Ex.No.3	Implementation of RMI in VMs

Aim:

To run RMI using java in Ubuntu on VMWare and Oracle VM Virtual Box and compare their performance.

Procedure:

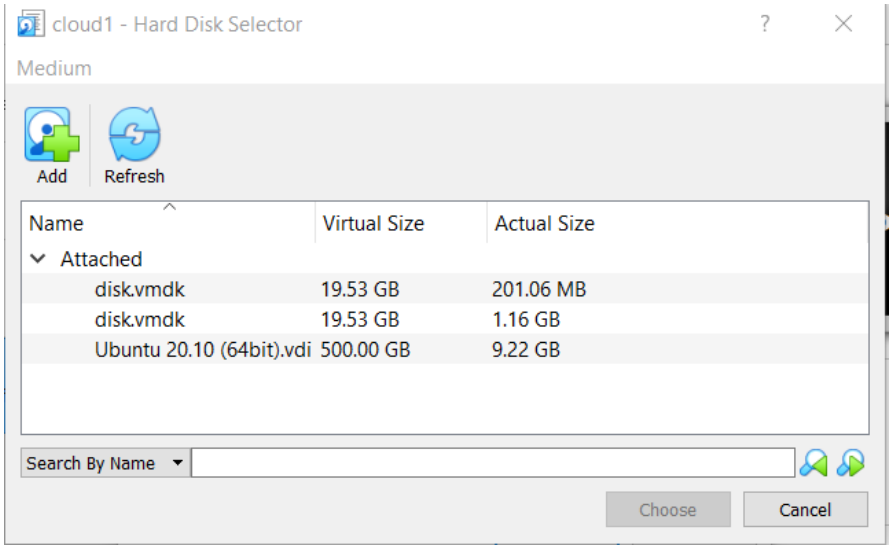
Oracle VM Virtual Box

1. Install Oracle VM VirtualBox in your machine.
2. Download ubuntu VirtualBox file from the website.

https://www.osboxes.org/ubuntu/

3. Extract the vdi file

4. Open the image through the VirtualBox

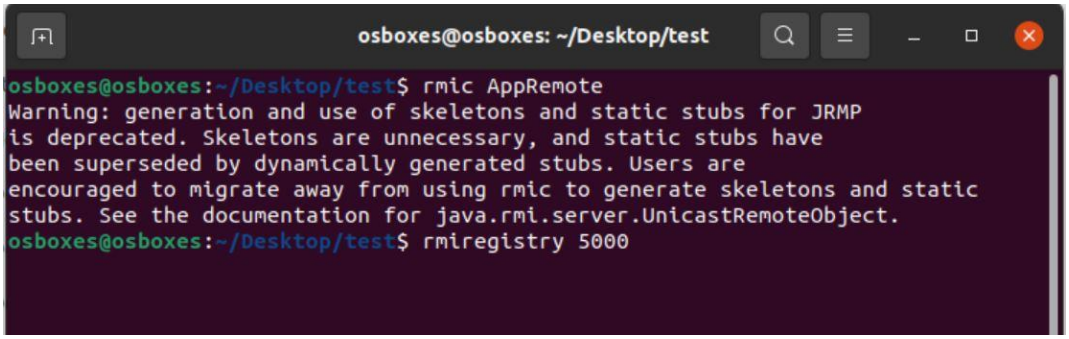


5. Install OpenJDK in ubuntu.

   sudo apt install default-jre (3 mins 10 sec)

   sudo apt install default-jdk (17 mins 40 secs)

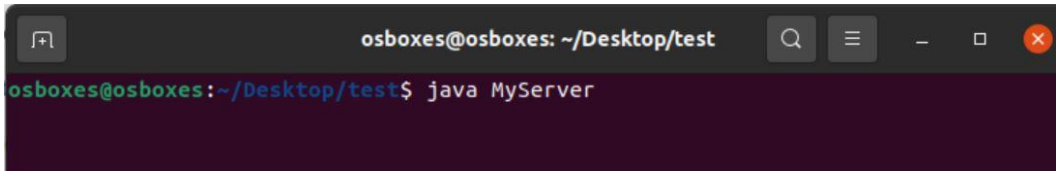
6. Copy the java and Class files to ubuntu OS.
7. Run rmic AppRemote



8. Run rmiregistry with the port number.
9. Run java MyServer

10. Run java MyClient

```
osboxes@osboxes:~/Desktop/test$ java MyClient
1.Armstrong
 2. Neon
Enter choice
1
Enter number
153
Armstrong Number
```

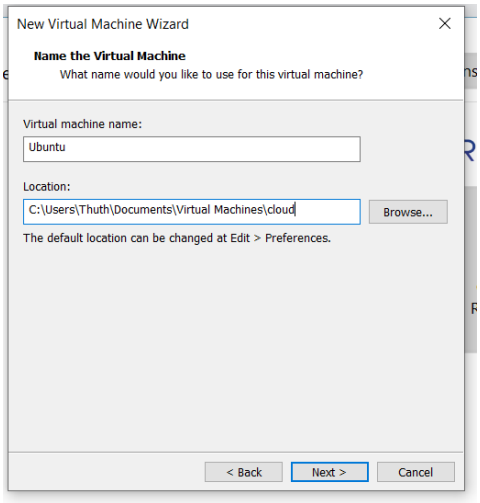


2. VMWare

1. Install VMWare in your machine.
2. Download ubuntu VMware image file from the website.
   https://www.osboxes.org/ubuntu/

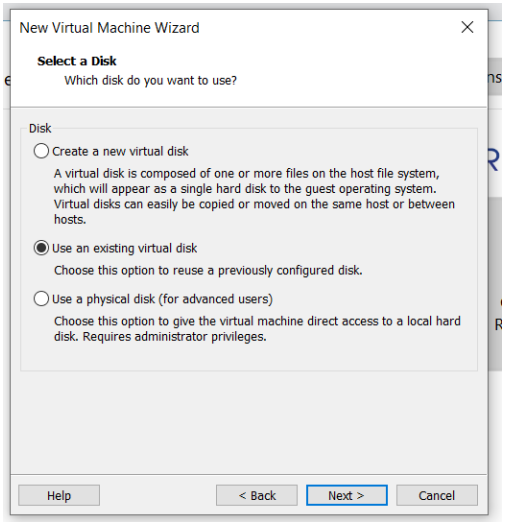
3. After installation choose “create new Virtual Machine”
4. Click custom, select hardware requirements, and then click next.
5. Choose “I will install OS later”.
6. Select the guest operating system that is installed, including the version. Click Next.

7. Provide a file name and choose the location where you want to save the virtual



machine. Click Next.

8. Click next and keep moving on. When prompted to select a disk choose “use an existing virtual disk”



9. Once done open the VM for usage
10. Install OpenJDK in ubuntu.

    sudo apt install default-jre (3 mins 10 sec)

    sudo apt install default-jdk (17 mins 40 secs)
11. Copy the java and Class files to ubuntu OS.
12. Run rmic AppRemote

13. Run rmi registry with the port number.

```
osboxes@osboxes:~/Desktop/test$ rmic AppRemote
Warning: generation and use of skeletons and static stubs for JRMP
is deprecated. Skeletons are unnecessary, and static stubs have
been superseded by dynamically generated stubs. Users are
encouraged to migrate away from using rmic to generate skeletons and static
stubs. See the documentation for java.rmi.server.UnicastRemoteObject.
osboxes@osboxes:~/Desktop/test$ rmiregistry 5000
```

14. Run java MyServer

15. Run java MyClient

```
osboxes@osboxes:~/Desktop/test$ java MyServer
```

```
osboxes@osboxes: ~/Desktop/test
osboxes@osboxes:~/Desktop/test$ java MyClient
1.Armstrong
 2. Neon
Enter choice
2
Enter number
83
Not an Neon Number
osboxes@osboxes:~/Desktop/test$
```

RMI in VM and Windows

1. Open Oracle VM Virtual Box
2. Set up the required files for server in windows.
3. Set up the necessary files for client in ubuntu.
4. Run the programs.

```
F:\Cloud Computing\Lab - Cloud\RMI>javac *.java

F:\Cloud Computing\Lab - Cloud\RMI>java MyServer
Registry Created and Initialized...
Server is ready...
```

Code:

App.java

```
import java.rmi.*;
public interface App extends Remote{
public String mathe(int x, int y)throws RemoteException;
}
```

AppRemote.java

```
import java.rmi.*;
import java.rmi.server.*;
import java.util.*;
import java.io.*;
public class AppRemote extends UnicastRemoteObject implements App{
AppRemote()throws RemoteException{
super();
}
public String mathe(int x, int y)
{
int temp,a;
int c = 0;
String ans1 = "Armstrong Number";
String ans2 = "Not an Armstrong Number";
```

```
String ans3 = "Neon Number";
String ans4 = "Not an Neon Number";
if(y==1)
{
 temp = x;
 while(x>0)
 {
 a=x%10;
 x=x/10;
 c=c+(a*a*a);
 }
 if(temp==c)
 return ans1;
 else
 return ans2;
}
else if(y==2)
{
 int square=x*x;
 int sum=0;
 while(square!=0)
 {
 int b=square%10;
 sum=sum+b;
 square=square/10;
 }
 if(sum==x)
 {
```

```
 return ans3;
 }
 else
 {
 return ans4;
 }
}
else if(y==3)
{
int i, last = 0 ;
Scanner sc = new Scanner(System.in);
System.out.println("\n Please Enter any Number: ");
int Number = sc.nextInt();
for(i = 1 ; i < x ; i++) {
if(x % i == 0) {
 last = last + i;
}
}
if (last == x) {
return "Perfect Number";
}
else {
return "Not a Perfect Number";
}
}
else
{
return "Sorry, Wrong Choice";
```

```
}
}
}
```

MyServer.java

```
import java.rmi.*;
import java.rmi.registry.*;
public class MyServer{
public static void main(String args[]){
try{
App stub=new AppRemote();
Naming.rebind("rmi://localhost:5000/sonoo",stub);
} catch(Exception e){System.out.println(e);}
}
}
```

MyClient.java

```
import java.rmi.*;
import java.io.*;
import java.util.*;
public class MyClient{
public static void main(String args[]){
try{
int a, b;
App stub=(App)Naming.lookup("rmi://localhost:5000/sonoo");
Scanner sc = new Scanner(System.in);
System.out.println("1.Armstrong \n 2. Neon");
System.out.println("Enter choice");
```

```
b =sc.nextInt();
System.out.println("Enter number");
a = sc.nextInt();
System.out.println(stub.mathe(a, b));
}catch(Exception e){}
}
}
```

RMI on two machines:

MyServer1.java

```
import java.net.MalformedURLException;
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
public class MyServer1 {
public static void main(String[] argv) throws RemoteException {
Sub Hello = new Sub();
System.setProperty("java.rmi.server.hostname","192.168.1.2");
int port = 1099;
try { // special exception handler for registry creation
LocateRegistry.createRegistry(port);
System.out.println("Registry Created and Initialized...");
} catch (RemoteException e) {
// do nothing, error means registry already exists
System.out.println("Java RMI registry already exists.");
}
String hostname = "192.168.1.2";
String bindLocation = "//" + hostname + ":" + port + "/Hello";
```

```
try {
Naming.bind(bindLocation, Hello);
System.out.println("Server is ready...");
} catch (RemoteException e) {
        // TODO Auto-generated catch block
e.printStackTrace();
} catch (MalformedURLException e) {
        // TODO Auto-generated catch block
e.printStackTrace();
} catch (java.lang.Exception e) {
System.out.println("Server failed: " + e);
}
}
}
```

MyClient1.java

```
import java.net.MalformedURLException;
import java.rmi.*;
import java.util.*;
public class MyClient {
public static void main(String[] args) {
Scanner ip = new Scanner(System.in);
String remoteHostName = "192.168.1.2";
int remotePort = 1099;
String connectLocation = "//" + remoteHostName + ":" + remotePort + "/Hello";
SubRemote hello = null;
try {
System.out.println("Connecting to client at : " + remoteHostName);
```

```
hello = (RemoteInterface) Naming.lookup(connectLocation);
} catch (MalformedURLException e1) {
        // TODO Auto-generated catch block
e1.printStackTrace();
} catch (RemoteException e1) {
        // TODO Auto-generated catch block
e1.printStackTrace();
} catch (NotBoundException e1) {
        // TODO Auto-generated catch block
e1.printStackTrace();
}
int result = 0;
System.out.println("Enter the first value: ");
int a = ip.nextInt();
System.out.println("Enter the second value: ");
int b = ip.nextInt();
try {
result = hello.Sub(a, b);
} catch (RemoteException e1) {
        // TODO Auto-generated catch block
e1.printStackTrace();}
System.out.println("The value is :" + result);
} }
```

Sub.java

```
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
public class Sub extends UnicastRemoteObject implements
```

```
RemoteInterface {
private static final long serialVersionUID = 1L;
public Sub() throws RemoteException {
    // TODO Auto-generated constructor stub
}
public int Sub(int a, int b) {
return a - b;
}
}
```

SubRemote.java

```
import java.rmi.Remote;
import java.rmi.RemoteException;
public interface RemoteInterface extends Remote
{
public int Sub(int a, int b)throws RemoteException;
}
```

Result:

The RMI using java in Ubuntu on VMWare and Oracle VM Virtual Box is done and their performance is compared.