PRACTICAL-2

AIM

Understanding of basic CloudSim Examples.

(1) Write a program in cloudsim using NetBeans IDE to create a datacenter with one host and run four cloudlet on it.

(2) Write a program in cloudsim using NetBeans IDE to create a datacenter with three hosts and run three cloudlets on it.

IMPLEMENTATION

Code for Datacenter with 1 host and 4 cloudlets:

```
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.LinkedList;
import java.util.List;

import org.cloudbus.cloudsim.Cloudlet;
import org.cloudbus.cloudsim.CloudletSchedulerTimeShared;
import org.cloudbus.cloudsim.Datacenter;
import org.cloudbus.cloudsim.DatacenterBroker;
import org.cloudbus.cloudsim.DatacenterCharacteristics;
import org.cloudbus.cloudsim.Host;
import org.cloudbus.cloudsim.Log;
import org.cloudbus.cloudsim.Pe;
import org.cloudbus.cloudsim.Storage;
import org.cloudbus.cloudsim.UtilizationModel;
import org.cloudbus.cloudsim.UtilizationModelFull;
import org.cloudbus.cloudsim.Vm;
import org.cloudbus.cloudsim.VmAllocationPolicySimple;
import org.cloudbus.cloudsim.VmSchedulerTimeShared;
import org.cloudbus.cloudsim.core.CloudSim;
import org.cloudbus.cloudsim.provisioners.BwProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.PeProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.RamProvisionerSimple;

public class Prac2_A {

   /**
    * @param args the command line arguments
    */
      private static List<Cloudlet> cloudletList;

   private static List<Vm> vmlist;
```

```
public static void main(String[] args)
{
   Log.printLine("Starting Cloudsim ...");

   try
   {
      int num_user = 1;
      Calendar calendar = Calendar.getInstance();
      boolean trace_flag = false;

      CloudSim.init(num_user, calendar, trace_flag);

      Datacenter datacenter0 = createDatacenter("Datacenter_0");

      DatacenterBroker broker = createBroker("Broker");
      int brokerId = broker.getId();

      vmlist = new ArrayList<Vm>();

      int vmid = 0;
      int mips = 1000;
      long size = 10000;
      int ram = 512;
      long bw = 1000;
      int pesNumber = 1;
      String vmm = "Xen";

      Vm vm = new Vm(vmid, brokerId, mips, pesNumber, ram, bw, size, vmm, new CloudletSchedulerTimeShared());

      vmlist.add(vm);

      broker.submitVmList(vmlist);

      cloudletList = new ArrayList<Cloudlet>();

      int id = 0;
      long length = 400000;
      long fileSize = 300;
      long outputSize = 300;
      UtilizationModel utilizationModel = new UtilizationModelFull();

      Cloudlet cloudlet0 = new Cloudlet(id++, length, pesNumber, fileSize, outputSize, utilizationModel, utilizationModel, utilizationModel);
      cloudlet0.setUserId(brokerId);
      cloudlet0.setVmId(vmid);
      cloudletList.add(cloudlet0);
```

```
        Cloudlet cloudlet1 = new Cloudlet(id++, length, pesNumber, fileSize, outputSize, utilizationModel, utilizationModel, utilizationModel);
        cloudlet1.setUserId(brokerId);
        cloudlet1.setVmId(vmid);
        cloudletList.add(cloudlet1);

        Cloudlet cloudlet2 = new Cloudlet(id++, length, pesNumber, fileSize, outputSize, utilizationModel, utilizationModel, utilizationModel);
        cloudlet2.setUserId(brokerId);
        cloudlet2.setVmId(vmid);
        cloudletList.add(cloudlet2);

        Cloudlet cloudlet3 = new Cloudlet(id++, length, pesNumber, fileSize, outputSize, utilizationModel, utilizationModel, utilizationModel);
        cloudlet3.setUserId(brokerId);
        cloudlet3.setVmId(vmid);
        cloudletList.add(cloudlet3);

        broker.submitCloudletList(cloudletList);

        CloudSim.startSimulation();

        CloudSim.stopSimulation();

        List<Cloudlet> newList = broker.getCloudletReceivedList();
        printCloudletList(newList);

        Log.printLine("CloudSim example finished!");
      }
      catch(Exception ex)
      {
        ex.printStackTrace();
        Log.printLine("Unwanted error occured");
      }
   }

   public static Datacenter createDatacenter(String name)
   {
      List<Host> hostList = new ArrayList<Host>();

      List<Pe> peList = new ArrayList<Pe>();

      int mips = 1000;

      peList.add(new Pe(0, new PeProvisionerSimple(mips)));

      int hostId = 0;
      int ram = 2048;
      long storage = 1000000;
      int bw = 10000;
```

```
        hostList.add(new Host(hostId, new RamProvisionerSimple(ram), new
BwProvisionerSimple(bw), storage, peList, new VmSchedulerTimeShared(peList)));

        String arch = "x86";
        String os = "Linux";
        String vmm = "Xen";
        double time_zone = 10.0;
        double cost = 3.0;
        double costPerMem = 0.05;
        double costPerStorage = 0.001;
        double costPerBw = 0.0;

        LinkedList<Storage> storageList = new LinkedList<Storage>();

        DatacenterCharacteristics characteristics = new DatacenterCharacteristics(arch, os, vmm,
hostList, time_zone, cost, costPerMem, costPerStorage, costPerBw);

        Datacenter datacenter = null;
        try
        {
            datacenter = new Datacenter(name, characteristics, new
VmAllocationPolicySimple(hostList), storageList, 0);
        }
        catch(Exception ex) {
            ex.printStackTrace();
        }

        return datacenter;
    }

    private static DatacenterBroker createBroker(String name) {
        DatacenterBroker broker = null;
        try
        {
            broker = new DatacenterBroker(name);
        }
        catch(Exception ex)
        {
            ex.printStackTrace();
        }
        return broker;
    }

    private static void printCloudletList(List<Cloudlet> list) {
                int size = list.size();
                Cloudlet cloudlet;

                String indent = "    ";
                Log.printLine();
```

```
                Log.printLine("========== OUTPUT ==========");
                Log.printLine("Cloudlet ID" + indent + "STATUS" + indent
                                + "Data center ID" + indent + "VM ID" + indent + "Time" +
indent
                                + "Start Time" + indent + "Finish Time");

                DecimalFormat dft = new DecimalFormat("###.##");
                for (int i = 0; i < size; i++) {
                        cloudlet = list.get(i);
                        Log.print(indent + cloudlet.getCloudletId() + indent + indent);

                        if (cloudlet.getCloudletStatus() == Cloudlet.SUCCESS) {
                                Log.print("SUCCESS");

                                Log.printLine(indent + indent + cloudlet.getResourceId()
                                                + indent + indent + indent + cloudlet.getVmId()
                                                + indent + indent
                                                +   dft.format(cloudlet.getActualCPUTime())   +
indent
                                                +                       indent                  +
dft.format(cloudlet.getExecStartTime())
                                                + indent + indent
                                                + dft.format(cloudlet.getFinishTime()));
                        }               }       }   }
```

Code for 1 datacenter with 3 hosts and 3 cloudlets:

```
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.LinkedList;
import java.util.List;

import org.cloudbus.cloudsim.Cloudlet;
import org.cloudbus.cloudsim.CloudletSchedulerTimeShared;
import org.cloudbus.cloudsim.Datacenter;
import org.cloudbus.cloudsim.DatacenterBroker;
import org.cloudbus.cloudsim.DatacenterCharacteristics;
import org.cloudbus.cloudsim.Host;
import org.cloudbus.cloudsim.Log;
import org.cloudbus.cloudsim.Pe;
import org.cloudbus.cloudsim.Storage;
import org.cloudbus.cloudsim.UtilizationModel;
import org.cloudbus.cloudsim.UtilizationModelFull;
import org.cloudbus.cloudsim.Vm;
import org.cloudbus.cloudsim.VmAllocationPolicySimple;
import org.cloudbus.cloudsim.VmSchedulerTimeShared;
import org.cloudbus.cloudsim.core.CloudSim;
import org.cloudbus.cloudsim.provisioners.BwProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.PeProvisionerSimple;
```

```
import org.cloudbus.cloudsim.provisioners.RamProvisionerSimple;

public class Prac2_B {

    /**
     * @param args the command line arguments
     */

    private static List<Cloudlet> cloudletList;

    private static List<Vm> vmlist;

    public static void main(String[] args) {
        // TODO code application logic here
        Log.printLine("Staring CloudSim ...");

        try
        {
            int num_user = 1;
            Calendar calendar = Calendar.getInstance();
            boolean trace_flag = false;

            CloudSim.init(num_user, calendar, trace_flag);

            Datacenter datacenter0 = createDatacenter("Datacenter_0");

            DatacenterBroker broker = createBroker("broker1");
            int brokerId = broker.getId();

            vmlist = new ArrayList<Vm>();

            int vmid = 0;
            int mips = 250;
            long size = 10000;
            int ram = 512;
            //int ram = 1024;
            long bw = 1000;
            int pesNumber = 1;
            String vmm = "Xen";

            //Vm vm = new Vm(vmid, brokerId, mips, pesNumber, ram, bw, size, vmm, new
CloudletSchedulerTimeShared());
            Vm vm0 = new Vm(vmid++, brokerId, mips, pesNumber, ram, bw, size, vmm, new
CloudletSchedulerTimeShared());
            Vm vm1 = new Vm(vmid++, brokerId, mips, pesNumber, ram, bw, size, vmm, new
CloudletSchedulerTimeShared());
            Vm vm2 = new Vm(vmid++, brokerId, mips, pesNumber, ram, bw, size, vmm, new
CloudletSchedulerTimeShared());

            //vmlist.add(vm);
```

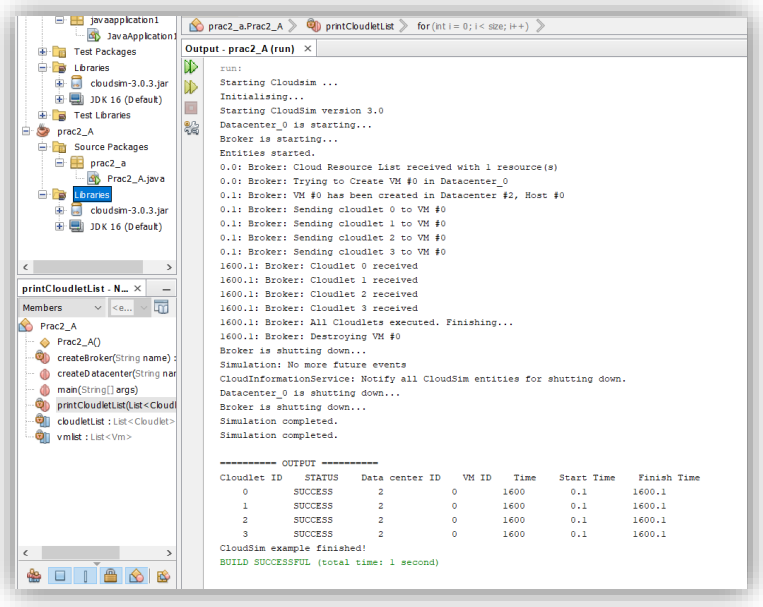
```
        vmlist.add(vm0);
        vmlist.add(vm1);
        vmlist.add(vm2);
        broker.submitVmList(vmlist);

        cloudletList = new ArrayList<Cloudlet>();

        int id = 0;
        long length = 400000;
        long fileSize = 300;
        long outputSize = 300;
        UtilizationModel utilizationModel = new UtilizationModelFull();

        Cloudlet cloudlet0 = new Cloudlet(id++, length, pesNumber, fileSize, outputSize, utilizationModel, utilizationModel, utilizationModel);
        cloudlet0.setUserId(brokerId);
        cloudlet0.setVmId(0);
        cloudletList.add(cloudlet0);

        Cloudlet cloudlet1 = new Cloudlet(id++, length, pesNumber, fileSize, outputSize, utilizationModel, utilizationModel, utilizationModel);
        cloudlet1.setUserId(brokerId);
        cloudlet1.setVmId(1);
        cloudletList.add(cloudlet1);

        Cloudlet cloudlet2 = new Cloudlet(id++, length, pesNumber, fileSize, outputSize, utilizationModel, utilizationModel, utilizationModel);
        cloudlet2.setUserId(brokerId);
        cloudlet2.setVmId(2);
        cloudletList.add(cloudlet2);

        broker.submitCloudletList(cloudletList);

        CloudSim.startSimulation();

        CloudSim.stopSimulation();

        List<Cloudlet> newList = broker.getCloudletReceivedList();
        printCloudletList(newList);

        Log.printLine("Cloudsim finished");
      }
      catch(Exception ex)
      {
        ex.printStackTrace();
        Log.printLine("Unwanted error occured");
      }
   }

   public static Datacenter createDatacenter(String name)
```

```
   {
      List<Host> hostList = new ArrayList<Host>();

      List<Pe> peList0 = new ArrayList<Pe>();
      List<Pe> peList1 = new ArrayList<Pe>();
      List<Pe> peList2 = new ArrayList<Pe>();

      int mips = 1000;

      peList0.add(new Pe(0, new PeProvisionerSimple(mips)));
      peList1.add(new Pe(0, new PeProvisionerSimple(mips)));
      peList2.add(new Pe(0, new PeProvisionerSimple(mips)));

      int hostId = 0;
      int ram = 2048;
      long storage = 1000000;
      int bw = 10000;

      hostList.add(new Host(hostId++, new RamProvisionerSimple(ram), new
BwProvisionerSimple(bw), storage, peList0, new VmSchedulerTimeShared(peList0)));
      hostList.add(new Host(hostId++, new RamProvisionerSimple(ram), new
BwProvisionerSimple(bw), storage, peList1, new VmSchedulerTimeShared(peList1)));
      hostList.add(new Host(hostId++, new RamProvisionerSimple(ram), new
BwProvisionerSimple(bw), storage, peList2, new VmSchedulerTimeShared(peList2)));

      String arch = "x86";
      String os = "Linux";
      String vmm = "Xen";
      double time_zone = 10.0;
      double cost = 3.0;
      double costPerMem = 0.05;
      double costPerStorage = 0.001;
      double costPerBw = 0.0;
      LinkedList<Storage> storageList = new LinkedList<Storage>();
      DatacenterCharacteristics characteristics = new DatacenterCharacteristics(arch, os, vmm,
hostList, time_zone, cost, costPerMem, costPerStorage, costPerBw);

      Datacenter datacenter = null;
      try
      {
         datacenter = new Datacenter(name, characteristics, new
VmAllocationPolicySimple(hostList), storageList, 0);
      }
      catch(Exception ex)
      {
         ex.printStackTrace();
         return null;
      }
      return datacenter;
   }
```

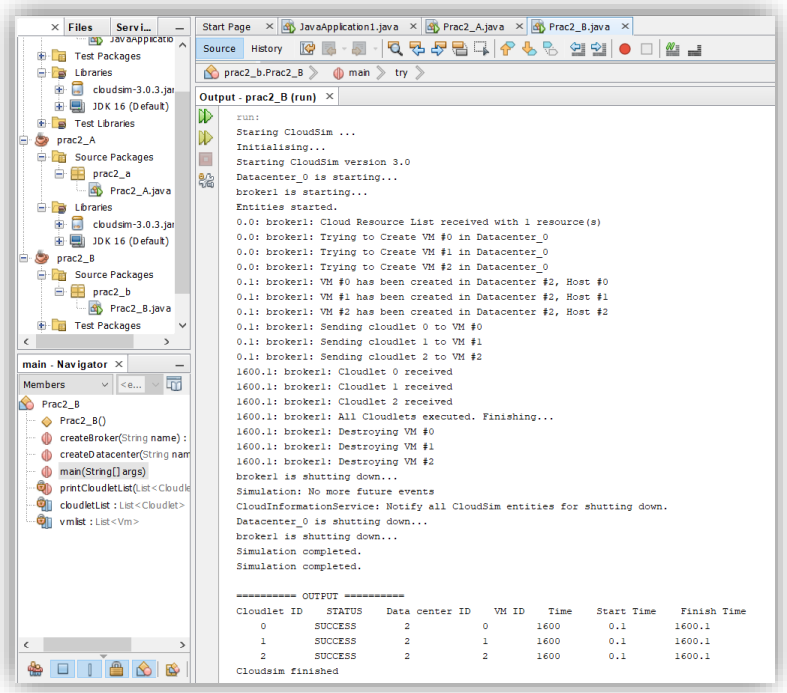
```
   public static DatacenterBroker createBroker(String name)
   {
      DatacenterBroker broker = null;
      try
      {
         broker = new DatacenterBroker(name);
      }
      catch(Exception ex)
      {
         ex.printStackTrace();
         return null;
      }
      return broker;
   }
   private static void printCloudletList(List<Cloudlet> list) {
                  int size = list.size();
                  Cloudlet cloudlet;

                  String indent = "    ";
                  Log.printLine();
                  Log.printLine("========== OUTPUT ==========");
                  Log.printLine("Cloudlet ID" + indent + "STATUS" + indent
                                 + "Data center ID" + indent + "VM ID" + indent + "Time" +
indent
                                 + "Start Time" + indent + "Finish Time");

                  DecimalFormat dft = new DecimalFormat("###.##");
                  for (int i = 0; i < size; i++) {
                        cloudlet = list.get(i);
                        Log.print(indent + cloudlet.getCloudletId() + indent + indent);

                        if (cloudlet.getCloudletStatus() == Cloudlet.SUCCESS) {
                              Log.print("SUCCESS");

                              Log.printLine(indent + indent + cloudlet.getResourceId()
                                          + indent + indent + indent + cloudlet.getVmId()
                                          + indent + indent
                                          +   dft.format(cloudlet.getActualCPUTime())   +
indent
                                          +                    indent                    +
dft.format(cloudlet.getExecStartTime())
                                          + indent + indent
                                          + dft.format(cloudlet.getFinishTime()));
                        }              }        }}
```

OUTPUT

For first:



For Second:



CONCLUSION

In this practical, we learnt about cloudsim architecture and implemented several different scenarios using different number of datacenters, hosts and cloudlets