**DISTRIBUTED COMPONENTS LABORATORY**

**EXERCISE 3 – JAVA RMI**

**Aim:**

To study about the concepts of JAVA RMI and to implement a simple application using the same.

**1. How does the Java RMI differ from CORBA?**

|  |  |
| --- | --- |
| **JAVA RMI** | **COBRA** |
| Java only | Language independent Can access foreign language objects |
| Interfaces specified in Java | Interfaces specified externally in IDL |
| Pass objects by remote reference or by value (making local copy) | Pass objects by reference (by value possible but awkward) |
| Distributed garbage collection integrated with local collectors | Remote references manually released |
| Generally simpler to use | More complicated |

**2. List out the differences between RPC and RMI.**

|  |  |  |
| --- | --- | --- |
| **BASIS** | **RPC** | **RMI** |
| Supports | Procedural programming | Object-oriented programming |
| Parameters | Ordinary data structures are passed to remote procedures. | Objects are passed to remote methods. |
| Efficiency | Lower than RMI | More than RPC and supported by modern programming approach (i.e. Object-oriented paradigms) |
| Overheads | More | Less comparatively |
| In-Out parameters are mandatory | Yes | Not necessarily needed |
| Provision of ease of programming | High | Low |

**3. What does the following statement indicate? Naming.rebind ("//rmi:3000/ HelloRegistry ", object1);**

The code Naming.rebind(“RMI\_INSTANCE”, implclass); informs RMI registry that impl class object is available with the name “RMI\_INSTANCE”

**About the rebind function:**

public static void rebind([String](https://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.lang.String.html#_top_) name, [Remote](https://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.rmi.Remote.html#_top_) obj) throws [RemoteException](https://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.rmi.RemoteException.html#_top_), [MalformedURLException](https://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.net.MalformedURLException.html#_top_), [UnknownHostException](https://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.rmi.UnknownHostException.html#_top_)

Rebind the name to a new object; replaces any existing binding.

**Throws:** [RemoteException](https://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.rmi.RemoteException.html#_top_) If registry could not be contacted.

**4. What is the default port number used by RMI Registry?**

By default, the RMI Registry uses port **1099**. Client and server (stubs, remote objects) communicate over random ports unless a fixed port has been specified when exporting a remote object. The communication is started via a socket factory which uses 0 as starting port, which means use any port that's available between 1 and 65535.

**5. What is the difference between bind () and rebind () methods of Naming Class?**

**bind** method (String name) binds the specified name to a remote object while **rebind** (String

name) method rebinds the specified name to a new remote object, Any existing binding

for the name is replaced.

About the function:

**Bind:**

public static void bind([String](https://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.lang.String.html#_top_) name, [Remote](https://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.rmi.Remote.html#_top_) obj) throws [AlreadyBoundException](https://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.rmi.AlreadyBoundException.html#_top_), [MalformedURLException](https://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.net.MalformedURLException.html#_top_), [UnknownHostException](https://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.rmi.UnknownHostException.html#_top_), [RemoteException](https://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.rmi.RemoteException.html#_top_)

Binds the name to the specified remote object.

**Throws:** [RemoteException](https://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.rmi.RemoteException.html#_top_)

If registry could not be contacted.

**Throws:** [AlreadyBoundException](https://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.rmi.AlreadyBoundException.html#_top_)

If name is already bound.

**Rebind:**

public static void rebind([String](https://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.lang.String.html#_top_) name, [Remote](https://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.rmi.Remote.html#_top_) obj) throws [RemoteException](https://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.rmi.RemoteException.html#_top_), [MalformedURLException](https://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.net.MalformedURLException.html#_top_), [UnknownHostException](https://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.rmi.UnknownHostException.html#_top_)

Rebind the name to a new object; replaces any existing binding.

**Throws:** [RemoteException](https://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java.rmi.RemoteException.html#_top_)

If registry could not be contacted.

**6. Implement Print Hello Program using Java RMI and include the screenshot**

**Java code:**

**RMI\_Interface.java:**

public interface RMI\_Interface extends java.rmi.Remote

{

    public String print() throws java.rmi.RemoteException;

}

**RMI\_Impl.java:**

import java.rmi.\*;

import java.rmi.server.UnicastRemoteObject;

public class RMI\_Impl extends UnicastRemoteObject implements RMI\_Interface

{

    private static final long serialVersionUID = 1L;

    public RMI\_Impl() throws RemoteException

    {

    }

    public String print()

    {

        return new String("Helloworld!");

    }

}

**RMI\_Server.java:**

import java.rmi.\*;

public class RMI\_Server {

    public static void main(String args[])

    {

        try

        {

            RMI\_Impl implclass=new RMI\_Impl();

            Naming.rebind("RMI\_INSTANCE",implclass);

        }

        catch (Exception e)

        {

            System.out.println("Exception occured:" + e);

        }

    }

}

**RMI\_Client.java:**

import java.rmi.\*;

public class RMI\_Client

{

    public static void main(String[] argv)

    {

        if (argv.length !=1)

        {

            System.out.println("Usage: java RMI\_Client &lt;IP addr of host running RMI Server");

            System.exit(0);

        }

        String serverName = argv[0];

        try

        {

            RMI\_Interface myserverobject = (RMI\_Interface)Naming.lookup("rmi://"+serverName+"/RMI\_INSTANCE");

            String s = myserverobject.print();

            System.out.println("Server says: " + s);

        }

        catch(Exception e)

        {

            System.out.println("Exception Occured " + e);

            System.exit(0);

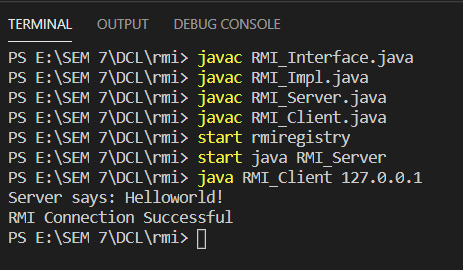
        }

        System.out.println("RMI Connection Successful");

    }

}

**OUTPUT:**

****

**Result:**

The concepts about JAVA RMI is studied and a simple application is executed using that successfully.