United States International University

MIS 6060: DISTRIBUTED COMPUTING & INTERNET TECHNOLOGY

Lab Exercise 4: A Remote Method Invocation (RMI) based application program to implement a client/server time service using Java RMI.

Objective

To write a program that shows object communication using RMI.

Overview - RMI is the infrastructure for a program on one computer to execute methods (procedures) located on another over a network. Methods may pass/receive parameters and return results, the calling program must normally wait until the called method completes execution as it would with a standard method call. Using RMI to build a service has advantages in that communication between the client and server consists of passing objects as parameters rather than text in a message. All the expressiveness of objects can be utilized (there are some restrictions such as complete thread state can't be passed) so that a linked list could be passed as a parameter rather than converted into text, transmitted, and converted back into a linked list on the receiving end. One disadvantage is that RMI only works with Java because other languages and computers represent data differently, a large problem for developing general services.

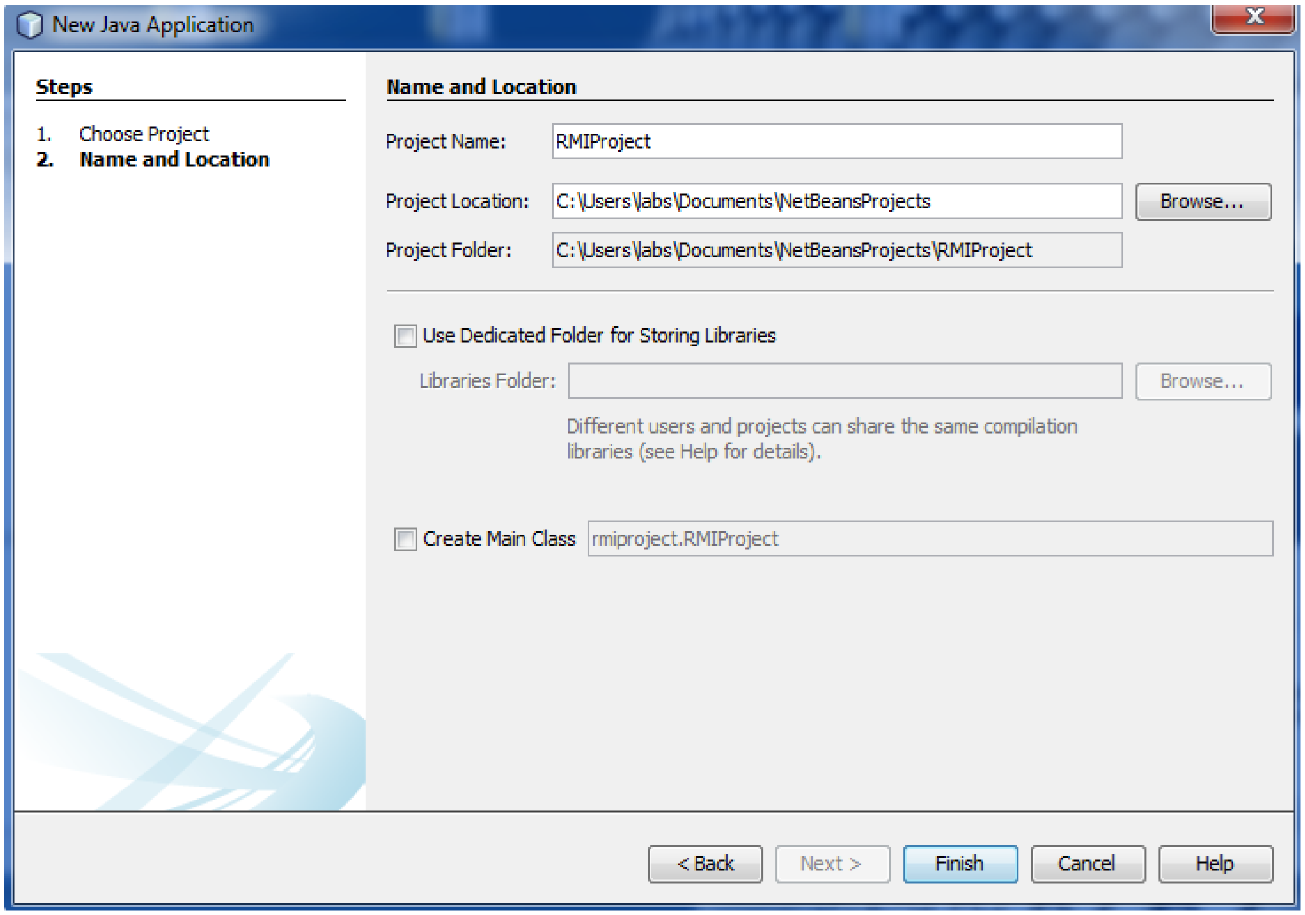
Requirements

We will implement a simple *Time* service client/server using Java RMI. Clients obtain the current time by invoking a method *getTime( )* on the remote time server. There are three Java files necessary to implement a client/server relation using RMI:

1. Interface - *TimeServerInf.java* - Interface definition of all server methods that can be called by the remote client. The server implementation must define *String getTime( )* method.
2. Server - *TimeServerImpl.java* - Implements the methods that are defined in the interface (i.e. *String getTime( )*).
3. Client - *TimeClient.java* - Client calls to remote methods on the server, (i.e. *getTime( )*).

Step 1

Run the NetBeans program and create a new project. Name the project RMIProject.



New Java Application

Steps

1. Choose Project
2. **Name and Location**

Name and Location

Project Name: RMIProject

Project Location: C:\Users\labs\Documents\NetBeansProjects

Project Folder: C:\Users\labs\Documents\NetBeansProjects\RMIProject

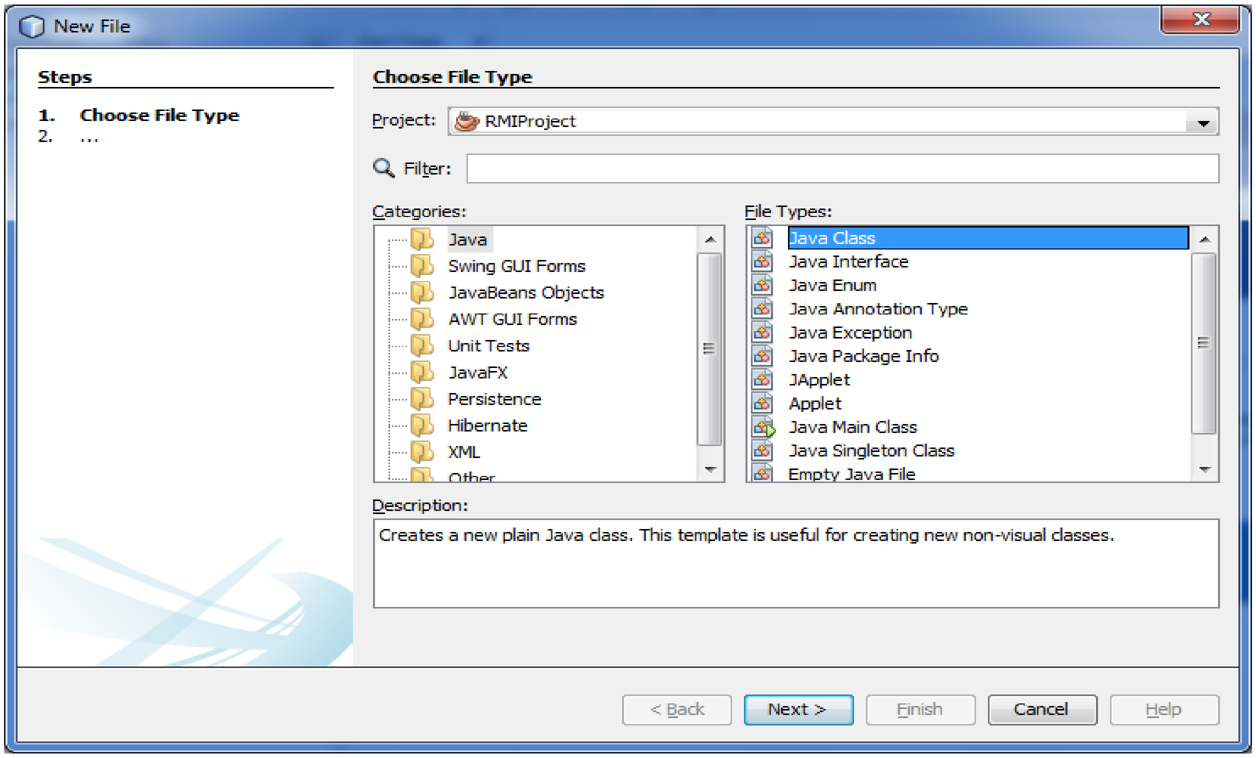
☐ Use Dedicated Folder for Storing Libraries

Libraries Folder:

Different users and projects can share the same compilation libraries (see Help for details).

☐ Create Main Class rmiproject.RMIProject

Create two new java classes and name them **TimeClient** and **TimeServerImpl**.


New File

Steps
1. Choose File Type
2. ...

Choose File Type

Project: RMIProject

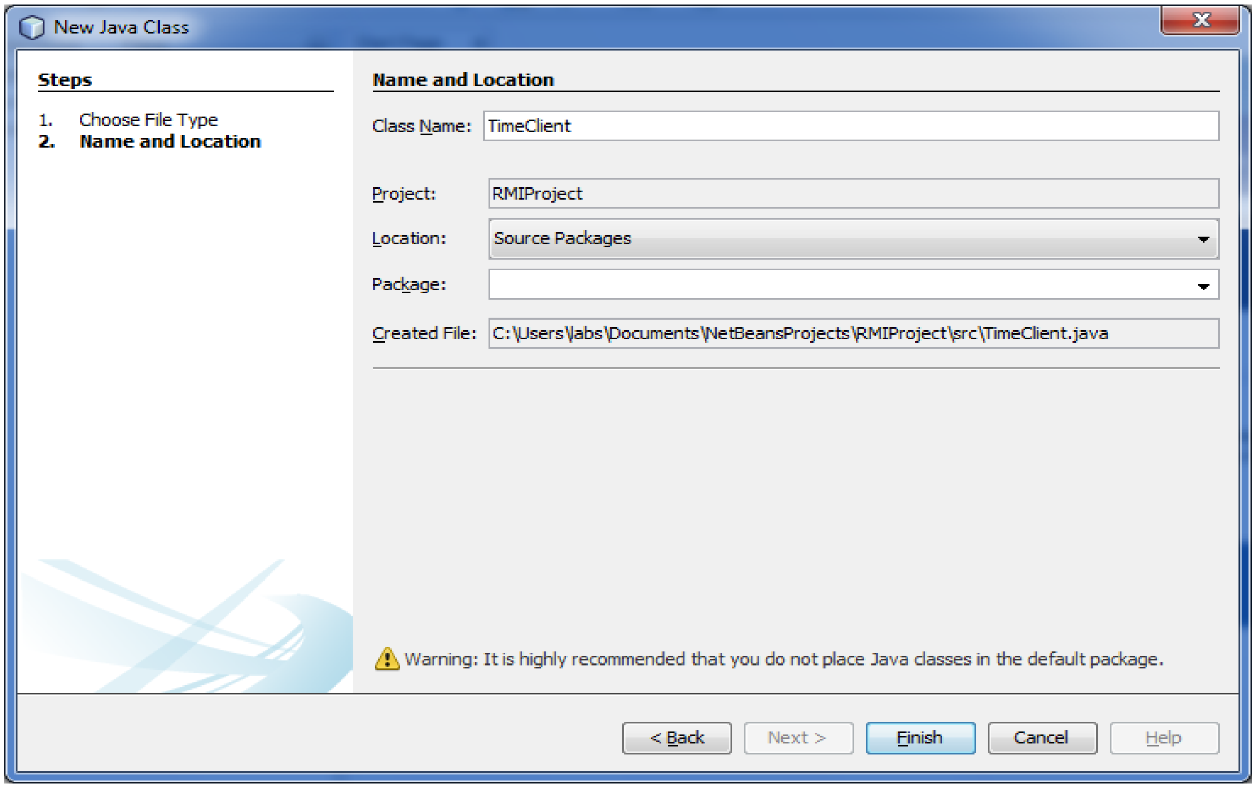
Filter:

Categories:
- Java
- Swing GUI Forms
- JavaBeans Objects
- AWT GUI Forms
- Unit Tests
- JavaFX
- Persistence
- Hibernate
- XML
- Other

File Types:
- Java Class
- Java Interface
- Java Enum
- Java Annotation Type
- Java Exception
- Java Package Info
- JApplet
- Applet
- Java Main Class
- Java Singleton Class
- Empty Java File

Description:
Creates a new plain Java class. This template is useful for creating new non-visual classes.

< Back | Next > | Finish | Cancel | Help


New Java Class

Steps
1. Choose File Type
2. Name and Location

Name and Location

Class Name: TimeClient

Project: RMIProject

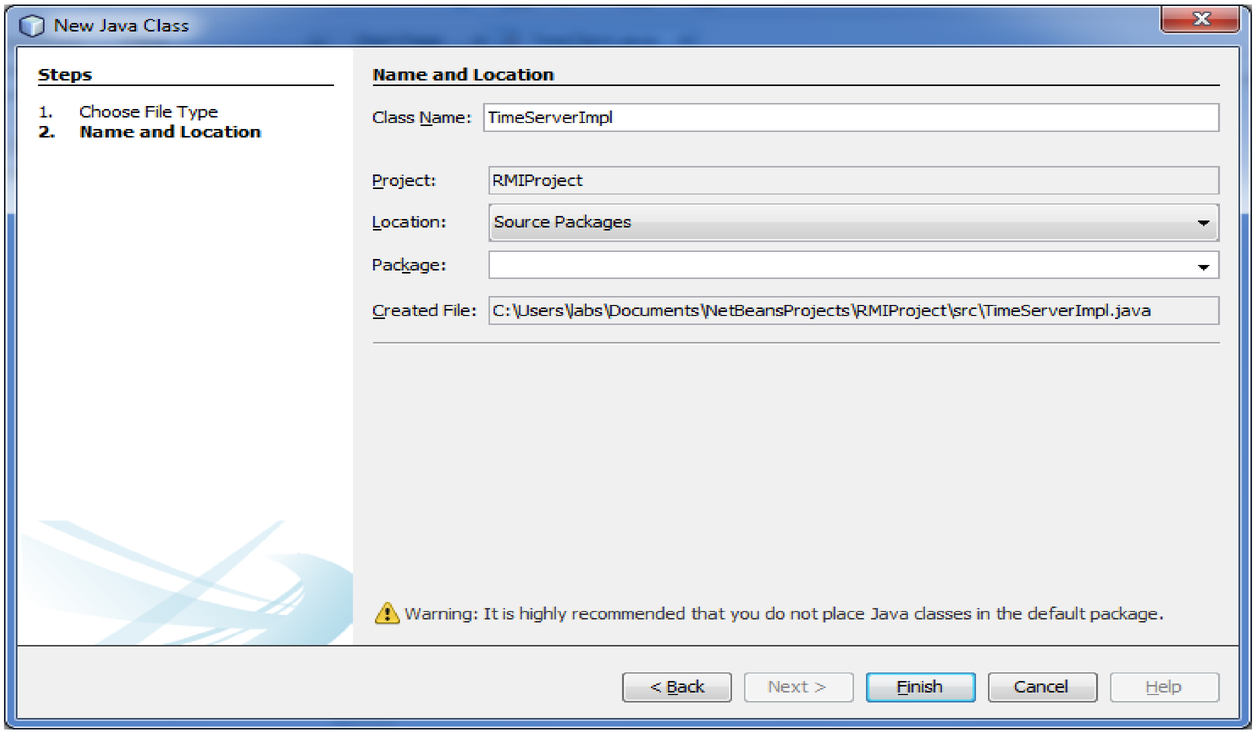
Location: Source Packages

Package:

Created File: C:\Users\labs\Documents\NetBeansProjects\RMIProject\src\TimeClient.java

Warning: It is highly recommended that you do not place Java classes in the default package.

< Back | Next > | Finish | Cancel | Help



Class Name: TimeServerImpl

Project: RMIProject

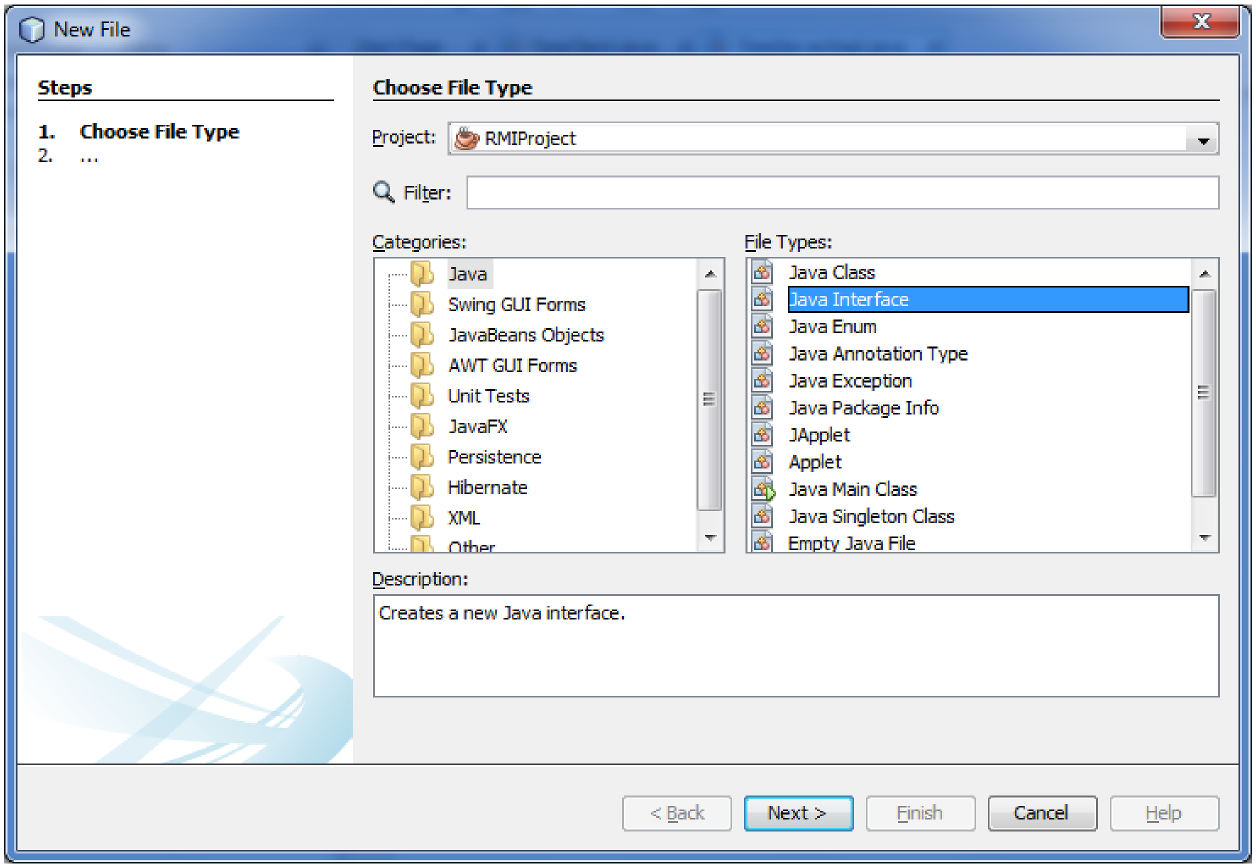
Location: Source Packages

Package:

Created File: C:\Users\labs\Documents\NetBeansProjects\RMIProject\src\TimeServerImpl.java

Warning: It is highly recommended that you do not place Java classes in the default package.

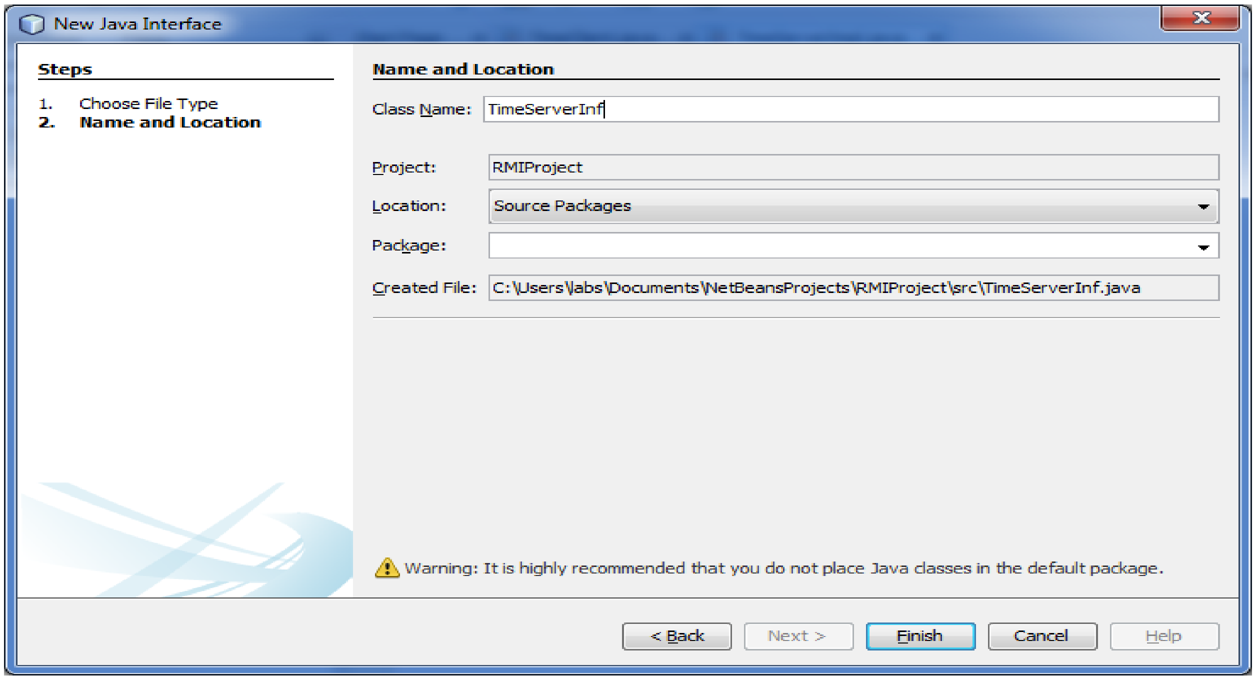
Create a new java interface and name it **TimeServerInf**.



Project: RMIProject

Description:

Creates a new Java interface.



Name and Location

Class Name: TimeServerInf

Project: RMIProject

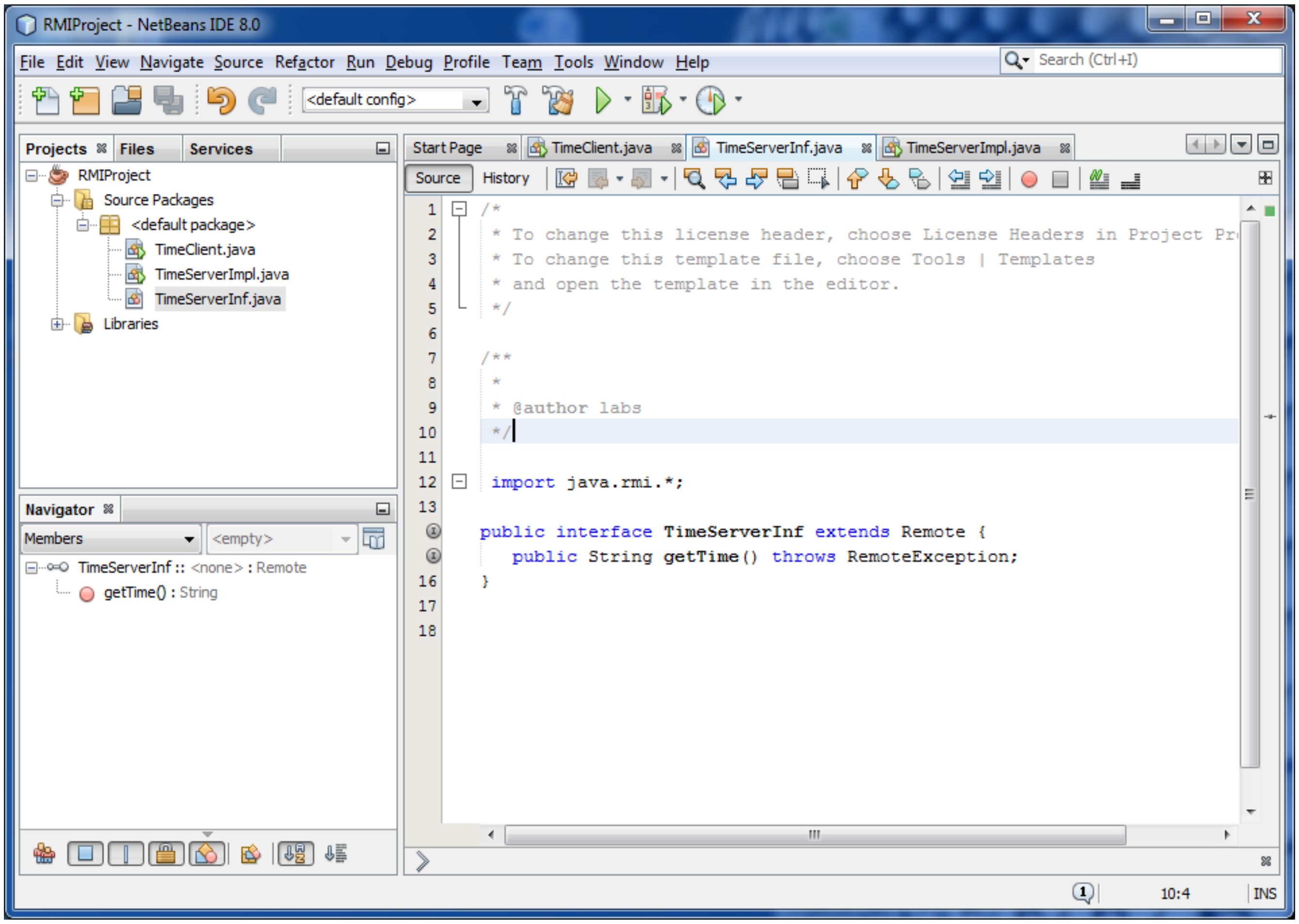
Location: Source Packages

Package:

Created File: C:\Users\labs\Documents\NetBeansProjects\RMIProject\src\TimeServerInf.java

Warning: It is highly recommended that you do not place Java classes in the default package.

The three files should look as shown below:



```java
/*
 * To change this license header, choose License Headers in Project Pr
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 *
 * @author labs
 */

import java.rmi.*;

public interface TimeServerInf extends Remote {
   public String getTime() throws RemoteException;
}
```

Step 2

Double click on the TimeServerinf.java file to open it. Insert the following code…

TimeServerInf.java code

```
* @author labs
 */

 import java.rmi.*;

public interface TimeServerInf extends Remote {
  public String getTime() throws RemoteException;
}
```

Step 2

Double click on the TimeServerImpl.java file to open it. Insert the following code

TimeServerImpl.java code

```
* @author labs
 */

import java.rmi.*;
import java.rmi.server.*;


public class TimeServerImpl extends UnicastRemoteObject implements TimeServerInf {

  public TimeServerImpl() throws RemoteException { super();  }

  // implementation for TimeServerInf interface method
  public String getTime()  {
    try {  return java.net.InetAddress.getLocalHost() + " " + new java.util.Date().toString(); }
    catch(Exception e) { return "Failed"; }
  }

  public static void main( String args[] ) throws Exception {
    System.err.println( "Initializing server: please wait." );

    // create server object and bind TimeServerImpl object to the rmiregistry
    Naming.rebind( "//localhost/Time", new TimeServerImpl() );

    System.err.println("The Time Server is up and running." );
  }
}
```

Step 3

Double click on the TimeClient.java file to open it. Insert the following code…

TimeClient.java code

```
* @author labs
 */

import java.rmi.*;

public class TimeClient {

   public static void main( String args[] ) throws Exception {

        String host = "localhost";
        if (args.length > 0) host = args[0];

        // lookup TimeServerInf remote object in rmiregistry
        TimeServerInf ts = (TimeServerInf) Naming.lookup( "//" + host + "/Time" );

        // get time from server
        System.out.println(ts.getTime());
   }
}
```

Step 4

Run the Server file then the client file. The output should be as below…

Output

TimeClient Output

```
The time is: Tue Jun 19 11:30:36 EDT 2014
```