58668', transport: 'socket'
连接成功!
登陆成功!
Disconnected from the target VM, address: '127.0.0.1:58668', transport: 'socket'
```

- 失败





作业五

1. 为了模拟多人聊天，创建了两个Client类，一个为Clinet1，一个为Clinet2。这两个类的内部逻辑是一样的。

```
try{
    socket = new Socket("127.0.0.1", 8888);
    sc = new Scanner(System.in);
    printStream = new PrintStream(socket.getOutputStream());
    while(true){
     System.out.println("--------------------");
     System.out.println("输入要发送的信息：");
     String string = sc.next();
```

```
    printStream.println(string);
    System.out.println("发送成功！");
    if(string.equals("退出程序")){
      break;
    }
    bufferedReader = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
    String string2 = bufferedReader.readLine();
    System.out.println();
    System.out.println("服务器回发的字符内容是：" + string2);
   }
}
```

2. 创建Server类，
   - 这里唯一不同的是声明一个list存放client的socket信息。

     ```
     List<Socket> socketList = new ArrayList<>();
     ```

   - 需要把这个list作为参数传到ServerThread的构造方法中，然后启动线程

     ```
     try {
       serverSocket = new ServerSocket(8888);
       while(true){
         System.out.println("--------------------");
         System.out.println("等待连接...");
         socket = serverSocket.accept();
         socketList.add(socket);
         System.out.println("客户端"+socket.getInetAddress()+"连接成功！");
         ServerThread serverThread = new ServerThread(socket, socketList);
         serverThread.start();
       }
     }
     ```

3. 创建ServerThread类，继承了Thread类，并重写了run方法
   - 构造方法

     ```
     public ServerThread(Socket socket, List<Socket> socketList){
       this.socket = socket;
       this.socketList = socketList;
     }
     ```

   - run方法

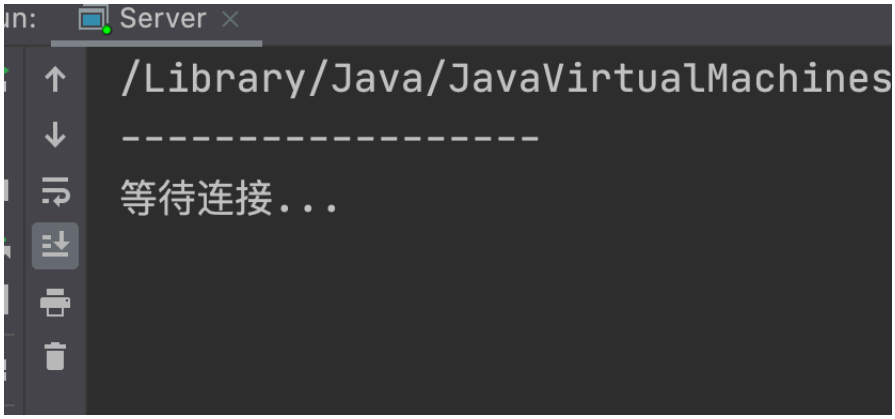
```
@Override
public void run(){
  ...
  try{
    bufferedReader = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
    while(true){
      System.out.println("------------------");
      String string = bufferedReader.readLine();
      System.out.println("客户端"+socket.getInetAddress()+"发来的字符内容是："
+ string);
      if(string.equals("退出程序")){
        System.out.println("客户端"+socket.getInetAddress()+"已下线！");
        break;
      }
      ...
    }
  }catch(){
    ...
  }finally{
    ...
  }

}
```

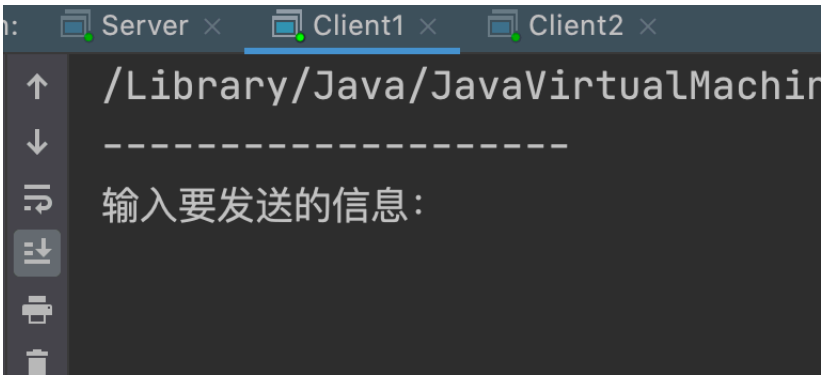
- 在while语句块中，遍历socket

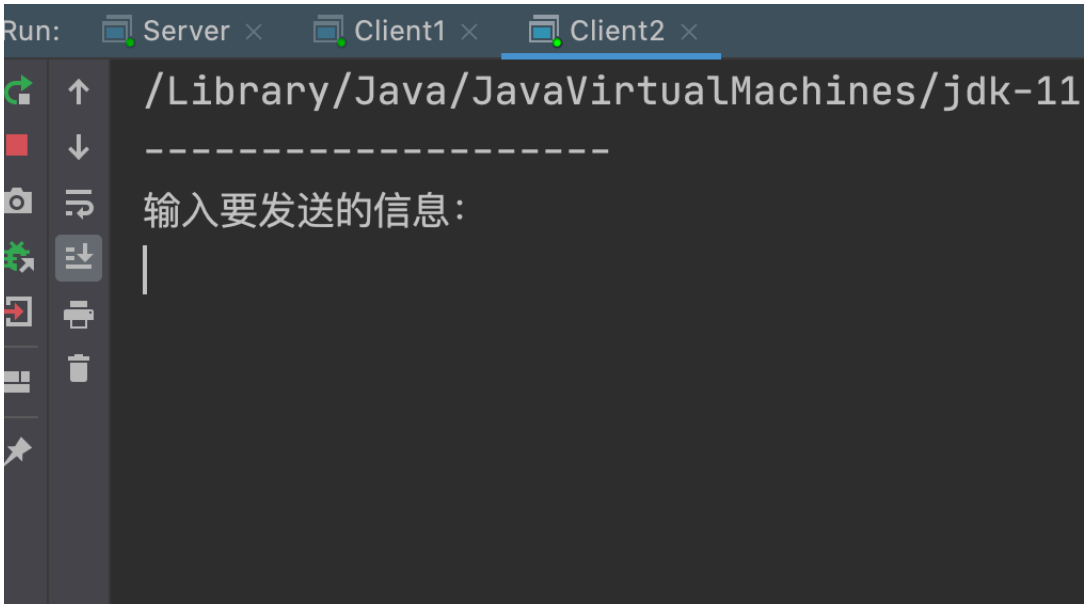
```
while(true){
  ...
  for(socket s : socketList){
    if(!s.equals(socket)){
      printStream = new PrintStream(s.getOutputStream());
      printStream.println(string);
    }
  }
}
```

4. 运行结果截图
   - 开启服务器，等待连接

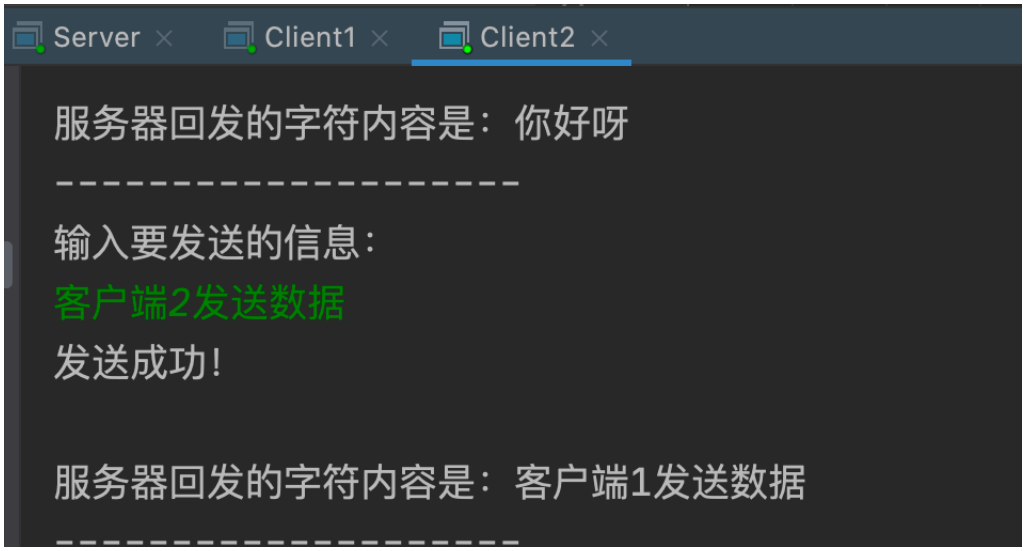


- 开启客户端1和客户端2

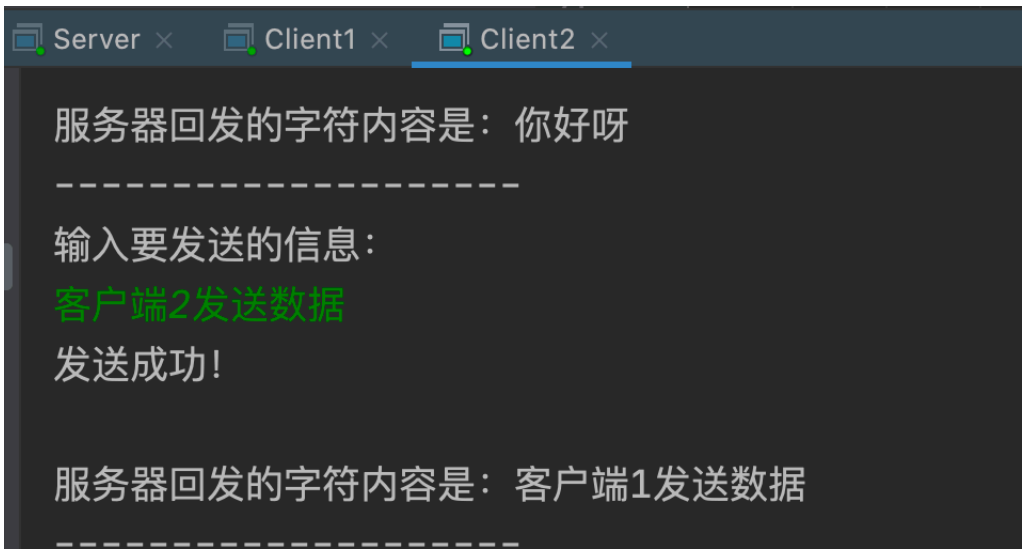




- 客户端1发送消息，客户端2会收到客户端1的消息



- 客户端2发送消息，客户端1会收到客户端2的消息



- 服务器端

注意⚠️：作业五还存在一些问题：

1. 终止客户端后服务器端会有异常
2. 如何让各个线程拿到的最新的list