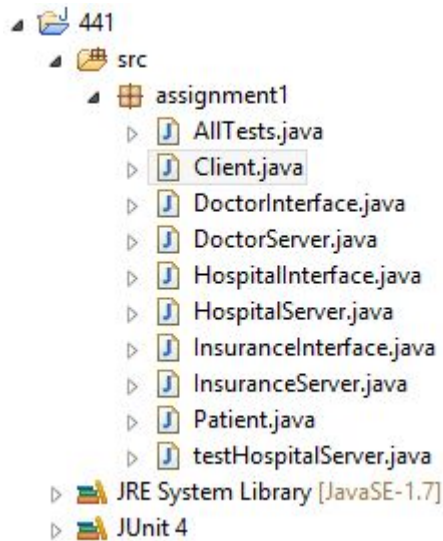


INSTALLATION AND RUNNING INSTRUCTIONS:

1. Extract the zip folder named Asg1_kgarg3.zip in a folder in your system.
2. Go to the directory where the Asg1_kgarg3 is extracted. Lets say the folder is C:/kgarg3/Asg1_kgarg3
3. Open Eclipse editor in java standard edition perspective and make a new java project named Asg1_kgarg3.
4. A new folder structure will be created when you click “finish”.
5. Right Click on “src” and a drop-down menu will be displayed.
6. Click on “Import”
7. A Import dialogue box will pop-up and select : General->File System and click on “next”.
8. Browse to the directory C:/kgarg3/Asg1_kgarg3/src, and select all the java files

Ensure the the java files are under a package named - assignment1 like this:

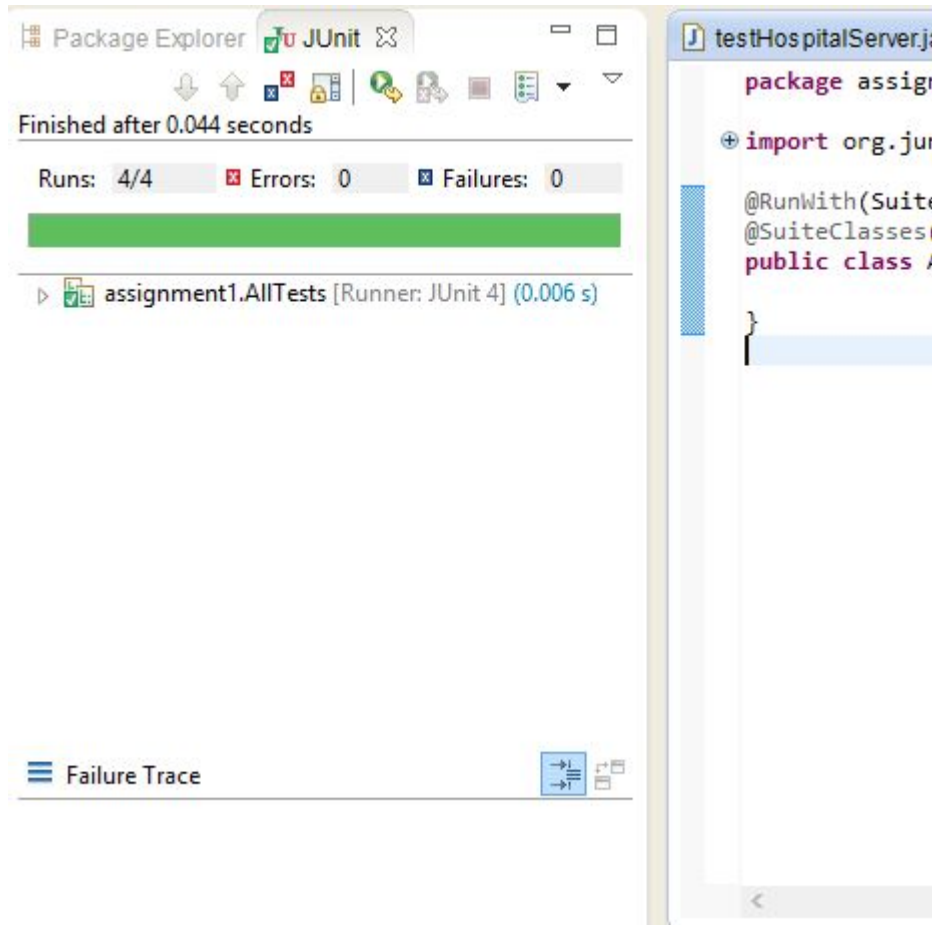


9. Make sure to attach the JUnit 4 Jar file in the external references in the java build path.
10. Make sure to build the project using Project -> Build All.
11. You need to compile and run the java files in this order: 1. InsuranceServer.Java, 2. DoctorServer.Java, 3. HospitalServer.Java, and 4. Client.Java.

SAMPLE RUN:

Assumption: You have proper java files and Port number - 1099 should be free on the machine on which you are running this application.

1. Select AllTests.java and Run AllTests. You will see the following screen on the left-hand side if all the tests are successfully passed.



2. Select InsuranceServer.java and Run InsuranceServer (using Green button of Play in eclipse), and you will see in console window: **Insurance Server Ready!**

3. Select DoctorServer.java and Run DoctorServer (using Green button of Play in eclipse), and you will see in console window: **Doctor Server Ready!**

4. Select HospitalServer.java and Run HospitalServer (using Green button of Play in eclipse), and you will see in console window: **Hospital Server Ready!**

5. Select Client.java and Run Client (using Green button of Play in eclipse), and you will see in console window: Enter 1 to see the list of hospitals.

6. Type the number "1" and hit enter.

6. You will see this:

-----LIST OF HOSPITALS-----

- 1-Rush Hospital
- 2-Noble Hospital
- 3-Family Medical Center
- 4-UIC Hospital

Select one hospital. Enter the valid number.

7. Type the number “3” and hit enter

8. You will see this:

-----LIST OF DOCTORS-----

- Dr. Bell
- Dr. Turan
- Dr. Cheek

Select a doctor. Enter the name of the doctor.

9. Type “Dr. Turan” and hit Enter.

10. You will see this: Wait for doctor's appointment confirmation. Your details are saved. Enter 1 to see the list of hospitals.

11. Type the number “1” and hit enter

12. You will see this:

-----LIST OF HOSPITALS-----

- 1-Rush Hospital
- 2-Noble Hospital
- 3-Family Medical Center
- 4-UIC Hospital

Select one hospital. Enter the valid number.

13. Type the number “3” and hit enter

14. You will see this:

-----LIST OF DOCTORS-----

- Dr. Bell
- Dr. Turan
- Dr. Cheek

Select a doctor. Enter the name of the doctor.

15. Type “Dr. Bell” and hit Enter.

The following will be the output on different server consoles:

Client Console:

```
<terminated> Client (2) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (Sep 30, 2014, 2:26:24 AM)
1-Rush Hospital
2-Noble Hospital
3-Family Medical Center
4-UIC Hospital
Select one hospital. Enter the valid number.
3
-----LIST OF DOCTORS-----
Dr. Bell
Dr. Turan
Dr. Cheek
Select a doctor. Enter the name of the doctor.
Dr. Turan
Wait for doctor's appointment confirmation. Your details are saved.
Enter 1 to see the list of hospitals.
1
-----LIST OF HOSPITALS-----
1-Rush Hospital
2-Noble Hospital
3-Family Medical Center
4-UIC Hospital
Select one hospital. Enter the valid number.
3
-----LIST OF DOCTORS-----
Dr. Bell
Dr. Turan
Dr. Cheek
Select a doctor. Enter the name of the doctor.
Dr. Bell
Wait for doctor's appointment confirmation. Your details are saved.

REFERENTIAL INTEGRITY ON THE LOCAL MACHINE WHERE THE OBJECTS ARE CREATED!
Patient object 1 created on Client Machine [ Name: Kanishka ID: 11 Doctor's name: Dr. Turan ]
Patient object 2 created on Client Machine [ Name: Kanishka ID: 11 Doctor's name: Dr. Turan ]
-----
Patient object 3 created on Client Machine [ Name: John ID: 21 Doctor's name: Dr. Bell ]
hello, patients
```

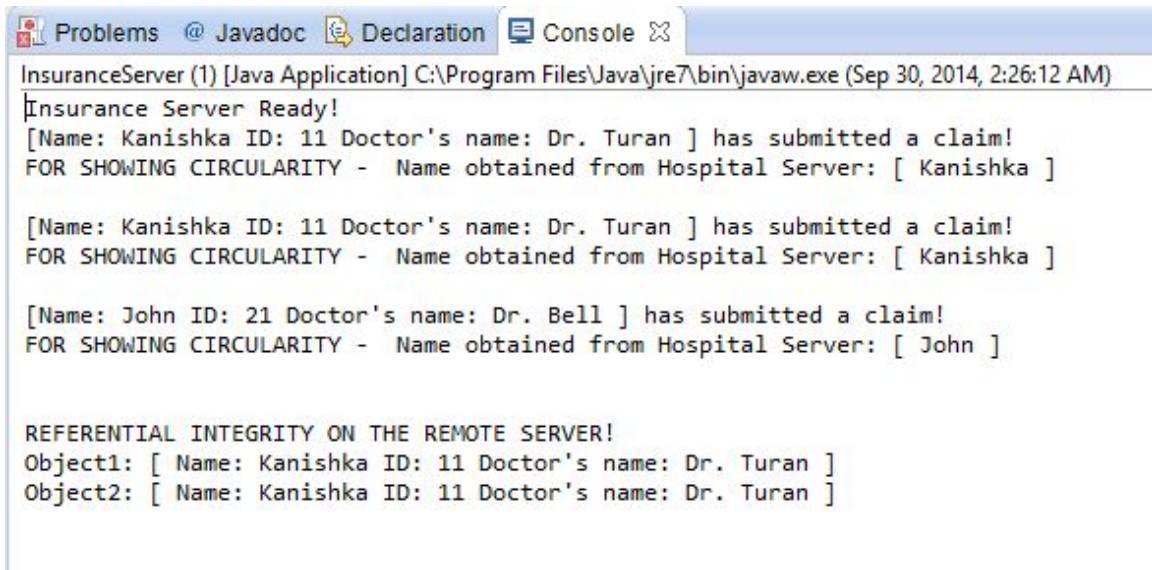
Hospital Server Console:

```
HospitalServer (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (Sep 30, 2014, 2:26:21 AM)
Hospital Server Ready!
Details Sent to Doctor for [ Name: Kanishka ID: 11 Doctor's name: Dr. Turan ]
Details Sent to Doctor for [ Name: Kanishka ID: 11 Doctor's name: Dr. Turan ]
Details Sent to Doctor for [ Name: John ID: 21 Doctor's name: Dr. Bell ]
```

Doctor Server Console:

```
DoctorServer (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (Sep 30, 2014, 2:26:17 AM)
Doctor Server Ready!
Claim submitted to insurance company for : [ Name: Kanishka ID: 11 Doctor's name: Dr. Turan ]
Claim submitted to insurance company for : [ Name: Kanishka ID: 11 Doctor's name: Dr. Turan ]
Claim submitted to insurance company for : [ Name: John ID: 21 Doctor's name: Dr. Bell ]
```

Insurance Server Console:



```
InsuranceServer (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (Sep 30, 2014, 2:26:12 AM)
[Insurance Server Ready!
[Name: Kanishka ID: 11 Doctor's name: Dr. Turan ] has submitted a claim!
FOR SHOWING CIRCULARITY - Name obtained from Hospital Server: [ Kanishka ]

[Name: Kanishka ID: 11 Doctor's name: Dr. Turan ] has submitted a claim!
FOR SHOWING CIRCULARITY - Name obtained from Hospital Server: [ Kanishka ]

[Name: John ID: 21 Doctor's name: Dr. Bell ] has submitted a claim!
FOR SHOWING CIRCULARITY - Name obtained from Hospital Server: [ John ]

REFERENTIAL INTEGRITY ON THE REMOTE SERVER!
Object1: [ Name: Kanishka ID: 11 Doctor's name: Dr. Turan ]
Object2: [ Name: Kanishka ID: 11 Doctor's name: Dr. Turan ]
```

CONCLUSION:

1. We are passing two object references (which points to the same object) from client to the hospital server using RMI, which in turn passes these two references to doctor server using RMI (in which case, hospital server acts as a client to doctor server), which in turn contacts insurance server and passes these two references again (in which case, doctor server acts as a client to the insurance server.). The principle of **REFERENTIAL INTEGRITY** is maintained and the objects are passed by copy (rather than pass-by-reference as done in a local method call.)
2. In order to observe the **circularity among servers**, when a patient object reaches insurance server from the doctor server, the name of the patient is validated in the records of the hospital. Validation is done in a way that Insurance Server calls a remote method call on hospital server.