实验 10：接口程序设计

一、实验目的

1. 掌握接口的基本概念，理解接口与类的区别。
2. 学会定义接口并实现接口内的方法。
3. 理解接口的多态性，能够通过接口引用调用实现类的方法。
4. 掌握客户端与服务端的交互设计，实现基于接口的远程服务调用。
5. 综合运用接口的知识，设计并实现一个较为复杂的医院患者管理系统，包含客户端和服务端。

二、实验学时

2 学时

三、实验类型

验证性

四、实验需求

1、硬件

每人配备计算机 1 台，建议优先使用个人计算机开展实验。

2、软件

IntelliJ IDEA，以及 Java 运行所需要的相关基础环境。

3、网络

本地主机能够访问互联网和实验中心网络。

4、工具

无。

五、实验任务

1. **定义接口**：设计一个 `MedicalService` 接口，包含医疗服务的基本方法。
2. **实现接口**：设计 `Doctor` 类和 `Nurse` 类，分别实现 `MedicalService` 接口。
3. **服务端实现**：实现服务端程序，提供远程医疗服务。
4. **客户端实现**：实现客户端程序，调用服务端的远程服务。
5. **综合任务**：设计一个完整的医院管理系统，支持客户端与服务端的交互。

六、实验内容及步骤

任务 1：定义接口

步骤：

1. 定义一个 `MedicalService` 接口，包含医疗服务的基本方法，如 `diagnose` 和 `treat`。

示例代码：

Java

```java
import java.rmi.Remote;
import java.rmi.RemoteException;

// MedicalService.java
public interface MedicalService extends Remote {
    String diagnose(String patientName) throws RemoteException;
    String treat(String patientName) throws RemoteException;
}

```

任务 2：实现接口

步骤：

1. 设计 `Doctor` 类和 `Nurse` 类，分别实现 `MedicalService` 接口。
2. 在实现类中重写接口方法，提供具体的医疗服务。

示例代码：

Java

```java
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class MedicalServiceImpl extends UnicastRemoteObject implements MedicalService {
    protected MedicalServiceImpl() throws RemoteException {
        super();
    }

    @Override
    public String diagnose(String patientName) {
        return new Doctor("王医生").diagnose(patientName);
    }

    @Override
    public String treat(String patientName) {
        return new Doctor("王医生").treat(patientName);
    }
}
```

任务 3：服务端实现

步骤：

1. 实现服务端程序，提供远程医疗服务。
2. 使用 Java 的 `RMI`（远程方法调用）技术实现远程服务。

示例代码：

Java

```java
// MedicalServiceImpl.java
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class MedicalServiceImpl extends UnicastRemoteObject implements MedicalService {
    protected MedicalServiceImpl() throws RemoteException {
        super();
    }

    @Override
    public String diagnose(String patientName) {
        return new Doctor("王医生").diagnose(patientName);
    }

    @Override
    public String treat(String patientName) {
        return new Doctor("王医生").treat(patientName);
    }
}

// MedicalServer.java
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;

public class MedicalServer {
    public static void main(String[] args) {
        try {
            // 创建远程对象
            MedicalService medicalService = new MedicalServiceImpl();

            // 注册远程对象
            Registry registry = LocateRegistry.createRegistry(1099);
            registry.rebind("MedicalService", medicalService);

            System.out.println("医疗服务端已启动，等待客户端调用...");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

任务 4：客户端实现

步骤：

1. 实现客户端程序，调用服务端的远程服务。
2. 使用 Java 的 `RMI` 技术调用远程方法。

示例代码：

Java

```java
// MedicalClient.java
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;

public class MedicalClient {
    public static void main(String[] args) {
        try {
            // 获取远程对象
            Registry registry = LocateRegistry.getRegistry("localhost", 1099);
            MedicalService medicalService = (MedicalService) registry.lookup("MedicalService");

            // 调用远程方法
            String diagnoseResult = medicalService.diagnose("张三");
            String treatResult = medicalService.treat("张三");

            System.out.println(diagnoseResult);
            System.out.println(treatResult);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

任务 5：综合任务 - 医院管理系统

步骤：

1. 设计一个完整的医院管理系统，支持客户端与服务端的交互。
2. 实现患者挂号、诊断和治疗功能。

示例代码：

Java

```java
// HospitalSystem.java
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.util.Scanner;

public class HospitalSystem {
    public static void main(String[] args) {
        try {
            // 获取远程对象
            Registry registry = LocateRegistry.getRegistry("localhost", 1099);
            MedicalService medicalService = (MedicalService) registry.lookup("MedicalService");

            Scanner scanner = new Scanner(System.in);

            while (true) {
                System.out.println("请选择操作: 1. 诊断 2. 治疗 3. 退出");
                int choice = scanner.nextInt();
                scanner.nextLine(); // 清除缓冲区

                switch (choice) {
                    case 1:
                        System.out.print("请输入患者姓名: ");
                        String patientName = scanner.nextLine();
                        String diagnoseResult = medicalService.diagnose(patientName);
                        System.out.println(diagnoseResult);
                        break;
                    case 2:
                        System.out.print("请输入患者姓名: ");
                        patientName = scanner.nextLine();
                        String treatResult = medicalService.treat(patientName);
                        System.out.println(treatResult);
                        break;
                    case 3:
                        System.out.println("退出系统。");
                        return;
                    default:
                        System.out.println("无效选择，请重新输入。");
                }
```

```
39              }
40          } catch (Exception e) {
41              e.printStackTrace();
42          }
43      }
44  }
```

操作示例：

启动服务端，如图 1、图 2 所示：

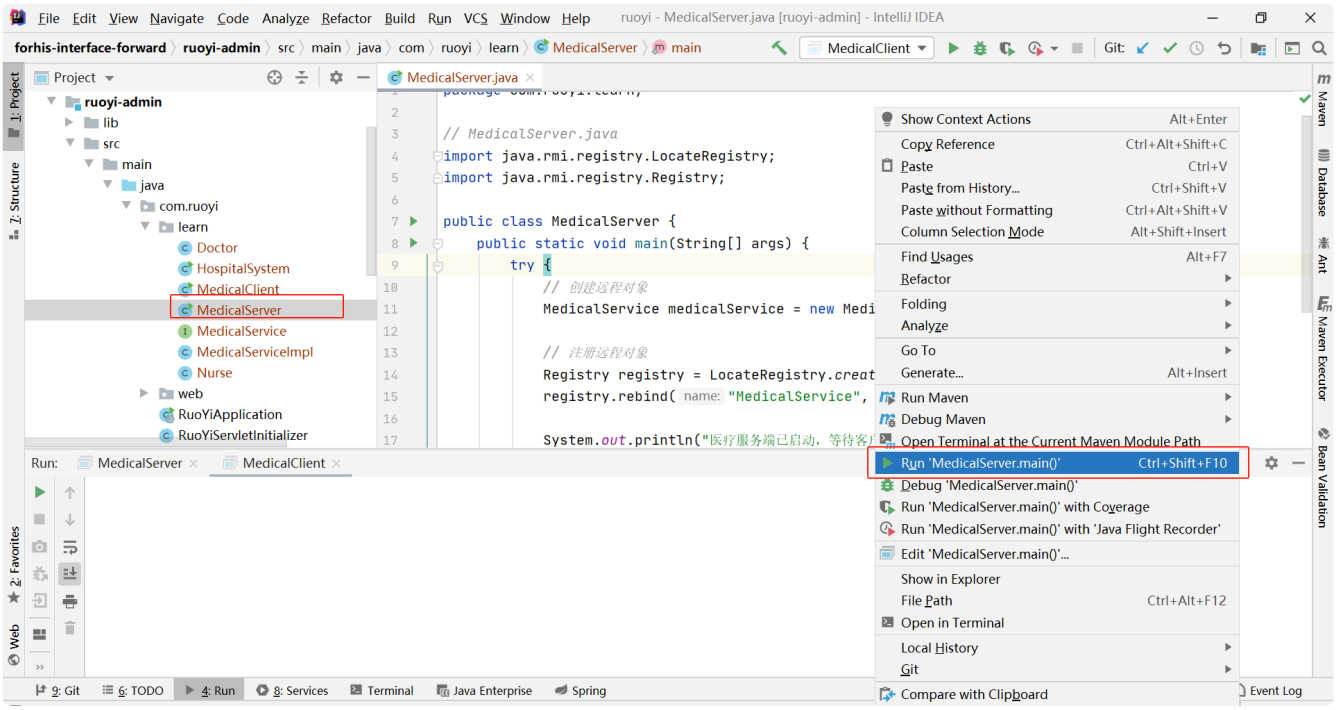

图 1 找到启动按钮

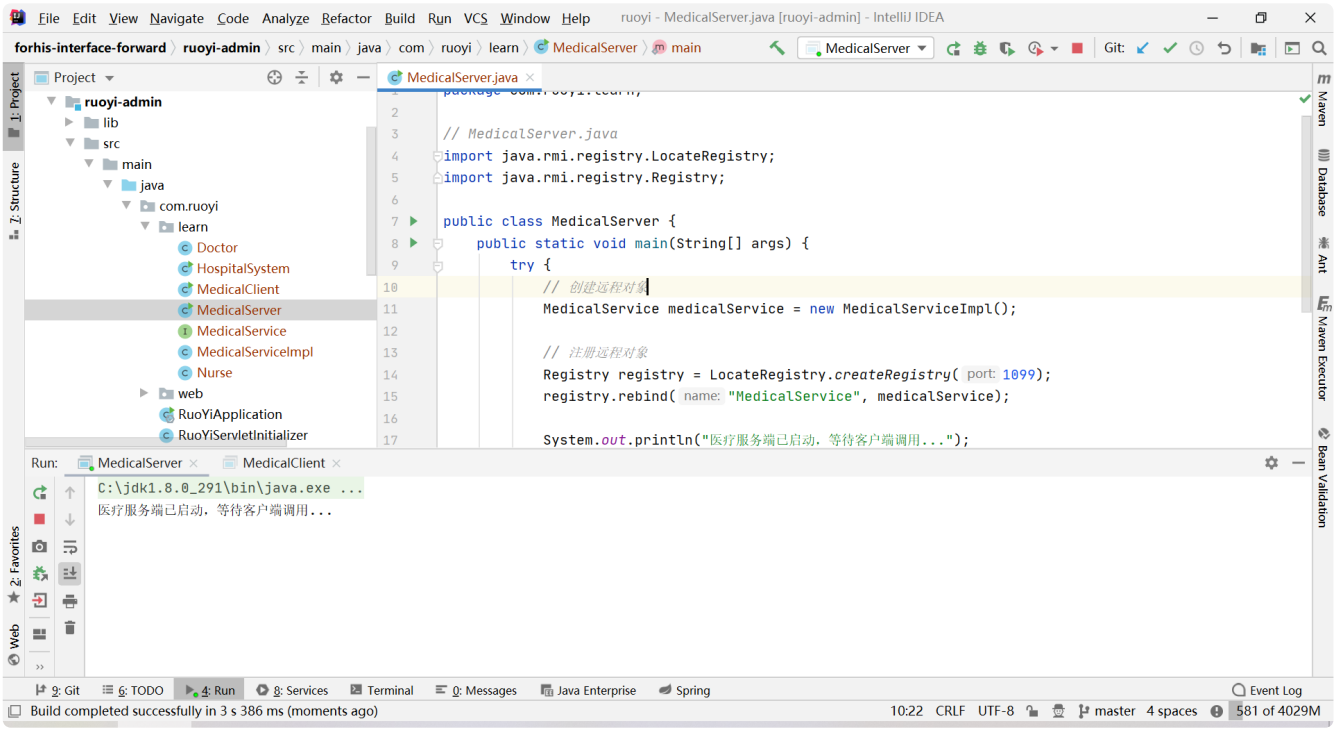

图 2 运行结果

启动客户端，如图 3、图 4 所示：

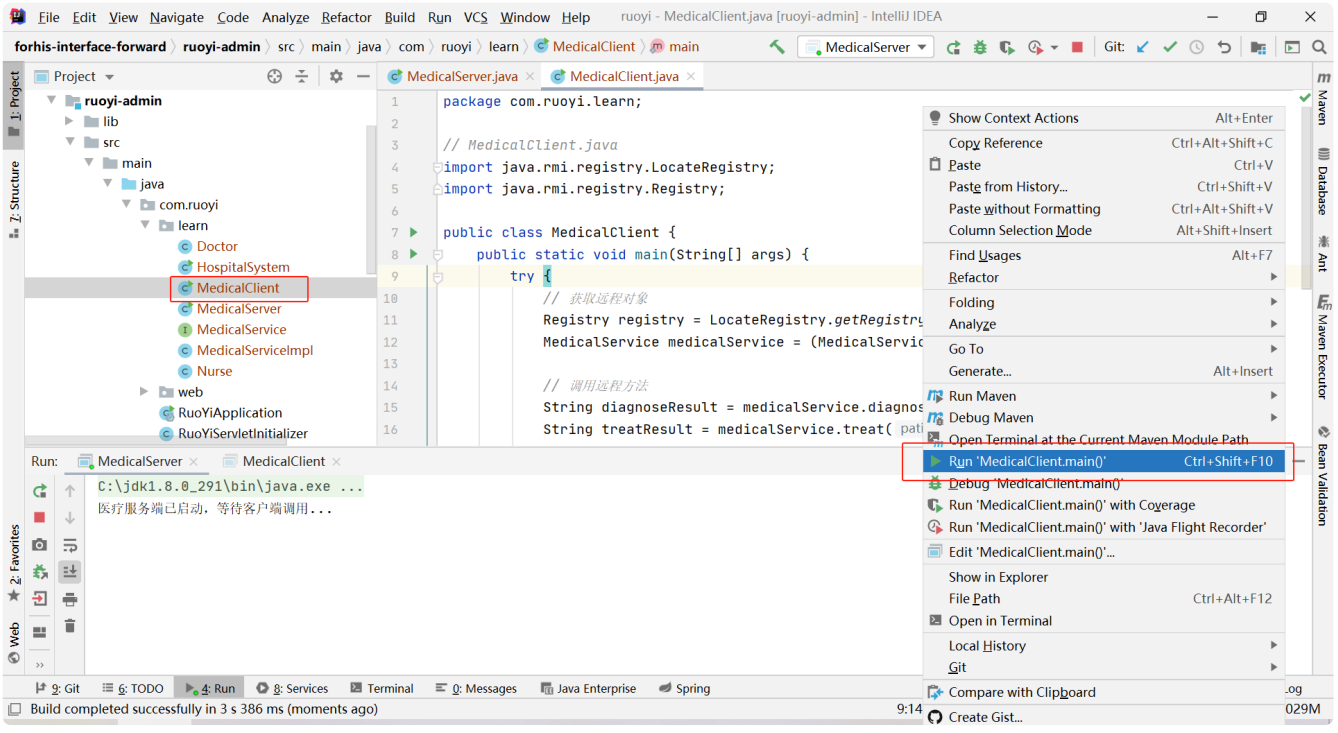

图 3 找到启动按钮

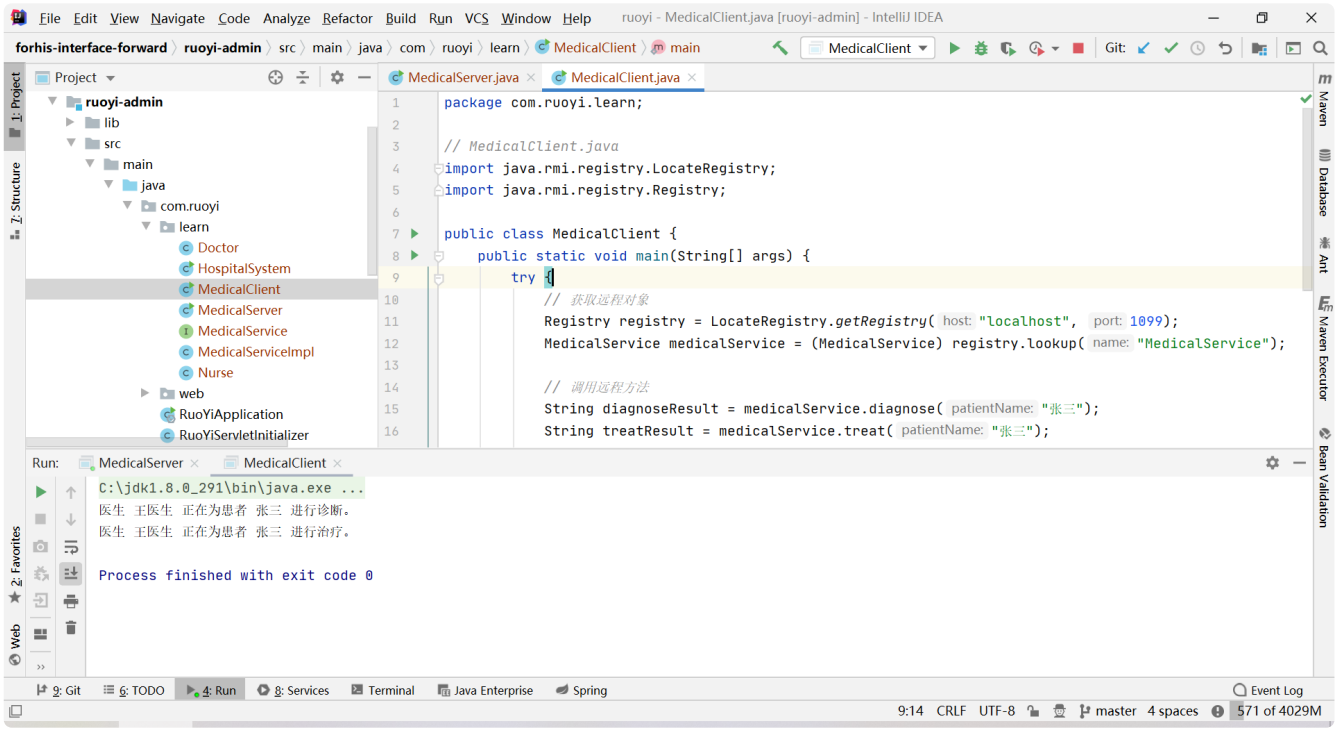

图 4 运行结果

运行医院管理系统，如图 5、图 6 所示：

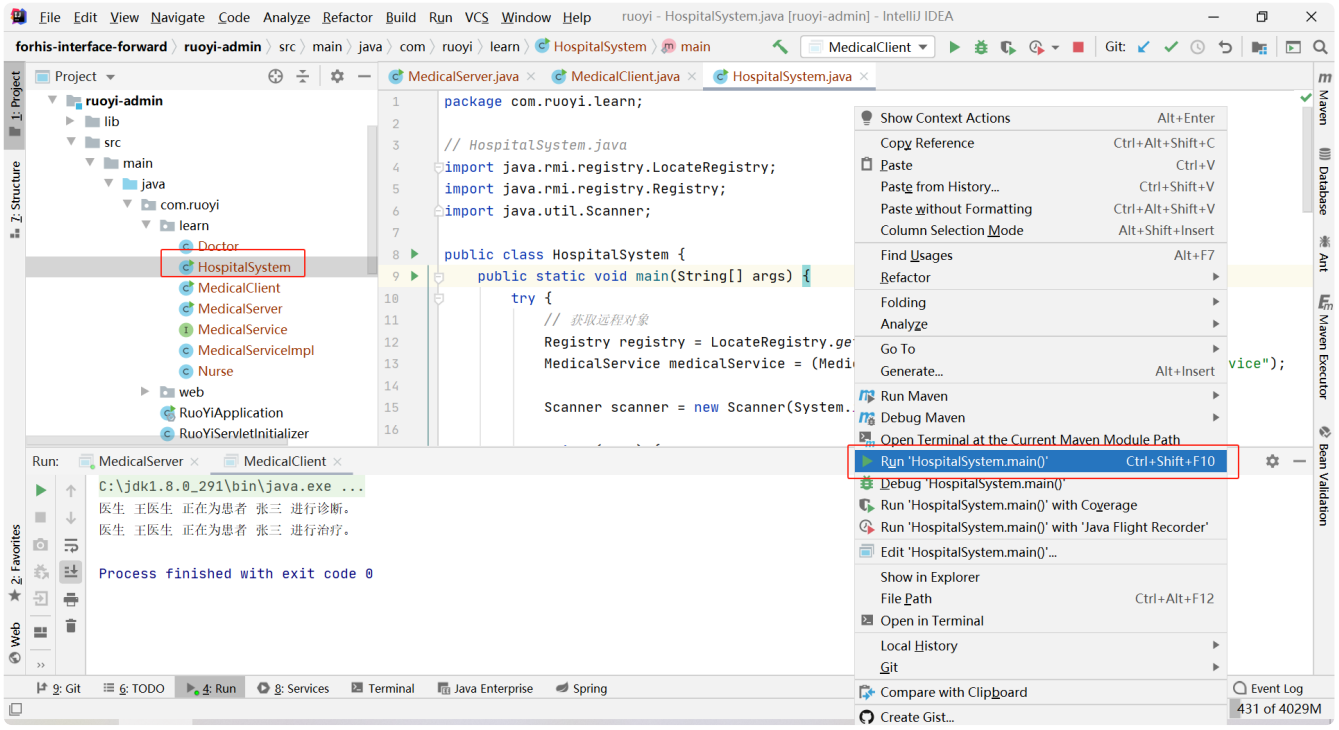

图 5 找到启动按钮

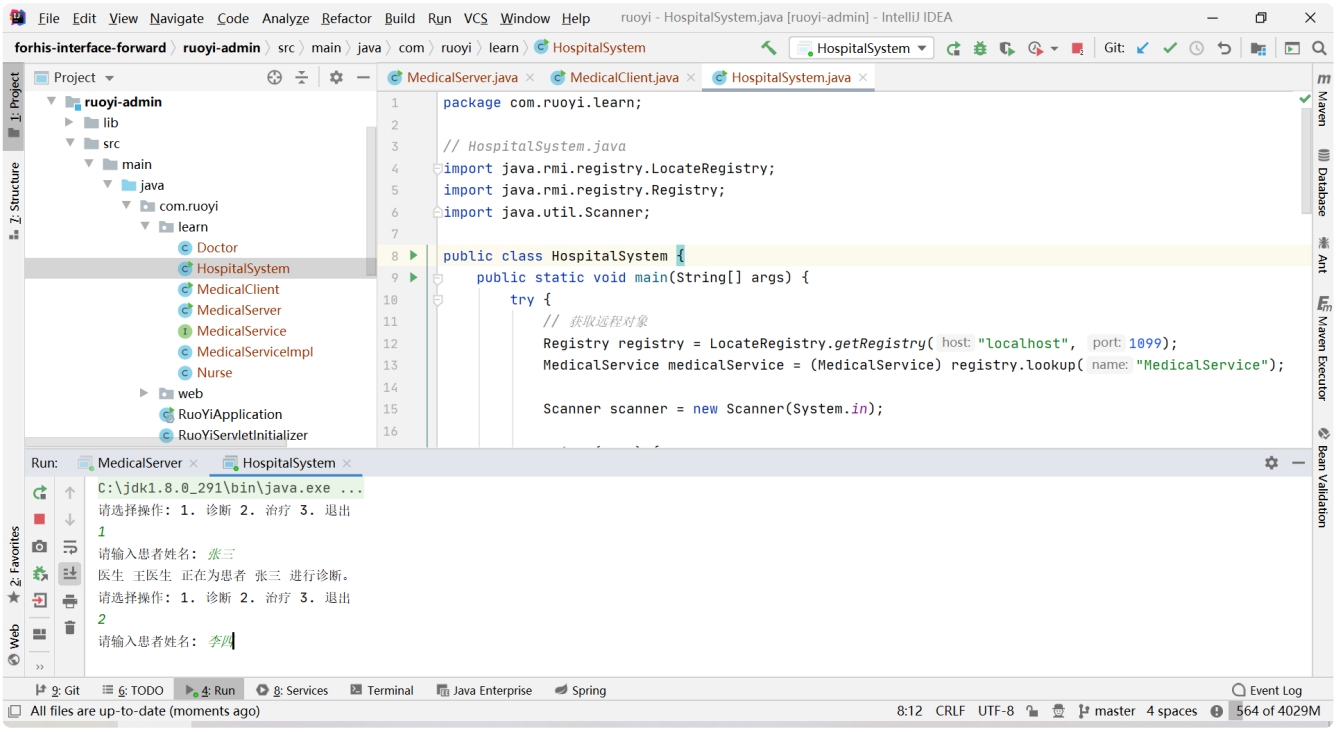

图 6 运行结果

七、实验考核

本实验考核采用【实验随堂查】方式开展。

每个实验完成后，在实验课上通过现场演示的方式向实验指导教师进行汇报，并完成现场问答交流。

每个实验考核满分 100 分，其中实验成果汇报 60 分，现场提问交流 40 分。

实验考核流程：

（1）学生演示汇报实验内容的完成情况，实验指导老师现场打分。

（2）指导老师结合实验内容进行提问，每位学生提问 2-3 个问题，根据回答的情况现场打分。

（3）实验考核结束后，进行公布成绩。