Assignment 2

Implement Remote Method Invocation (RMI) by checking if "**RMServer**" already exists on the registry.

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

public interface RMInterface extends Remote {
String sayHello() throws RemoteException;
}
```

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

public class RMImplementation extends UnicastRemoteObject implements
RMInterface {
public RMImplementation() throws RemoteException {
super();
}

@Override
public String sayHello() throws RemoteException {
return "Hello from the remote server!";
}
}
```

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

public class RMServer {
public static void main(String[] args) {
try {
RMImplementation rmImplementation = new RMImplementation();
Registry registry = LocateRegistry.createRegistry(1099);
```

```
// Bind the remote object to the registry
registry.rebind("RMServer", rmImplementation);

System.out.println("RMServer is ready to accept requests.");
} catch (Exception e) {
System.err.println("RMServer exception: " + e.toString());
e.printStackTrace();
}
}
}
```

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

public class RMClient {
public static void main(String[] args) {
try {
Registry registry = LocateRegistry.getRegistry("localhost", 1099);

// Look up the remote object from the registry
RMInterface rmInterface = (RMInterface) registry.lookup("RMServer");

String result = rmInterface.sayHello();
System.out.println("Response from server: " + result);
} catch (Exception e) {
System.err.println("RMClient exception: " + e.toString());
e.printStackTrace();
}
}
}
```

Output

```
➜ a2 java RMServer.java
RMServer is ready to accept requests.
```

```
➜ a2 java RMClient
Response from server: Hello from the remote server!
```