Remote Method Invocation (RMI)

Section 1

BY: sara Elhamouly

Introduction

- Remote Method Invocation (RMI) is an API that provides a mechanism to create distributed applications in java.
- Using RMI, a Java object on one system can invoke a method in an object on another system on the network.
- All the objects you have used before are called **local objects.** Local objects are accessible only within the local host.

Remote object

For an object to be invoked remotely:

- It must be defined in a Java interface accessible to both the server and the client.
- Must extend the **java.rmi.Remote** interface.

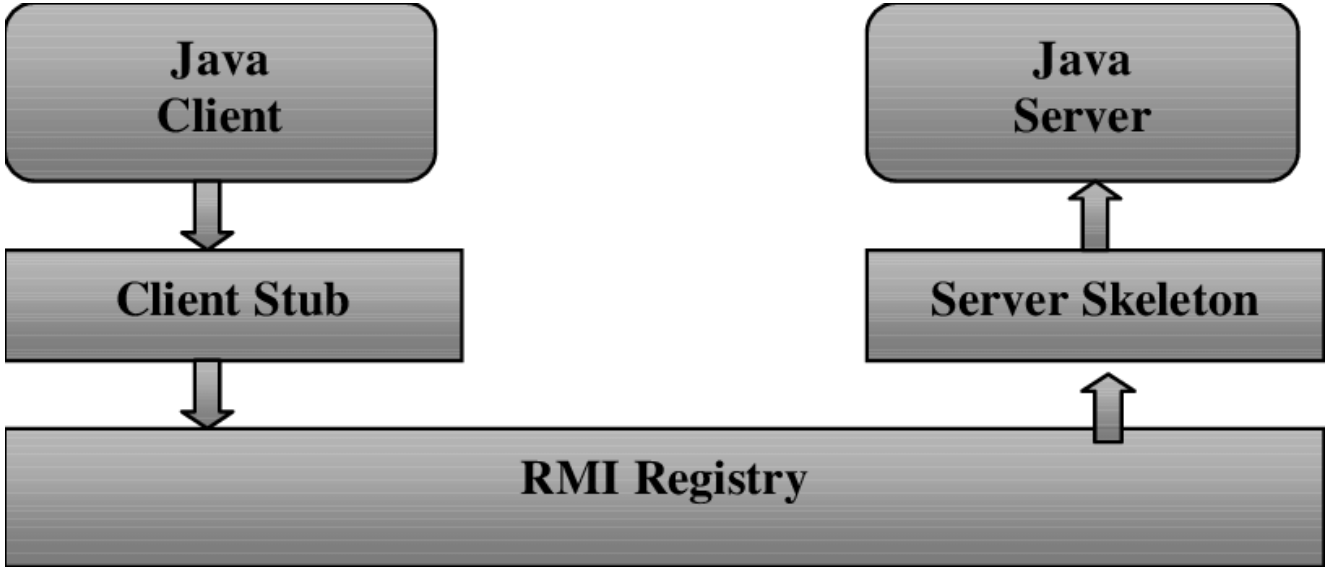
RMI architecture

- **Server object interface:** A subinterface of java.rmi.Remote that defines the methods for the server object.
- **Server class:** A class that implements the remote object interface.
- **Server object:** An instance of the server class.
- **RMI registry:** A utility that registers remote objects and provides naming services for locating objects.
- **Client program:** A program that invokes the methods in the remote server object.

RMI architecture

- **Stub object:** is responsible for sending parameters to the server and for receiving the result from the server and returning it to the client.
- **Skeleton object:** communicates with the stub on the server side. The skeleton receives parameters from the client, passes them to the server for execution, and returns the result to the stub.

RMI architecture



Implementation

- 1- create two java applications (client and server)
- 2- In server application
  - 1- Define Server Object Interface
  - 2- Define Server Implementation Class
  - 3- Create and Register Server Object
  - 4- build and run the the server app
  - 5- add Server.jar which created in Server/dist to the client’s libraries.
- 3- Develop Client Program
- 4- Run Server then run Client

Server Object Interface

```
public interface Server_Interface extends Remote{
    public int sum (int num1, int num2) throws Exception;
}
```

Server Implementation Class

```
public class Server_Class extends UnicastRemoteObject implements
Server_Interface{
    public Server_Class() throws Exception{}
    @Override
    public int sum(int x, int y) throws Exception {
        return x+y;
    }
}
```

Server Implementation Class

- UnicastRemoteObject class provides support for point-to-point active object references using TCP streams.

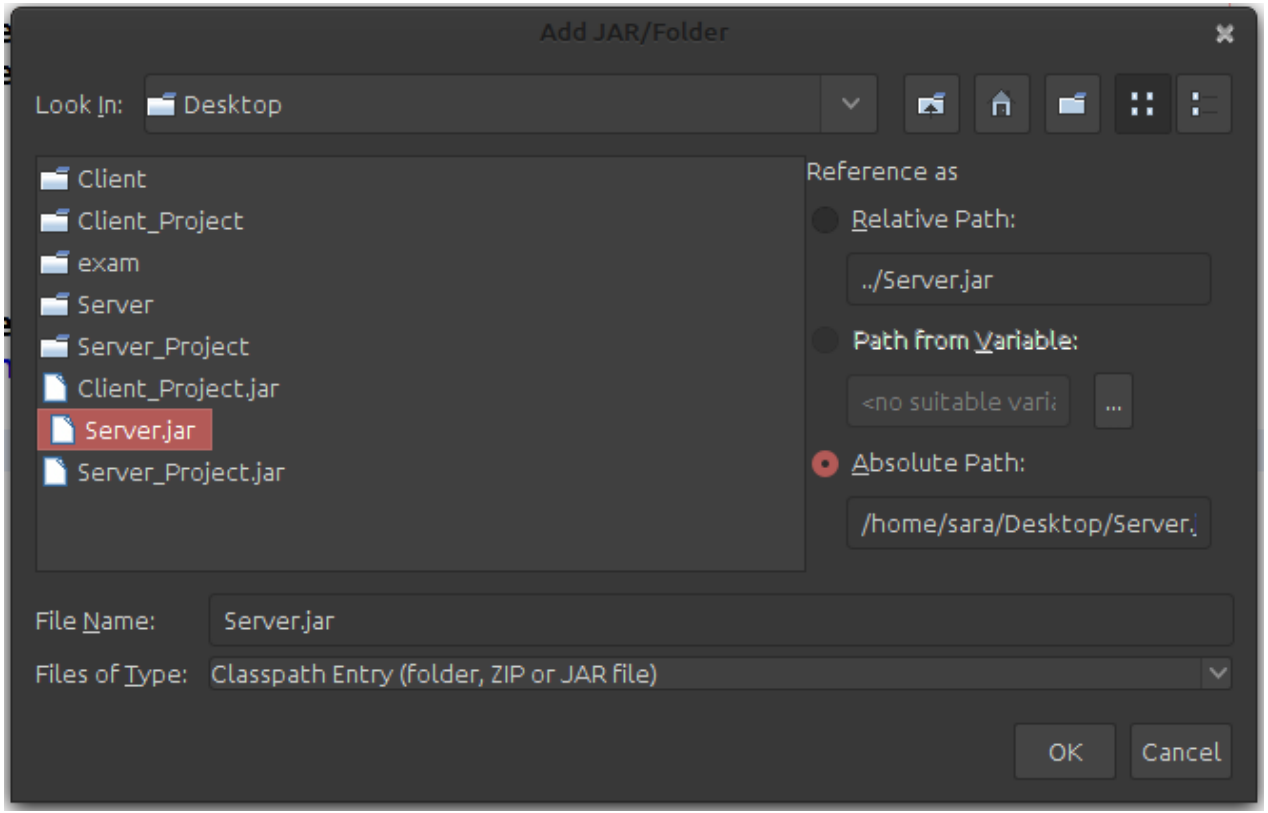
Create and Register Server Object

```
public class Run_Server {
    public static void main(String[] args) throws Exception{
        Server_Class server_object = new Server_Class();
        Registry registry = LocateRegistry.createRegistry(5000);
        registry.bind("Server_object1", server_object);
    }
}
```

build and run server app

- build and run server app

add Server.jar to client's libraries



Develop Client Program

```
import server.Server_Interface;
public class Client {
    public static void main(String[] args) throws Exception{
        Server_Interface server_object;
        Registry registry = LocateRegistry.getRegistry("localhost",5000);
        server_object = (Server_Interface)registry.lookup("Server_object1");
        System.out.println(server_object.sum(5, 3));
    }
}
```

Run

- Run server
- Then Run client