Cloud Computing (ITITE03) Practical File

Submitted by:-

Shaurya Singh

2019UEC2679

Index

S. No.	Topic	Signature
1.	Program to study and Install Cloudsim simulator and write the steps of installation.	
2.	Program to create a datacenter with one host and run one cloudlet on it.	
3.	Program to create two datacenters with one host each and run cloudlets of two users with network topology on them.	
4.	Program to create two datacenters with one host each and run two cloudlets on them.	
5.	Program to create two datacenters with one host each and run cloudlets of two users on them.	
6.	Program to create a datacenter with two hosts and run two cloudlets on it.	
7.	Program to create scalable simulations.	

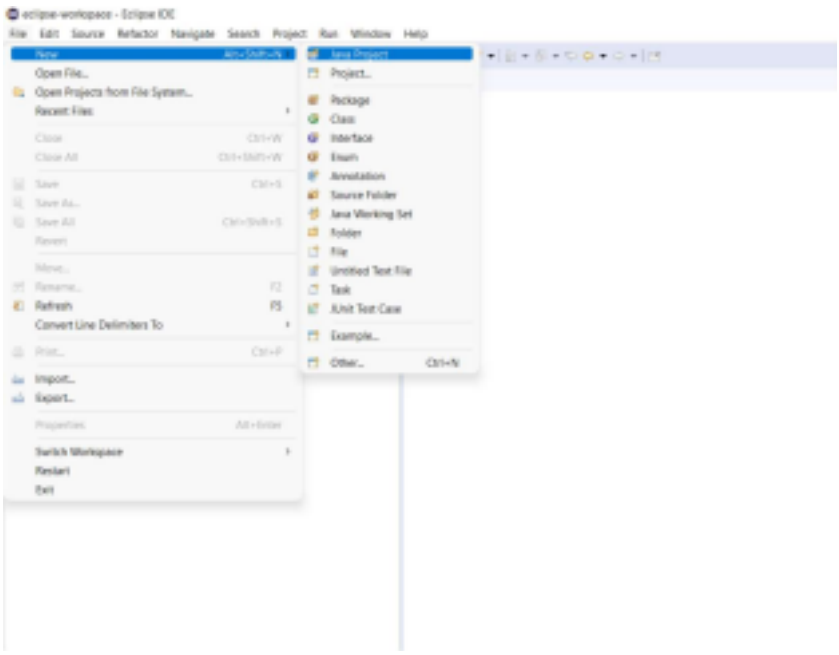
8.	Program to pause and resume the simulation, and create simulation entities (a DatacenterBroker in this example) dynamically.	
9.	Program to create simulation entities (a DatacenterBroker in this example) in run-time using a global manager entity (GlobalBroker).	
10.	Execute a Python Program for the following in Google-Colab Platform: a. Multiplication of 3 matrices of size 2 X 2. b. Analyse boston house prices dataset using Matplotlib/Seaborn Library.	

Experiment – 1

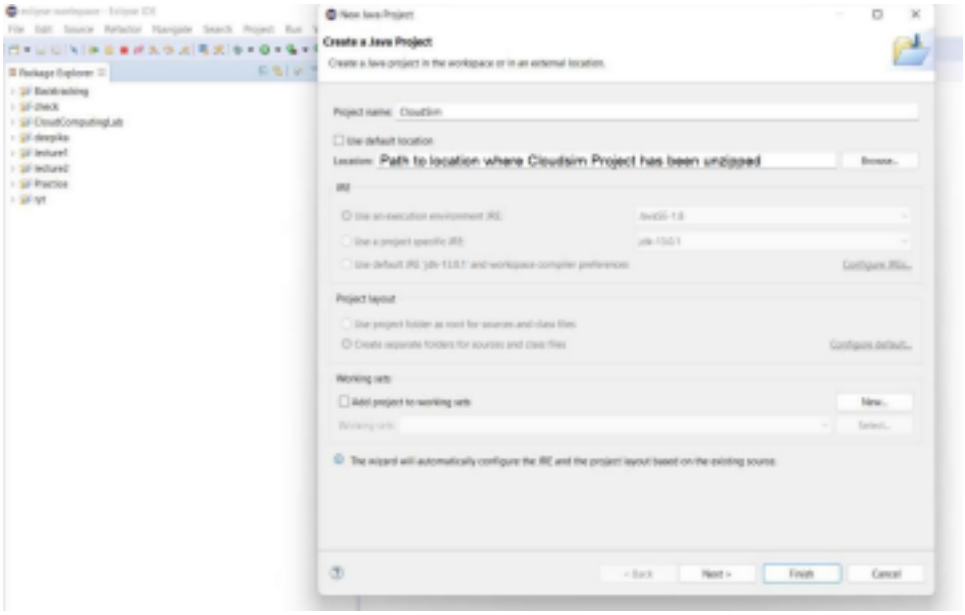
Aim - Program to study and Install Cloudsim simulator and write the steps of installation.

Procedure –

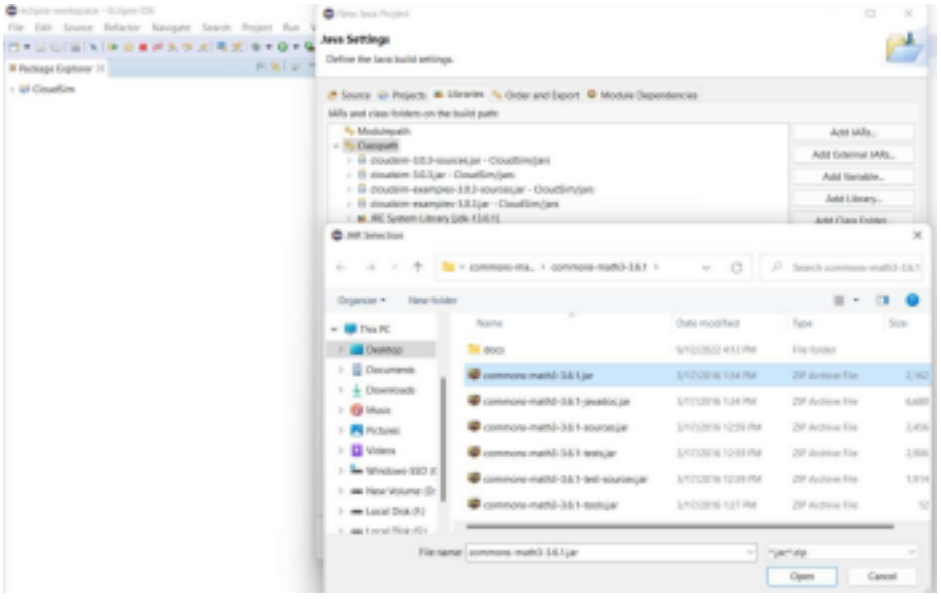
Step 1: Before installing CloudSim following resources must be downloaded on the local system: a) Eclipse IDE for JAVA Developer. b) CloudSim Project code: CloudSim code can be downloaded from following project page https://code.google.com/p/cloudsim/, always download latest version to avoid any bug issues of previous packages. c) One external requirement of CloudSim i.e., common jar package of math related functions is to be downloaded. Step 2: Unzip Eclipse and Cloudsim to some common folder.
Step 3: Open Eclipse.exe from Eclipse folder.
Step 4: Now in Eclipse go to menu File-> New -> Java Project.



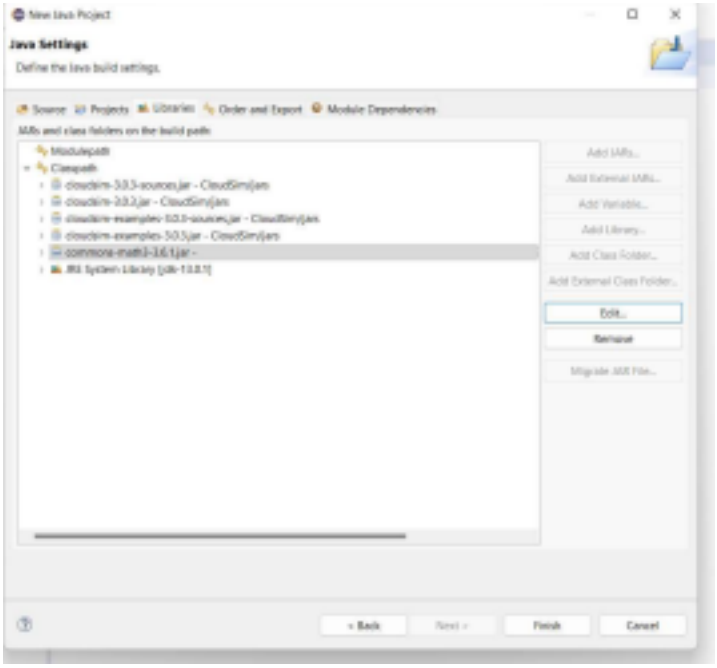
Step 5:. Now a detailed new project window will be opened, here you provide project name and the path where you have unzipped the CloudSim project source code or you can put details as specified: a) Project Name: CloudSim b) Unselect the 'Use default location' option and then click on browse to open the path where
you have unzipped the CloudSim project and finally click Next to set project settings.



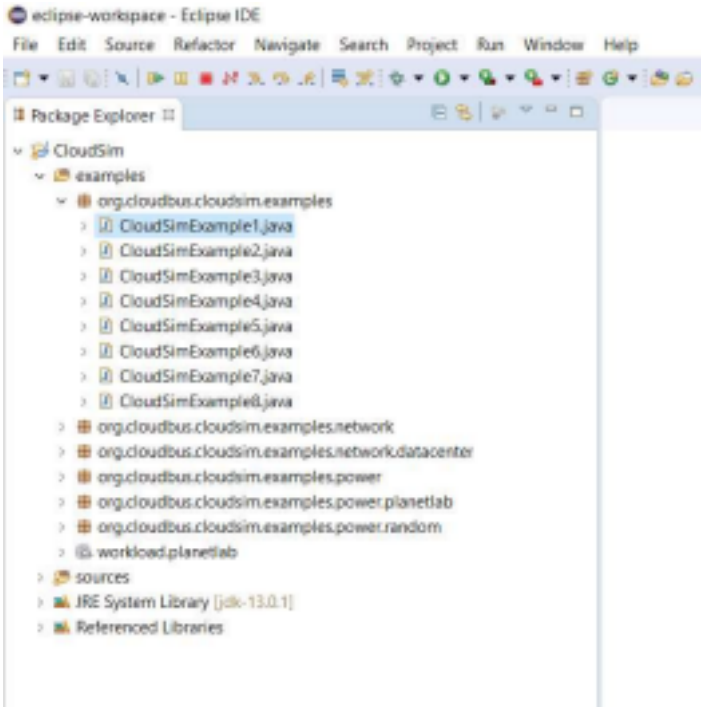
c) Now open ‘Libraries’ tab and click on add external jar (Math common jar will be included in project from this step). d) Open the path where you have unzipped the math common binaries and select ‘Common math 3.3.2.jar’ and click on open.



e) Ensure external jar that you opened in previous step is displayed in list and then click on ‘Finish’.



Step 6: Once the project is configured you can open the project explorer and start exploring the CloudSim project. In this manual we will be discussing all the examples that are provided under package ‘org.cloudbus.cloudsim.examples’ (as shown in figure below).



Experiment – 2

Aim - Program to create a datacenter with one host and run one cloudlet on it. Software Used – Eclipse

Package Used – Cloudsim(Framework)

Code –

```
package org.cloudbus.cloudsim.examples;
/*
 * Title: CloudSim Toolkit
 * Description: CloudSim (Cloud Simulation) Toolkit for Modeling and Simulation  * of Clouds
 * Licence: GPL - http://www.gnu.org/copyleft/gpl.html
 *
 * Copyright (c) 2009, The University of Melbourne, Australia  */
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;  /**
 * A simple example showing how to create a datacenter with one host and run one  * cloudlet on it.
 */
public class CloudSimExample1 {
 /** The cloudlet list. */
 private static List<Cloudlet> cloudletList;
 /** The vmlist. */
 private static List<Vm> vmlist;
```

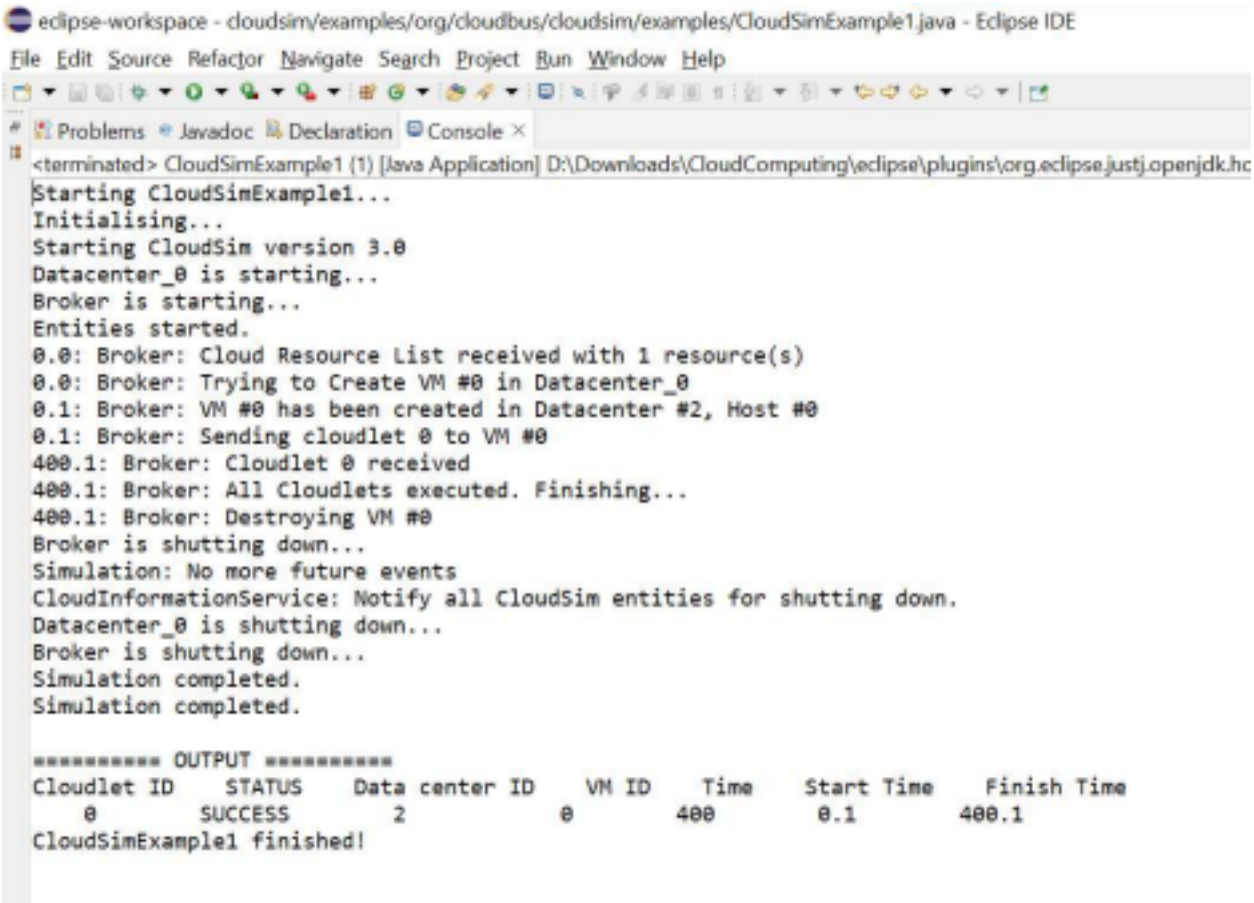
```
 /**
 * Creates main() to run this example.
 *
 * @param args the args
 */
 @SuppressWarnings("unused")
 public static void main(String[] args) {
 Log.printLine("Starting CloudSimExample1...");
 try {
 // First step: Initialize the CloudSim package. It should be called  // before creating any entities.
 int num_user = 1; // number of cloud users
 Calendar calendar = Calendar.getInstance();
 boolean trace_flag = false; // mean trace events  // Initialize the CloudSim
library
 CloudSim.init(num_user, calendar, trace_flag);
 // Second step: Create Datacenters
 // Datacenters are the resource providers in CloudSim. We need at  // list one of them to run a
CloudSim simulation  Datacenter datacenter0 = createDatacenter("Datacenter_0");   // Third
step: Create Broker
 DatacenterBroker broker = createBroker();
 int brokerId = broker.getId();
 // Fourth step: Create one virtual machine
 vmlist = new ArrayList<Vm>();
 // VM description
 int vmid = 0;
 int mips = 1000;
 long size = 10000; // image size (MB)
 int ram = 512; // vm memory (MB)
 long bw = 1000;
 int pesNumber = 1; // number of cpus
 String vmm = "Xen"; // VMM name
 // create VM
 Vm vm = new Vm(vmid, brokerId, mips, pesNumber, ram, bw, size, vmm,  new
CloudletSchedulerTimeShared());
 // add the VM to the vmList
 vmlist.add(vm);
 // submit vm list to the broker
 broker.submitVmList(vmlist);
 // Fifth step: Create one Cloudlet
 cloudletList = new ArrayList<Cloudlet>();
 // Cloudlet properties
 int id = 0;
 long length = 400000;
 long fileSize = 300;
 long outputSize = 300;
 UtilizationModel utilizationModel = new UtilizationModelFull();   Cloudlet cloudlet = new
Cloudlet(id, length, pesNumber, fileSize,  outputSize, utilizationModel, utilizationModel,
utilizationModel);   cloudlet.setUserId(brokerId);
 cloudlet.setVmId(vmid);
```

```
 // add the cloudlet to the list
 cloudletList.add(cloudlet);
 // submit cloudlet list to the broker
 broker.submitCloudletList(cloudletList);
 // Sixth step: Starts the simulation
 CloudSim.startSimulation();
 CloudSim.stopSimulation();
 //Final step: Print results when simulation is over  List<Cloudlet> newList =
broker.getCloudletReceivedList();   printCloudletList(newList);
 Log.printLine("CloudSimExample1 finished!");
 } catch (Exception e) {
 e.printStackTrace();
 Log.printLine("Unwanted errors happen");
 }
 }
 /**
 * Creates the datacenter.
 *
 * @param name the name
 *
 * @return the datacenter
 */
 private static Datacenter createDatacenter(String name) {   // Here are the steps
needed to create a PowerDatacenter:  // 1. We need to create a list to store
 // our machine
 List<Host> hostList = new ArrayList<Host>();
 // 2. A Machine contains one or more PEs or CPUs/Cores.  // In this example,
it will have only one core.
 List<Pe> peList = new ArrayList<Pe>();
 int mips = 1000;
 // 3. Create PEs and add these into a list.
 peList.add(new Pe(0, new PeProvisionerSimple(mips))); // need to store Pe  id and MIPS Rating
 // 4. Create Host with its id and list of PEs and add them to the list  // of machines
 int hostId = 0;
 int ram = 2048; // host memory (MB)
 long storage = 1000000; // host storage
 int bw = 10000;
 hostList.add(
 new Host(
 hostId,
 new RamProvisionerSimple(ram),
 new BwProvisionerSimple(bw),
 storage,
 peList,
 new VmSchedulerTimeShared(peList)
 )
 ); // This is our machine
 // 5. Create a DatacenterCharacteristics object that stores the  // properties of a data
```

```
center: architecture, OS, list of  // Machines, allocation policy: time- or space-shared, time
zone  // and its price (G$/Pe time unit).
 String arch = "x86"; // system architecture
 String os = "Linux"; // operating system
 String vmm = "Xen";
 double time_zone = 10.0; // time zone this resource located  double cost = 3.0; // the cost of
using processing in this resource  double costPerMem = 0.05; // the cost of using memory in this
resource  double costPerStorage = 0.001; // the cost of using storage in this  // resource
 double costPerBw = 0.0; // the cost of using bw in this resource  LinkedList<Storage> storageList
= new LinkedList<Storage>(); // we are  not adding SAN
 // devices by now  DatacenterCharacteristics characteristics = new
DatacenterCharacteristics(
 arch, os, vmm, hostList, time_zone, cost, costPerMem,   costPerStorage, costPerBw);
 // 6. Finally, we need to create a PowerDatacenter object.
 Datacenter datacenter = null;
 try {
 datacenter = new Datacenter(name, characteristics, new
VmAllocationPolicySimple(hostList), storageList, 0);
 } catch (Exception e) {
 e.printStackTrace();
 }
 return datacenter;
 }
 // We strongly encourage users to develop their own broker policies, to  // submit vms and
cloudlets according
 // to the specific rules of the simulated scenario
 /**
 * Creates the broker.
 *
 * @return the datacenter broker
 */
 private static DatacenterBroker createBroker() {
 DatacenterBroker broker = null;
 try {
 broker = new DatacenterBroker("Broker");
 } catch (Exception e) {
 e.printStackTrace();
 return null;
 }
 return broker;
 }
 /**
 * Prints the Cloudlet objects.
 *
 * @param list list of Cloudlets
 */
 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 –



Experiment – 3

Aim - Program to create two datacenters with one host each and run cloudlets of two users with network topology on them.

Software Used – Eclipse

Package Used – Cloudsim(Framework)

Code –

```
* Title: CloudSim Toolkit
 * Description: CloudSim (Cloud Simulation) Toolkit for Modeling and Simulation  * of Clouds
 * Licence: GPL - http://www.gnu.org/copyleft/gpl.html
 *
 * Copyright (c) 2009, The University of Melbourne, Australia  */
package org.cloudbus.cloudsim.examples.network;
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.NetworkTopology;
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.VmSchedulerSpaceShared;
import org.cloudbus.cloudsim.core.CloudSim;
import     org.cloudbus.cloudsim.provisioners.BwProvisionerSimple;         import
org.cloudbus.cloudsim.provisioners.PeProvisionerSimple;                   import
org.cloudbus.cloudsim.provisioners.RamProvisionerSimple;
/**
 * A simple example showing how to create
 * two datacenters with one host each and
 * run cloudlets of two users with network
 * topology on them.
```

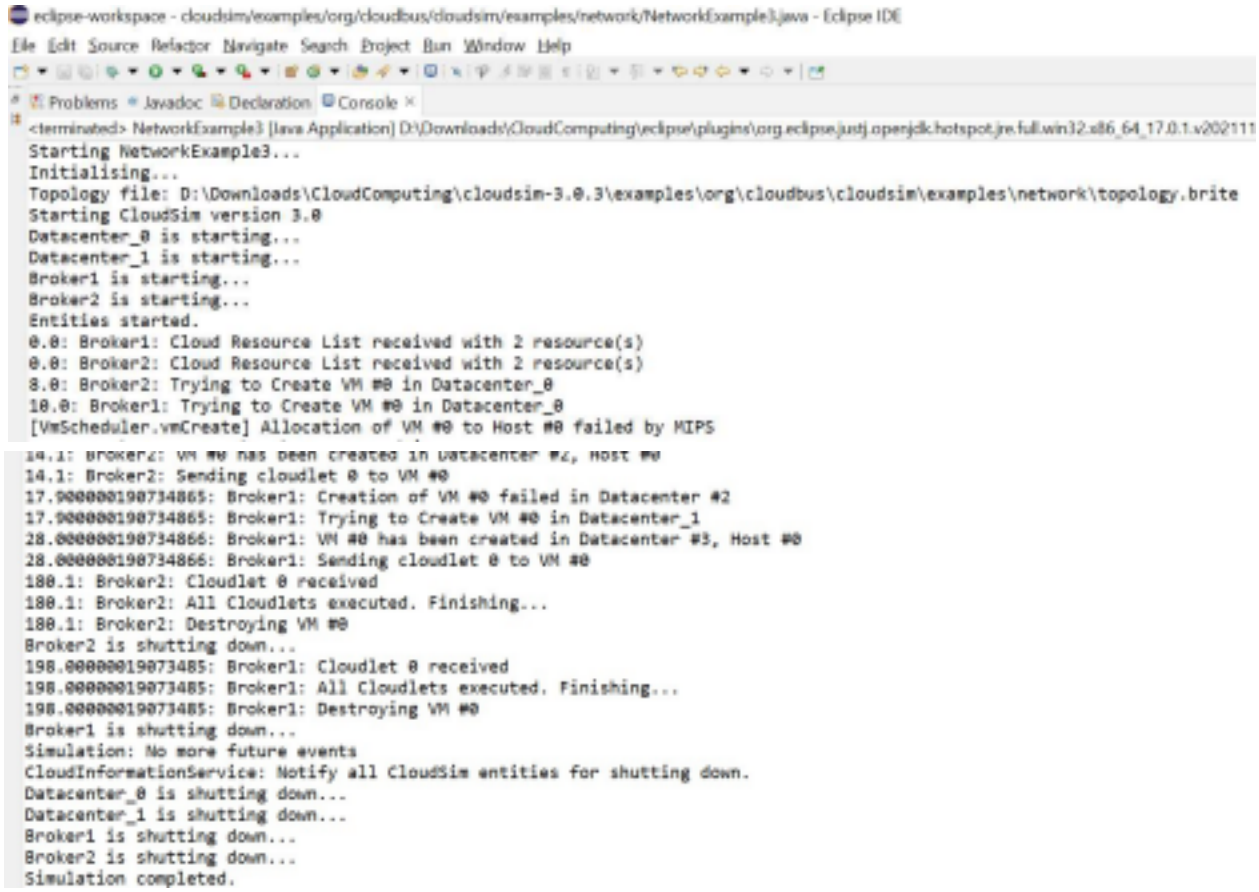
```
 */
public class NetworkExample3 {
 /** The cloudlet list. */
 private static List<Cloudlet> cloudletList1;
 private static List<Cloudlet> cloudletList2;
 /** The vmlist. */
 private static List<Vm> vmlist1;
 private static List<Vm> vmlist2;
 /**
 * Creates main() to run this example
 */
 public static void main(String[] args) {
 Log.printLine("Starting NetworkExample3...");
 try {
 // First step: Initialize the CloudSim package. It should be called  // before creating any entities.
 int num_user = 2; // number of cloud users
 Calendar calendar = Calendar.getInstance();
 boolean trace_flag = false; // mean trace events
 // Initialize the CloudSim library
 CloudSim.init(num_user, calendar, trace_flag);
 // Second step: Create Datacenters
 //Datacenters are the resource providers in CloudSim. We need at list  one of them to run a
CloudSim simulation
 Datacenter datacenter0 = createDatacenter("Datacenter_0");    Datacenter datacenter1
= createDatacenter("Datacenter_1");   //Third step: Create Brokers
 DatacenterBroker broker1 = createBroker(1);
 int brokerId1 = broker1.getId();
 DatacenterBroker broker2 = createBroker(2);
 int brokerId2 = broker2.getId();
 //Fourth step: Create one virtual machine for each broker/user  vmlist1 = new
ArrayList<Vm>();
 vmlist2 = new ArrayList<Vm>();
 //VM description
 int vmid = 0;
 long size = 10000; //image size (MB)
 int mips = 250;
 int ram = 512; //vm memory (MB)
 long bw = 1000;
 int pesNumber = 1; //number of cpus
 String vmm = "Xen"; //VMM name
 //create two VMs: the first one belongs to user1
 Vm vm1 = new Vm(vmid, brokerId1, mips, pesNumber, ram, bw, size, vmm,  new
CloudletSchedulerTimeShared());
 //the second VM: this one belongs to user2
 Vm vm2 = new Vm(vmid, brokerId2, mips, pesNumber, ram, bw, size, vmm,  new
CloudletSchedulerTimeShared());
 //add the VMs to the vmlists
 vmlist1.add(vm1);
 vmlist2.add(vm2);
```

```
 //submit vm list to the broker
 broker1.submitVmList(vmlist1);
 broker2.submitVmList(vmlist2);
 //Fifth step: Create two Cloudlets
 cloudletList1 = new ArrayList<Cloudlet>();
 cloudletList2 = new ArrayList<Cloudlet>();
 //Cloudlet properties
 int id = 0;
 long length = 40000;
 long fileSize = 300;
 long outputSize = 300;
 UtilizationModel utilizationModel = new UtilizationModelFull();   Cloudlet cloudlet1 = new
Cloudlet(id, length, pesNumber, fileSize,  outputSize, utilizationModel, utilizationModel,
utilizationModel);   cloudlet1.setUserId(brokerId1);
 Cloudlet cloudlet2 = new Cloudlet(id, length, pesNumber, fileSize,  outputSize, utilizationModel,
utilizationModel, utilizationModel);   cloudlet2.setUserId(brokerId2);
 //add the cloudlets to the lists: each cloudlet belongs to one user  cloudletList1.add(cloudlet1);
 cloudletList2.add(cloudlet2);
 //submit cloudlet list to the brokers
 broker1.submitCloudletList(cloudletList1);
 broker2.submitCloudletList(cloudletList2);
 //Sixth step: configure network
 //load the network topology file
 String path = "D:\\Downloads\\CloudComputing\\cloudsim
3.0.3\\examples\\org\\cloudbus\\cloudsim\\examples\\network\\";
NetworkTopology.buildNetworkTopology(path + "topology.brite");   //maps CloudSim entities
to BRITE entities
 //Datacenter0 will correspond to BRITE node 0
 int briteNode=0;
 NetworkTopology.mapNode(datacenter0.getId(),briteNode);   //Datacenter1 will
correspond to BRITE node 2
 briteNode=2;
 NetworkTopology.mapNode(datacenter1.getId(),briteNode);   //Broker1 will
correspond to BRITE node 3
 briteNode=3;
 NetworkTopology.mapNode(broker1.getId(),briteNode);   //Broker2 will
correspond to BRITE node 4
 briteNode=4;
 NetworkTopology.mapNode(broker2.getId(),briteNode);   // Sixth step: Starts
the simulation
 CloudSim.startSimulation();
 // Final step: Print results when simulation is over  List<Cloudlet> newList1 =
broker1.getCloudletReceivedList();   List<Cloudlet> newList2 =
broker2.getCloudletReceivedList();   CloudSim.stopSimulation();
 Log.print("=============> User "+brokerId1+" ");
printCloudletList(newList1);
 Log.print("=============> User "+brokerId2+" ");
printCloudletList(newList2);
 Log.printLine("NetworkExample3 finished!");
```

```
 }
 catch (Exception e) {
 e.printStackTrace();
 Log.printLine("The simulation has been terminated due to an  unexpected error");
 }
 }
 private static Datacenter createDatacenter(String name){
 // Here are the steps needed to create a PowerDatacenter:  // 1. We need to
create a list to store
 // our machine
 List<Host> hostList = new ArrayList<Host>();
 // 2. A Machine contains one or more PEs or CPUs/Cores.  // In this example,
it will have only one core.
 List<Pe> peList = new ArrayList<Pe>();
 int mips = 1000;
 // 3. Create PEs and add these into a list.
 peList.add(new Pe(0, new PeProvisionerSimple(mips))); // need to store Pe  id and MIPS Rating
 //4. Create Host with its id and list of PEs and add them to the list of  machines
 int hostId=0;
 int ram = 2048; //host memory (MB)
 long storage = 1000000; //host storage
 int bw = 10000;
 //in this example, the VMAllocatonPolicy in use is SpaceShared. It means  that only one VM
 //is allowed to run on each Pe. As each Host has only one Pe, only one VM  can run on each Host.
 hostList.add(
 new Host(
 hostId,
 new RamProvisionerSimple(ram),
 new BwProvisionerSimple(bw),
 storage,
 peList,
 new VmSchedulerSpaceShared(peList)
 )
 ); // This is our machine
 // 5. Create a DatacenterCharacteristics object that stores the  // properties of a data center:
architecture, OS, list of  // Machines, allocation policy: time- or space-shared, time zone  //
and its price (G$/Pe time unit).
 String arch = "x86"; // system architecture
 String os = "Linux"; // operating system
 String vmm = "Xen";
 double time_zone = 10.0; // time zone this resource located  double cost = 3.0; // the cost of using
processing in this  resource
 double costPerMem = 0.05; // the cost of using memory in this  resource
 double costPerStorage = 0.001; // the cost of using storage in this  resource
 double costPerBw = 0.0; // the cost of using bw in this resource  LinkedList<Storage> storageList =
new LinkedList<Storage>(); //we are  not adding SAN devices by now
 DatacenterCharacteristics characteristics = new
DatacenterCharacteristics(
```

```
 arch, os, vmm, hostList, time_zone, cost, costPerMem,   costPerStorage, costPerBw);
 // 6. Finally, we need to create a PowerDatacenter object.  Datacenter datacenter
= null;
 try {
 datacenter = new Datacenter(name, characteristics, new
VmAllocationPolicySimple(hostList), storageList, 0);
 } catch (Exception e) {
 e.printStackTrace();
 }
 return datacenter;
 }
 //We strongly encourage users to develop their own broker policies, to submit  vms and cloudlets
according
 //to the specific rules of the simulated scenario
 private static DatacenterBroker createBroker(int id){
 DatacenterBroker broker = null;
 try {
 broker = new DatacenterBroker("Broker"+id);
 } catch (Exception e) {
 e.printStackTrace();
 return null;
 }
 return broker;
 }
 /**
 * Prints the Cloudlet objects
 * @param list list of Cloudlets
 */
 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");
 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");
 DecimalFormat dft = new DecimalFormat("###.##");   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 –




```
Simulation completed.
Simulation completed.
==============> User 4
========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time
    0        SUCCESS        3             0        160        33         193
==============> User 5
========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time
    0        SUCCESS        2             0        160        17.1         177.1
NetworkExample3 finished!
```

Experiment – 4

Aim - Program to create two datacenters with one host each and run two cloudlets on them.two users with network topology on them. Software Used – Eclipse

Package Used – Cloudsim(Framework)

Code –

```
* Title: CloudSim Toolkit
 * Description: CloudSim (Cloud Simulation) Toolkit for Modeling and Simulation  * of Clouds
 * Licence: GPL - http://www.gnu.org/copyleft/gpl.html
 *
 * Copyright (c) 2009, The University of Melbourne, Australia  */
package org.cloudbus.cloudsim.examples;
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.VmSchedulerSpaceShared;
import org.cloudbus.cloudsim.core.CloudSim;
import org.cloudbus.cloudsim.provisioners.BwProvisionerSimple;  import
org.cloudbus.cloudsim.provisioners.PeProvisionerSimple;  import
org.cloudbus.cloudsim.provisioners.RamProvisionerSimple;  /**
 * A simple example showing how to create
 * two datacenters with one host each and
 * run two cloudlets on them.
 */
public class CloudSimExample4 {
 /** The cloudlet list. */
```

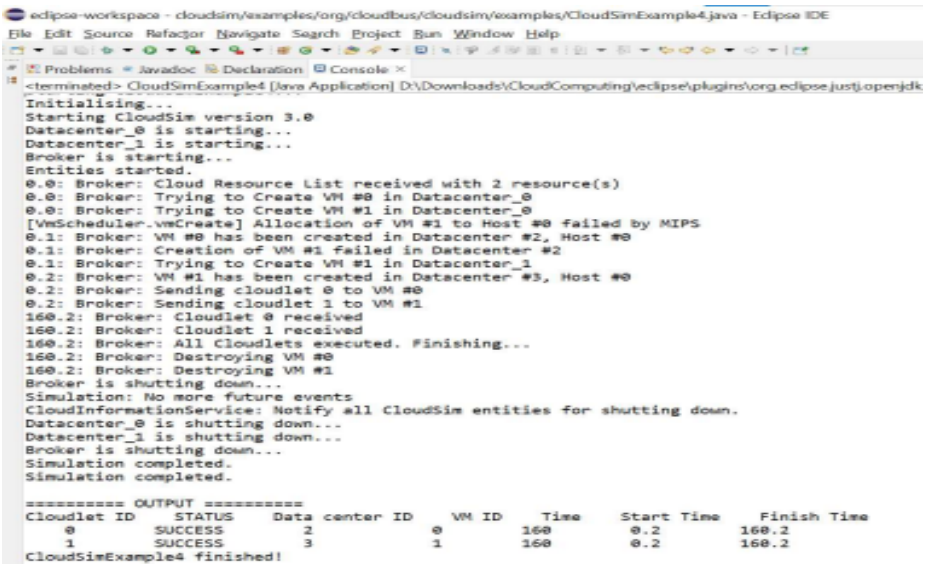
```
 private static List<Cloudlet> cloudletList;
 /** The vmlist. */
 private static List<Vm> vmlist;
 /**
 * Creates main() to run this example
 */
 public static void main(String[] args) {
 Log.printLine("Starting CloudSimExample4...");
 try {
 // First step: Initialize the CloudSim package. It should be called  // before creating any entities.
 int num_user = 1; // number of cloud users
 Calendar calendar = Calendar.getInstance();
 boolean trace_flag = false; // mean trace events
 // Initialize the GridSim library
 CloudSim.init(num_user, calendar, trace_flag);
 // Second step: Create Datacenters
 //Datacenters are the resource providers in CloudSim. We need at list  one of them to run a
CloudSim simulation
 @SuppressWarnings("unused")
 Datacenter datacenter0 = createDatacenter("Datacenter_0");
@SuppressWarnings("unused")
 Datacenter datacenter1 = createDatacenter("Datacenter_1");   //Third step: Create
Broker
 DatacenterBroker broker = createBroker();
 int brokerId = broker.getId();
 //Fourth step: Create one virtual machine
 vmlist = new ArrayList<Vm>();
 //VM description
 int vmid = 0;
 int mips = 250;
 long size = 10000; //image size (MB)
 int ram = 512; //vm memory (MB)
 long bw = 1000;
 int pesNumber = 1; //number of cpus
 String vmm = "Xen"; //VMM name
 //create two VMs
 Vm vm1 = new Vm(vmid, brokerId, mips, pesNumber, ram, bw, size, vmm,  new
CloudletSchedulerTimeShared());
 vmid++;
 Vm vm2 = new Vm(vmid, brokerId, mips, pesNumber, ram, bw, size, vmm,  new
CloudletSchedulerTimeShared());
 //add the VMs to the vmList
 vmlist.add(vm1);
 vmlist.add(vm2);
 //submit vm list to the broker
 broker.submitVmList(vmlist);
 //Fifth step: Create two Cloudlets
 cloudletList = new ArrayList<Cloudlet>();
```

```
 //Cloudlet properties
 int id = 0;
 long length = 40000;
 long fileSize = 300;
 long outputSize = 300;
 UtilizationModel utilizationModel = new UtilizationModelFull();   Cloudlet cloudlet1 = new
Cloudlet(id, length, pesNumber, fileSize,  outputSize, utilizationModel, utilizationModel,
utilizationModel);   cloudlet1.setUserId(brokerId);
 id++;
 Cloudlet cloudlet2 = new Cloudlet(id, length, pesNumber, fileSize,  outputSize, utilizationModel,
utilizationModel, utilizationModel);   cloudlet2.setUserId(brokerId);
 //add the cloudlets to the list
 cloudletList.add(cloudlet1);
 cloudletList.add(cloudlet2);
 //submit cloudlet list to the broker
 broker.submitCloudletList(cloudletList);
 //bind the cloudlets to the vms. This way, the broker  // will submit the bound cloudlets only to
the specific VM  broker.bindCloudletToVm(cloudlet1.getCloudletId(),vm1.getId());
broker.bindCloudletToVm(cloudlet2.getCloudletId(),vm2.getId());   // Sixth step: Starts the
simulation
 CloudSim.startSimulation();
 // Final step: Print results when simulation is over  List<Cloudlet> newList =
broker.getCloudletReceivedList();   CloudSim.stopSimulation();
 printCloudletList(newList);
 Log.printLine("CloudSimExample4 finished!");
 }
 catch (Exception e) {
 e.printStackTrace();
 Log.printLine("The simulation has been terminated due to an  unexpected error");
 }
 }
 private static Datacenter createDatacenter(String name){
 // Here are the steps needed to create a PowerDatacenter:  // 1. We need to
create a list to store
 // our machine
 List<Host> hostList = new ArrayList<Host>();
 // 2. A Machine contains one or more PEs or CPUs/Cores.  // In this example,
it will have only one core.
 List<Pe> peList = new ArrayList<Pe>();
 int mips = 1000;
 // 3. Create PEs and add these into a list.
 peList.add(new Pe(0, new PeProvisionerSimple(mips))); // need to store Pe  id and MIPS Rating
 //4. Create Host with its id and list of PEs and add them to the list of  machines
 int hostId=0;
 int ram = 2048; //host memory (MB)
 long storage = 1000000; //host storage
 int bw = 10000;
 //in this example, the VMAllocatonPolicy in use is SpaceShared. It means  that only one VM
```

```
//is allowed to run on each Pe. As each Host has only one Pe, only one VM  can run on each Host.
hostList.add(
new Host(
hostId,
new RamProvisionerSimple(ram),
new BwProvisionerSimple(bw),
storage,
peList,
new VmSchedulerSpaceShared(peList)
)
); // This is our first machine
// 5. Create a DatacenterCharacteristics object that stores the  // properties of a data center:
architecture, OS, list of  // Machines, allocation policy: time- or space-shared, time zone  //
and its price (G$/Pe time unit).
String arch = "x86"; // system architecture
String os = "Linux"; // operating system
String vmm = "Xen";
double time_zone = 10.0; // time zone this resource located  double cost = 3.0; // the cost of using
processing in this  resource
double costPerMem = 0.05; // the cost of using memory in this  resource
double costPerStorage = 0.001; // the cost of using storage in this  resource
double costPerBw = 0.0; // the cost of using bw in this resource  LinkedList<Storage> storageList =
new LinkedList<Storage>(); //we are  not adding SAN devices by now
DatacenterCharacteristics characteristics = new
DatacenterCharacteristics(
arch, os, vmm, hostList, time_zone, cost, costPerMem,  costPerStorage, costPerBw);
// 6. Finally, we need to create a PowerDatacenter object.  Datacenter datacenter
= null;
try {
datacenter = new Datacenter(name, characteristics, new
VmAllocationPolicySimple(hostList), storageList, 0);
} catch (Exception e) {
e.printStackTrace();
}
return datacenter;
}
//We strongly encourage users to develop their own broker policies, to submit  vms and cloudlets
according
//to the specific rules of the simulated scenario
private static DatacenterBroker createBroker(){
DatacenterBroker broker = null;
try {
broker = new DatacenterBroker("Broker");
} catch (Exception e) {
e.printStackTrace();
return null;
}
return broker;
}
```

```
/**
* Prints the Cloudlet objects
* @param list list of Cloudlets
*/
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 –



Experiment – 5

Aim - Program to create two datacenters with one host each and run cloudlets of two users on them.

Software Used – Eclipse

Package Used – Cloudsim(Framework)

Code –

```
* Title: CloudSim Toolkit
 * Description: CloudSim (Cloud Simulation) Toolkit for Modeling and Simulation  * of Clouds
 * Licence: GPL - http://www.gnu.org/copyleft/gpl.html
 *
 * Copyright (c) 2009, The University of Melbourne, Australia  */
package org.cloudbus.cloudsim.examples;
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.VmSchedulerSpaceShared;
import org.cloudbus.cloudsim.core.CloudSim;
import org.cloudbus.cloudsim.provisioners.BwProvisionerSimple;  import
org.cloudbus.cloudsim.provisioners.PeProvisionerSimple;  import
org.cloudbus.cloudsim.provisioners.RamProvisionerSimple;  /**
 * A simple example showing how to create
 * two datacenters with one host each and
 * run cloudlets of two users on them.
 */
public class CloudSimExample5 {
 /** The cloudlet lists. */
 private static List<Cloudlet> cloudletList1;
```

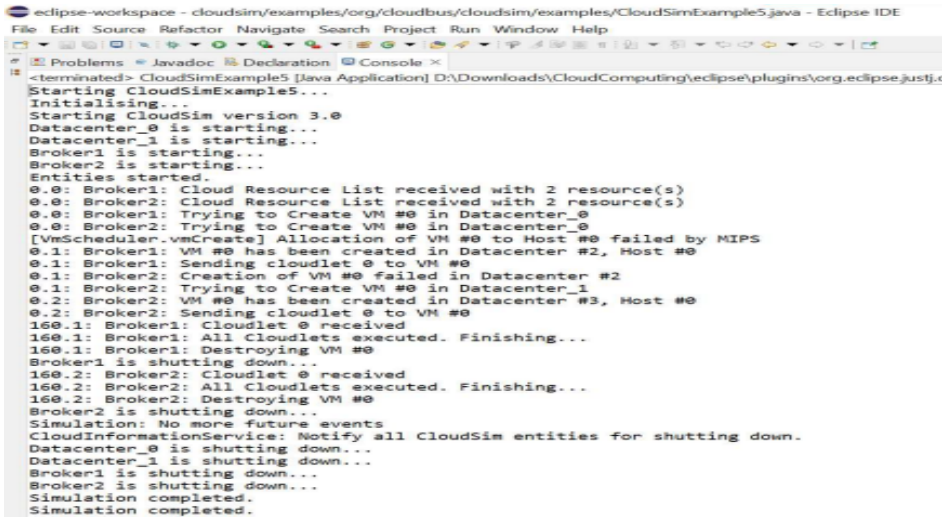
```
 private static List<Cloudlet> cloudletList2;
 /** The vmlists. */
 private static List<Vm> vmlist1;
 private static List<Vm> vmlist2;
 /**
 * Creates main() to run this example
 */
 public static void main(String[] args) {
 Log.printLine("Starting CloudSimExample5...");
 try {
 // First step: Initialize the CloudSim package. It should be called  // before creating any entities.
 int num_user = 2; // number of cloud users
 Calendar calendar = Calendar.getInstance();
 boolean trace_flag = false; // mean trace events
 // Initialize the CloudSim library
 CloudSim.init(num_user, calendar, trace_flag);
 // Second step: Create Datacenters
 //Datacenters are the resource providers in CloudSim. We need at list  one of them to run a
CloudSim simulation
 @SuppressWarnings("unused")
 Datacenter datacenter0 = createDatacenter("Datacenter_0");
@SuppressWarnings("unused")
 Datacenter datacenter1 = createDatacenter("Datacenter_1");   //Third step: Create
Brokers
 DatacenterBroker broker1 = createBroker(1);
 int brokerId1 = broker1.getId();
 DatacenterBroker broker2 = createBroker(2);
 int brokerId2 = broker2.getId();
 //Fourth step: Create one virtual machine for each broker/user  vmlist1 = new
ArrayList<Vm>();
 vmlist2 = new ArrayList<Vm>();
 //VM description
 int vmid = 0;
 int mips = 250;
 long size = 10000; //image size (MB)
 int ram = 512; //vm memory (MB)
 long bw = 1000;
 int pesNumber = 1; //number of cpus
 String vmm = "Xen"; //VMM name
 //create two VMs: the first one belongs to user1
 Vm vm1 = new Vm(vmid, brokerId1, mips, pesNumber, ram, bw, size, vmm,  new
CloudletSchedulerTimeShared());
 //the second VM: this one belongs to user2
 Vm vm2 = new Vm(vmid, brokerId2, mips, pesNumber, ram, bw, size, vmm,  new
CloudletSchedulerTimeShared());
 //add the VMs to the vmlists
 vmlist1.add(vm1);
 vmlist2.add(vm2);
```

```
 //submit vm list to the broker
 broker1.submitVmList(vmlist1);
 broker2.submitVmList(vmlist2);
 //Fifth step: Create two Cloudlets
 cloudletList1 = new ArrayList<Cloudlet>();
 cloudletList2 = new ArrayList<Cloudlet>();
 //Cloudlet properties
 int id = 0;
 long length = 40000;
 long fileSize = 300;
 long outputSize = 300;
 UtilizationModel utilizationModel = new UtilizationModelFull();   Cloudlet cloudlet1 = new
Cloudlet(id, length, pesNumber, fileSize,  outputSize, utilizationModel, utilizationModel,
utilizationModel);   cloudlet1.setUserId(brokerId1);
 Cloudlet cloudlet2 = new Cloudlet(id, length, pesNumber, fileSize,  outputSize, utilizationModel,
utilizationModel, utilizationModel);   cloudlet2.setUserId(brokerId2);
 //add the cloudlets to the lists: each cloudlet belongs to one user  cloudletList1.add(cloudlet1);
 cloudletList2.add(cloudlet2);
 //submit cloudlet list to the brokers
 broker1.submitCloudletList(cloudletList1);
 broker2.submitCloudletList(cloudletList2);
 // Sixth step: Starts the simulation
 CloudSim.startSimulation();
 // Final step: Print results when simulation is over  List<Cloudlet> newList1 =
broker1.getCloudletReceivedList();   List<Cloudlet> newList2 =
broker2.getCloudletReceivedList();   CloudSim.stopSimulation();
 Log.print("==============> User "+brokerId1+" ");
printCloudletList(newList1);
 Log.print("==============> User "+brokerId2+" ");
printCloudletList(newList2);
 Log.printLine("CloudSimExample5 finished!");
 }
 catch (Exception e) {
 e.printStackTrace();
 Log.printLine("The simulation has been terminated due to an  unexpected error");
 }
 }
 private static Datacenter createDatacenter(String name){
 // Here are the steps needed to create a PowerDatacenter:  // 1. We need to
create a list to store
 // our machine
 List<Host> hostList = new ArrayList<Host>();
 // 2. A Machine contains one or more PEs or CPUs/Cores.  // In this example,
it will have only one core.
 List<Pe> peList = new ArrayList<Pe>();
 int mips=1000;
 // 3. Create PEs and add these into a list.
 peList.add(new Pe(0, new PeProvisionerSimple(mips))); // need to store Pe  id and MIPS Rating
```

```
//4. Create Host with its id and list of PEs and add them to the list of  machines
int hostId=0;
int ram = 2048; //host memory (MB)
long storage = 1000000; //host storage
int bw = 10000;
//in this example, the VMAllocatonPolicy in use is SpaceShared. It means  that only one VM
//is allowed to run on each Pe. As each Host has only one Pe, only one VM  can run on each Host.
hostList.add(
new Host(
hostId,
new RamProvisionerSimple(ram),
new BwProvisionerSimple(bw),
storage,
peList,
new VmSchedulerSpaceShared(peList)
)
); // This is our first machine
// 5. Create a DatacenterCharacteristics object that stores the  // properties of a data center:
architecture, OS, list of  // Machines, allocation policy: time- or space-shared, time zone  //
and its price (G$/Pe time unit).
String arch = "x86"; // system architecture
String os = "Linux"; // operating system
String vmm = "Xen";
double time_zone = 10.0; // time zone this resource located  double cost = 3.0; // the cost of using
processing in this  resource
double costPerMem = 0.05; // the cost of using memory in this  resource
double costPerStorage = 0.001; // the cost of using storage in this  resource
double costPerBw = 0.0; // the cost of using bw in this resource  LinkedList<Storage> storageList =
new LinkedList<Storage>(); //we are  not adding SAN devices by now
DatacenterCharacteristics characteristics = new
DatacenterCharacteristics(
arch, os, vmm, hostList, time_zone, cost, costPerMem,  costPerStorage, costPerBw);
// 6. Finally, we need to create a PowerDatacenter object.  Datacenter datacenter
= null;
try {
datacenter = new Datacenter(name, characteristics, new
VmAllocationPolicySimple(hostList), storageList, 0);
} catch (Exception e) {
e.printStackTrace();
}
return datacenter;
}
//We strongly encourage users to develop their own broker policies, to submit  vms and cloudlets
according
//to the specific rules of the simulated scenario
private static DatacenterBroker createBroker(int id){
DatacenterBroker broker = null;
try {
broker = new DatacenterBroker("Broker"+id);
```

```
} catch (Exception e) {
e.printStackTrace();
return null;
}
return broker;
}
/**
* Prints the Cloudlet objects
* @param list list of Cloudlets
*/
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 –



```
==============> User 4
========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time
    0        SUCCESS        2             0        160        0.1        160.1
==============> User 5
========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time
    0        SUCCESS        3             0        160        0.2        160.2
CloudSimExample5 finished!
```

Experiment – 6

Aim - Program to create a datacenter with two hosts and run two cloudlets on it. Software Used – Eclipse

Package Used – Cloudsim(Framework)

Code –

```
* Title: CloudSim Toolkit
 * Description: CloudSim (Cloud Simulation) Toolkit for Modeling and Simulation  * of Clouds
 * Licence: GPL - http://www.gnu.org/copyleft/gpl.html
 *
 * Copyright (c) 2009, The University of Melbourne, Australia  */
package org.cloudbus.cloudsim.examples;
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;  /**
 * A simple example showing how to create
 * a datacenter with two hosts and run two
 * cloudlets on it. The cloudlets run in
 * VMs with different MIPS requirements.
 * The cloudlets will take different time
 * to complete the execution depending on
 * the requested VM performance.
 */
public class CloudSimExample3 {
```

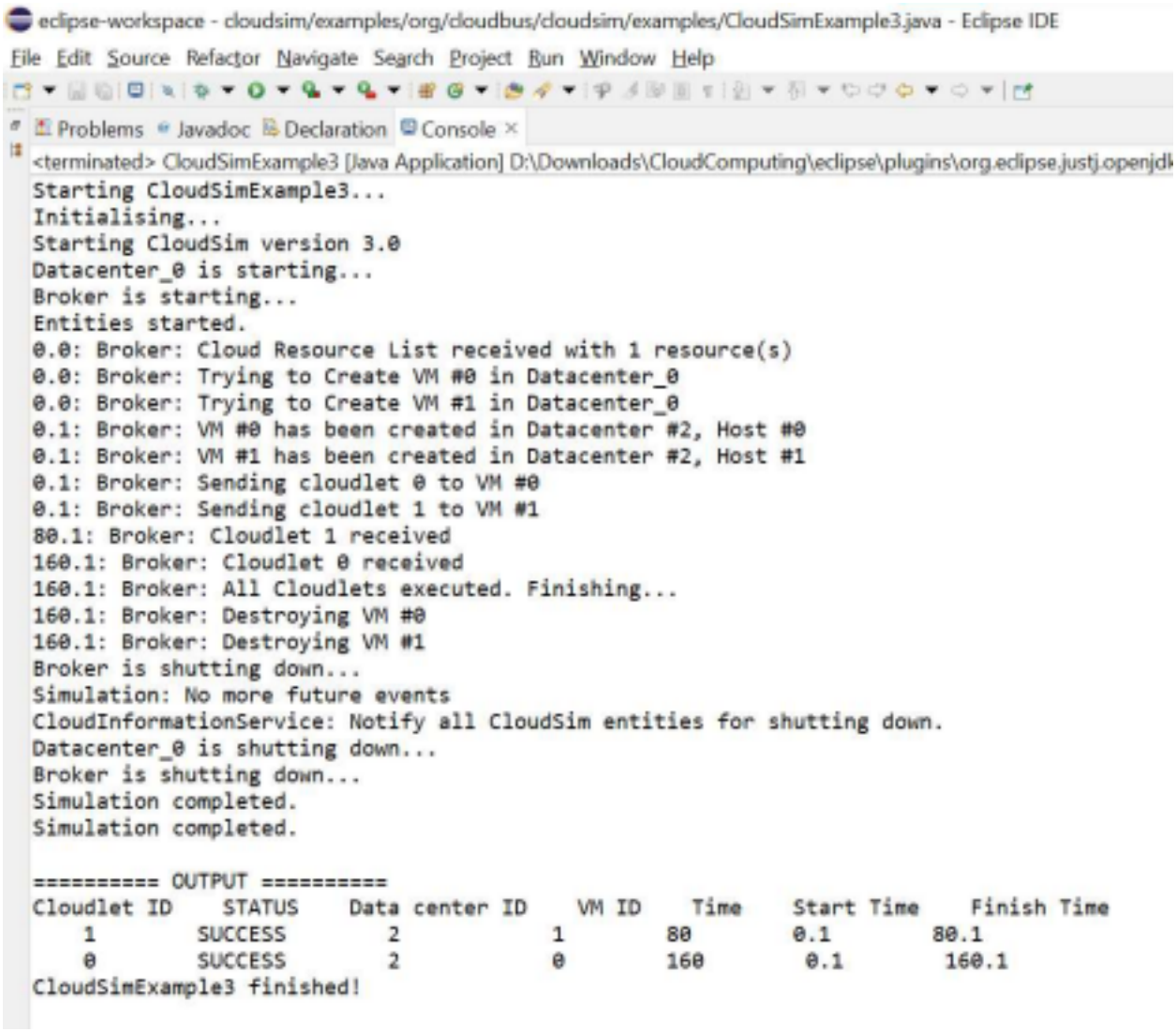
```
/** The cloudlet list. */
private static List<Cloudlet> cloudletList;
/** The vmlist. */
private static List<Vm> vmlist;
/**
* Creates main() to run this example
*/
public static void main(String[] args) {
Log.printLine("Starting CloudSimExample3...");
try {
// First step: Initialize the CloudSim package. It should be called  // before creating any entities.
int num_user = 1; // number of cloud users
Calendar calendar = Calendar.getInstance();
boolean trace_flag = false; // mean trace events
// Initialize the CloudSim library
CloudSim.init(num_user, calendar, trace_flag);
// Second step: Create Datacenters
//Datacenters are the resource providers in CloudSim. We need at list  one of them to run a
CloudSim simulation
@SuppressWarnings("unused")
Datacenter datacenter0 = createDatacenter("Datacenter_0");   //Third step: Create
Broker
DatacenterBroker broker = createBroker();
int brokerId = broker.getId();
//Fourth step: Create one virtual machine
vmlist = new ArrayList<Vm>();
//VM description
int vmid = 0;
int mips = 250;
long size = 10000; //image size (MB)
int ram = 2048; //vm memory (MB)
long bw = 1000;
int pesNumber = 1; //number of cpus
String vmm = "Xen"; //VMM name
//create two VMs
Vm vm1 = new Vm(vmid, brokerId, mips, pesNumber, ram, bw, size, vmm,  new
CloudletSchedulerTimeShared());
//the second VM will have twice the priority of VM1 and so will  receive twice CPU time
vmid++;
Vm vm2 = new Vm(vmid, brokerId, mips * 2, pesNumber, ram, bw, size,  vmm, new
CloudletSchedulerTimeShared());
//add the VMs to the vmList
vmlist.add(vm1);
vmlist.add(vm2);
//submit vm list to the broker
broker.submitVmList(vmlist);
//Fifth step: Create two Cloudlets
cloudletList = new ArrayList<Cloudlet>();
```

```
//Cloudlet properties
int id = 0;
long length = 40000;
long fileSize = 300;
long outputSize = 300;
UtilizationModel utilizationModel = new UtilizationModelFull();   Cloudlet cloudlet1 = new
Cloudlet(id, length, pesNumber, fileSize,  outputSize, utilizationModel, utilizationModel,
utilizationModel);   cloudlet1.setUserId(brokerId);
id++;
Cloudlet cloudlet2 = new Cloudlet(id, length, pesNumber, fileSize,  outputSize, utilizationModel,
utilizationModel, utilizationModel);   cloudlet2.setUserId(brokerId);
//add the cloudlets to the list
cloudletList.add(cloudlet1);
cloudletList.add(cloudlet2);
//submit cloudlet list to the broker
broker.submitCloudletList(cloudletList);
//bind the cloudlets to the vms. This way, the broker  // will submit the bound cloudlets only to
the specific VM  broker.bindCloudletToVm(cloudlet1.getCloudletId(),vm1.getId());
broker.bindCloudletToVm(cloudlet2.getCloudletId(),vm2.getId());   // Sixth step: Starts the
simulation
CloudSim.startSimulation();
// Final step: Print results when simulation is over  List<Cloudlet> newList =
broker.getCloudletReceivedList();   CloudSim.stopSimulation();
printCloudletList(newList);
Log.printLine("CloudSimExample3 finished!");
}
catch (Exception e) {
e.printStackTrace();
Log.printLine("The simulation has been terminated due to an  unexpected error");
}
}
private static Datacenter createDatacenter(String name){
// Here are the steps needed to create a PowerDatacenter:  // 1. We need to
create a list to store
// our machine
List<Host> hostList = new ArrayList<Host>();
// 2. A Machine contains one or more PEs or CPUs/Cores.  // In this example,
it will have only one core.
List<Pe> peList = new ArrayList<Pe>();
int mips = 1000;
// 3. Create PEs and add these into a list.
peList.add(new Pe(0, new PeProvisionerSimple(mips))); // need to store Pe  id and MIPS Rating
//4. Create Hosts with its id and list of PEs and add them to the list of  machines
int hostId=0;
int ram = 2048; //host memory (MB)
long storage = 1000000; //host storage
int bw = 10000;
hostList.add(
```

```
new Host(
hostId,
new RamProvisionerSimple(ram),
new BwProvisionerSimple(bw),
storage,
peList,
new VmSchedulerTimeShared(peList)
)
); // This is our first machine
//create another machine in the Data center
List<Pe> peList2 = new ArrayList<Pe>();
peList2.add(new Pe(0, new PeProvisionerSimple(mips)));   hostId++;
hostList.add(
new Host(
hostId,
new RamProvisionerSimple(ram),
new BwProvisionerSimple(bw),
storage,
peList2,
new VmSchedulerTimeShared(peList2)
)
); // This is our second machine
// 5. Create a DatacenterCharacteristics object that stores the  // properties of a data center:
architecture, OS, list of  // Machines, allocation policy: time- or space-shared, time zone  //
and its price (G$/Pe time unit).
String arch = "x86"; // system architecture
String os = "Linux"; // operating system
String vmm = "Xen";
double time_zone = 10.0; // time zone this resource located  double cost = 3.0; // the cost of using
processing in this  resource
double costPerMem = 0.05; // the cost of using memory in this  resource
double costPerStorage = 0.001; // the cost of using storage in this  resource
double costPerBw = 0.0; // the cost of using bw in this resource  LinkedList<Storage> storageList =
new LinkedList<Storage>(); //we are  not adding SAN devices by now
DatacenterCharacteristics characteristics = new
DatacenterCharacteristics(
arch, os, vmm, hostList, time_zone, cost, costPerMem,  costPerStorage, costPerBw);
// 6. Finally, we need to create a PowerDatacenter object.  Datacenter datacenter
= null;
try {
datacenter = new Datacenter(name, characteristics, new
VmAllocationPolicySimple(hostList), storageList, 0);
} catch (Exception e) {
e.printStackTrace();
}
return datacenter;
}
//We strongly encourage users to develop their own broker policies, to submit  vms and cloudlets
```

```
according
 //to the specific rules of the simulated scenario
 private static DatacenterBroker createBroker(){
 DatacenterBroker broker = null;
 try {
 broker = new DatacenterBroker("Broker");
 } catch (Exception e) {
 e.printStackTrace();
 return null;
 }
 return broker;
 }
 /**
 * Prints the Cloudlet objects
 * @param list list of Cloudlets
 */
 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 –


eclipse-workspace - cloudsim/examples/org/cloudbus/cloudsim/examples/CloudSimExample3.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Problems Javadoc Declaration Console
<terminated> CloudSimExample3 [Java Application] D:\Downloads\CloudComputing\eclipse\plugins\org.eclipse.justj.openjdk
Starting CloudSimExample3...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 1 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.0: Broker: Trying to Create VM #1 in Datacenter_0
0.1: Broker: VM #0 has been created in Datacenter #2, Host #0
0.1: Broker: VM #1 has been created in Datacenter #2, Host #1
0.1: Broker: Sending cloudlet 0 to VM #0
0.1: Broker: Sending cloudlet 1 to VM #1
80.1: Broker: Cloudlet 1 received
160.1: Broker: Cloudlet 0 received
160.1: Broker: All Cloudlets executed. Finishing...
160.1: Broker: Destroying VM #0
160.1: Broker: Destroying VM #1
Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.
========== OUTPUT ==========
Cloudlet ID STATUS Data center ID VM ID Time Start Time Finish Time
1 SUCCESS 2 1 80 0.1 80.1
0 SUCCESS 2 0 160 0.1 160.1
CloudSimExample3 finished!

Experiment – 7

Aim - Program to create scalable simulations.

Software Used – Eclipse

Package Used – Cloudsim(Framework)

Code –

```
* Title: CloudSim Toolkit
 * Description: CloudSim (Cloud Simulation) Toolkit for Modeling and Simulation  * of Clouds
 * Licence: GPL - http://www.gnu.org/copyleft/gpl.html
 *
 * Copyright (c) 2009, The University of Melbourne, Australia  */
package org.cloudbus.cloudsim.examples;
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;  /**
 * An example showing how to create
 * scalable simulations.
 */
public class CloudSimExample6 {
 /** The cloudlet list. */
 private static List<Cloudlet> cloudletList;
 /** The vmlist. */
 private static List<Vm> vmlist;
 private static List<Vm> createVM(int userId, int vms) {   //Creates a container to store VMs. This
```

```
list is passed to the broker  later
 LinkedList<Vm> list = new LinkedList<Vm>();
 //VM Parameters
 long size = 10000; //image size (MB)
 int ram = 512; //vm memory (MB)
 int mips = 1000;
 long bw = 1000;
 int pesNumber = 1; //number of cpus
 String vmm = "Xen"; //VMM name
 //create VMs
 Vm[] vm = new Vm[vms];
 for(int i=0;i<vms;i++){
 vm[i] = new Vm(i, userId, mips, pesNumber, ram, bw, size, vmm, new
CloudletSchedulerTimeShared());
 //for creating a VM with a space shared scheduling policy for  cloudlets:
 //vm[i] = Vm(i, userId, mips, pesNumber, ram, bw, size, priority,  vmm, new
CloudletSchedulerSpaceShared());
 list.add(vm[i]);
 }
 return list;
 }
 private static List<Cloudlet> createCloudlet(int userId, int cloudlets){   // Creates a container to
store Cloudlets
 LinkedList<Cloudlet> list = new LinkedList<Cloudlet>();   //cloudlet parameters
 long length = 1000;
 long fileSize = 300;
 long outputSize = 300;
 int pesNumber = 1;
 UtilizationModel utilizationModel = new UtilizationModelFull();   Cloudlet[] cloudlet = new
Cloudlet[cloudlets];
 for(int i=0;i<cloudlets;i++){
 cloudlet[i] = new Cloudlet(i, length, pesNumber, fileSize,  outputSize, utilizationModel,
utilizationModel, utilizationModel);   // setting the owner of these Cloudlets
 cloudlet[i].setUserId(userId);
 list.add(cloudlet[i]);
 }
 return list;
 }
 /////////////////////////// STATIC METHODS ///////////////////////  /**
 * Creates main() to run this example
 */
 public static void main(String[] args) {
 Log.printLine("Starting CloudSimExample6...");
 try {
 // First step: Initialize the CloudSim package. It should be called  // before creating any entities.
 int num_user = 1; // number of grid users
 Calendar calendar = Calendar.getInstance();
 boolean trace_flag = false; // mean trace events
```

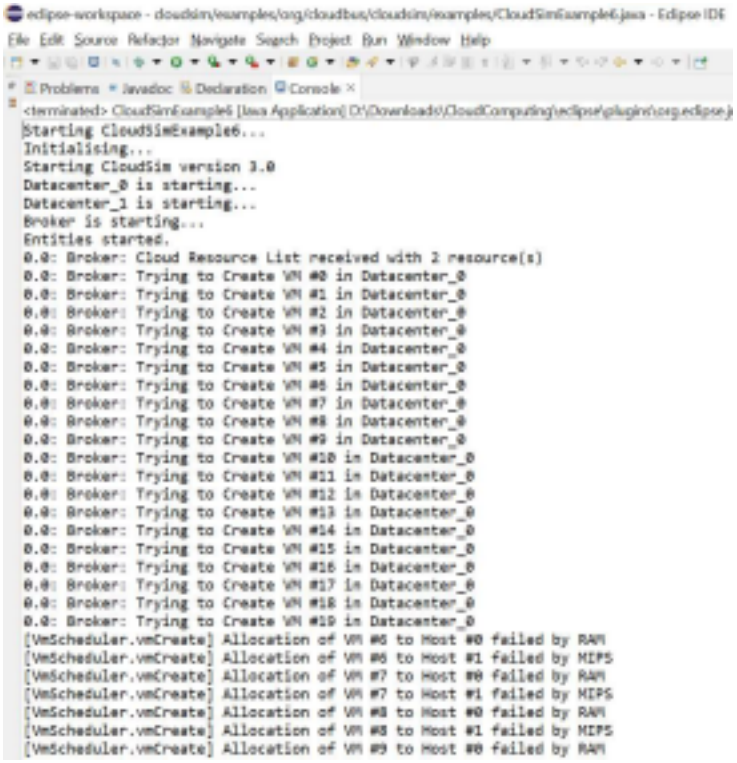
```
 // Initialize the CloudSim library
 CloudSim.init(num_user, calendar, trace_flag);
 // Second step: Create Datacenters
 //Datacenters are the resource providers in CloudSim. We need at list  one of them to run a
CloudSim simulation
 @SuppressWarnings("unused")
 Datacenter datacenter0 = createDatacenter("Datacenter_0");
@SuppressWarnings("unused")
 Datacenter datacenter1 = createDatacenter("Datacenter_1");   //Third step: Create
Broker
 DatacenterBroker broker = createBroker();
 int brokerId = broker.getId();
 //Fourth step: Create VMs and Cloudlets and send them to broker  vmlist =
createVM(brokerId,20); //creating 20 vms
 cloudletList = createCloudlet(brokerId,40); // creating 40 cloudlets  broker.submitVmList(vmlist);
 broker.submitCloudletList(cloudletList);
 // Fifth step: Starts the simulation
 CloudSim.startSimulation();
 // Final step: Print results when simulation is over  List<Cloudlet> newList =
broker.getCloudletReceivedList();   CloudSim.stopSimulation();
 printCloudletList(newList);
 Log.printLine("CloudSimExample6 finished!");
 }
 catch (Exception e)
 {
 e.printStackTrace();
 Log.printLine("The simulation has been terminated due to an  unexpected error");
 }
 }
 private static Datacenter createDatacenter(String name){
 // Here are the steps needed to create a PowerDatacenter:  // 1. We need to
create a list to store one or more
 // Machines
 List<Host> hostList = new ArrayList<Host>();
 // 2. A Machine contains one or more PEs or CPUs/Cores. Therefore, should  // create a list to store
these PEs before creating  // a Machine.
 List<Pe> peList1 = new ArrayList<Pe>();
 int mips = 1000;
 // 3. Create PEs and add these into the list.
 //for a quad-core machine, a list of 4 PEs is required:  peList1.add(new Pe(0, new
PeProvisionerSimple(mips))); // need to store  Pe id and MIPS Rating
  peList1.add(new  Pe(1,  new  PeProvisionerSimple(mips)));    peList1.add(new
Pe(2,  new  PeProvisionerSimple(mips)));         peList1.add(new  Pe(3,  new
PeProvisionerSimple(mips)));   //Another list, for a dual-core machine
 List<Pe> peList2 = new ArrayList<Pe>();
 peList2.add(new Pe(0, new PeProvisionerSimple(mips)));   peList2.add(new Pe(1, new
PeProvisionerSimple(mips)));   //4. Create Hosts with its id and list of PEs and add them to the list of
machines
 int hostId=0;
```

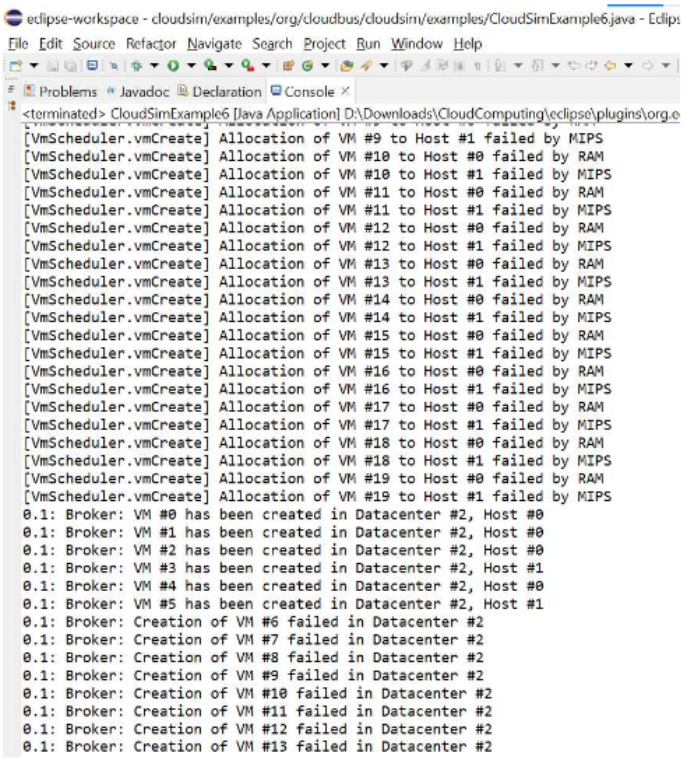
```
int ram = 2048; //host memory (MB)
long storage = 1000000; //host storage
int bw = 10000;
hostList.add(
new Host(
hostId,
new RamProvisionerSimple(ram),
new BwProvisionerSimple(bw),
storage,
peList1,
new VmSchedulerTimeShared(peList1)
)
); // This is our first machine
hostId++;
hostList.add(
new Host(
hostId,
new RamProvisionerSimple(ram),
new BwProvisionerSimple(bw),
storage,
peList2,
new VmSchedulerTimeShared(peList2)
)
); // Second machine
//To create a host with a space-shared allocation policy for PEs to VMs:  //hostList.add(
// new Host(
// hostId,
// new CpuProvisionerSimple(peList1),
// new RamProvisionerSimple(ram),
// new BwProvisionerSimple(bw),
// storage,
// new VmSchedulerSpaceShared(peList1)
// )
// );
//To create a host with a oportunistic space-shared allocation policy for  PEs to VMs:
//hostList.add(
// new Host(
// hostId,
// new CpuProvisionerSimple(peList1),
// new RamProvisionerSimple(ram),
// new BwProvisionerSimple(bw),
// storage,
// new VmSchedulerOportunisticSpaceShared(peList1)  // )
// );
// 5. Create a DatacenterCharacteristics object that stores the  // properties of a data center:
architecture, OS, list of  // Machines, allocation policy: time- or space-shared, time zone  //
and its price (G$/Pe time unit).
String arch = "x86"; // system architecture
```

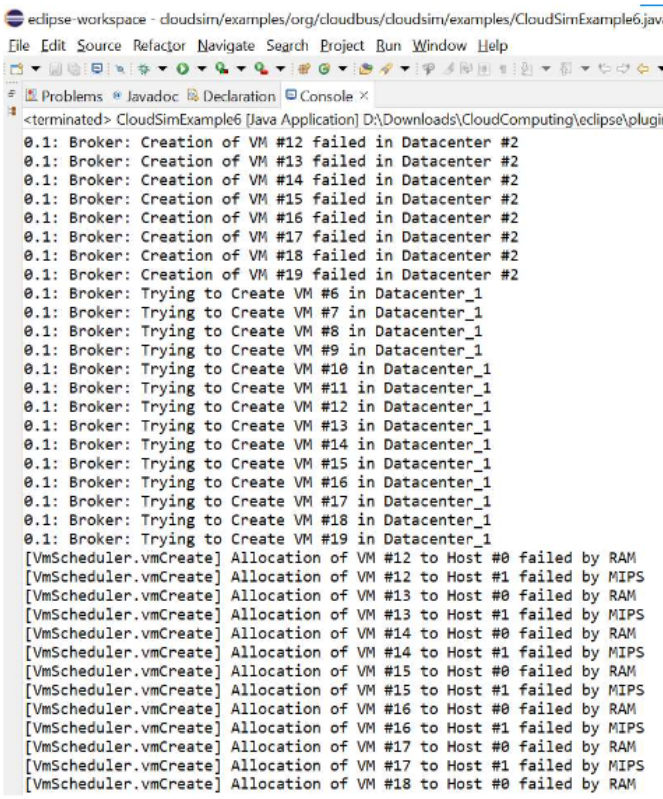
```
 String os = "Linux"; // operating system
 String vmm = "Xen";
 double time_zone = 10.0; // time zone this resource located  double cost = 3.0; // the cost of using
processing in this  resource
 double costPerMem = 0.05; // the cost of using memory in this  resource
 double costPerStorage = 0.1; // the cost of using storage in this  resource
                                        double costPerBw = 0.1; // the cost of using bw in this resource
 LinkedList<Storage> storageList = new LinkedList<Storage>(); //we are  not adding SAN devices by
now
 DatacenterCharacteristics characteristics = new
DatacenterCharacteristics(
 arch, os, vmm, hostList, time_zone, cost, costPerMem,  costPerStorage, costPerBw);
 // 6. Finally, we need to create a PowerDatacenter object.  Datacenter datacenter
= null;
 try {
 datacenter = new Datacenter(name, characteristics, new
VmAllocationPolicySimple(hostList), storageList, 0);
 } catch (Exception e) {
 e.printStackTrace();
 }
 return datacenter;
 }
 //We strongly encourage users to develop their own broker policies, to submit  vms and cloudlets
according
 //to the specific rules of the simulated scenario
 private static DatacenterBroker createBroker(){
 DatacenterBroker broker = null;
 try {
 broker = new DatacenterBroker("Broker");
 } catch (Exception e) {
 e.printStackTrace();
 return null;
 }
 return broker;
 }
 /**
 * Prints the Cloudlet objects
 * @param list list of Cloudlets
 */
 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 + indent + "Time" +  indent + "Start Time" + indent + "Finish Time");
 DecimalFormat dft = new DecimalFormat("###.##");
 for (int i = 0; i < size; i++) {
```

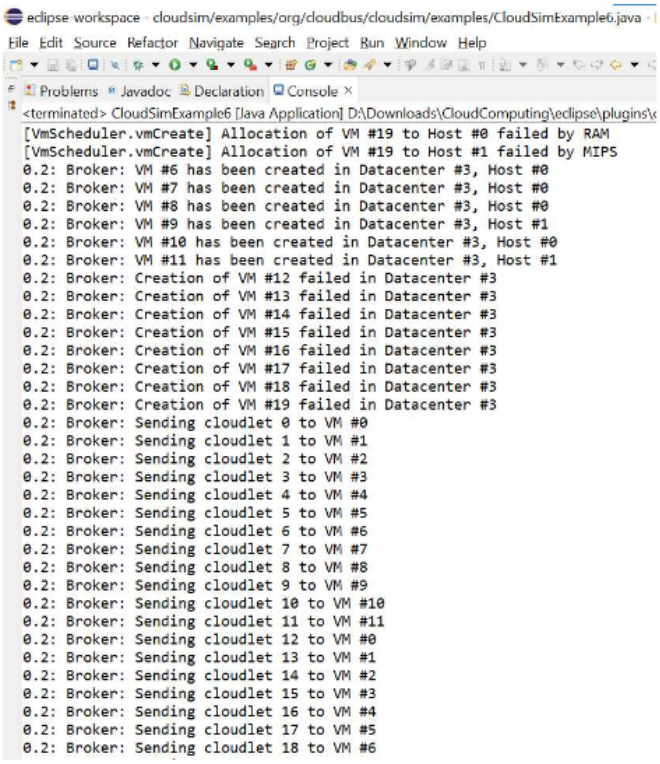
```
 cloudlet = list.get(i);
 Log.print(indent + cloudlet.getCloudletId() + indent + indent);   if (cloudlet.getCloudletStatus()
== Cloudlet.SUCCESS){   Log.print("SUCCESS");
 Log.printLine( indent + indent + cloudlet.getResourceId() +  indent + indent + indent +
cloudlet.getVmId() +
 indent + indent + indent +
dft.format(cloudlet.getActualCPUTime()) +
 indent + indent +
dft.format(cloudlet.getExecStartTime())+ indent + indent + indent +
dft.format(cloudlet.getFinishTime()));
 }
 }
 }
}
```

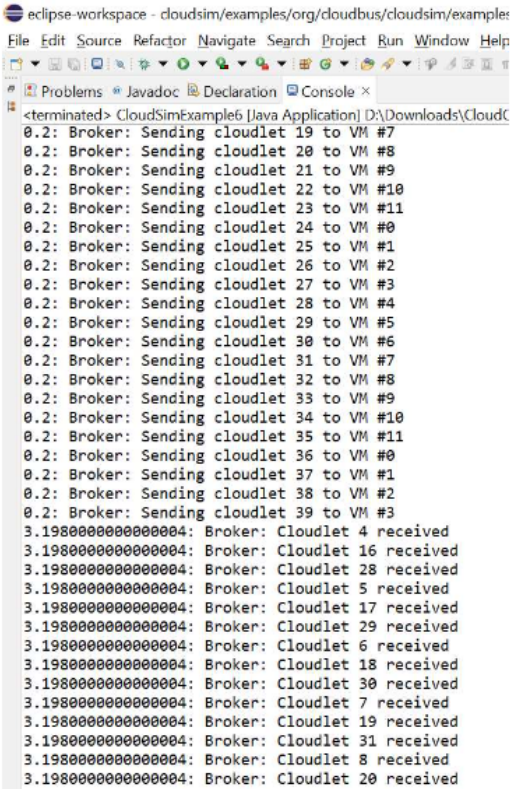
Output –

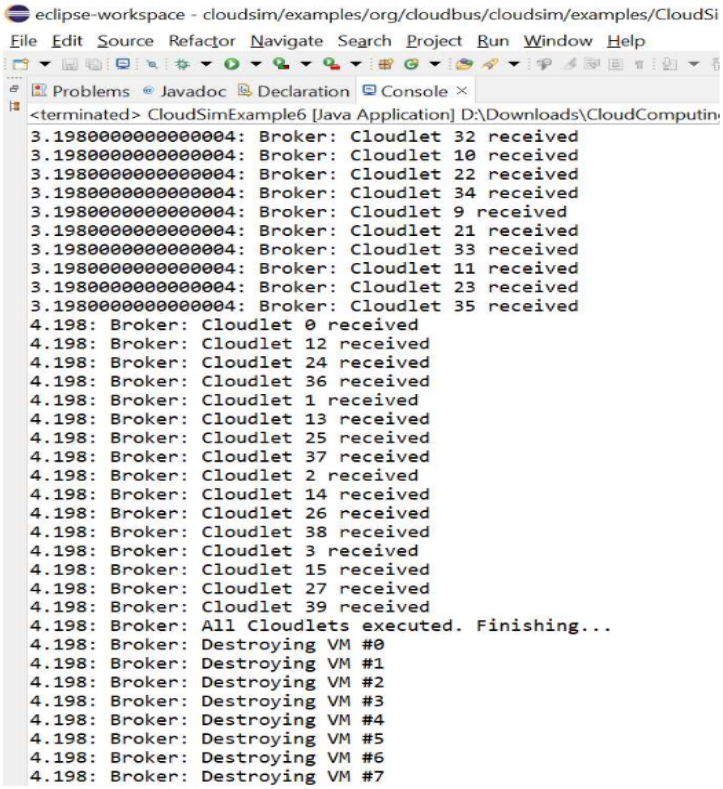


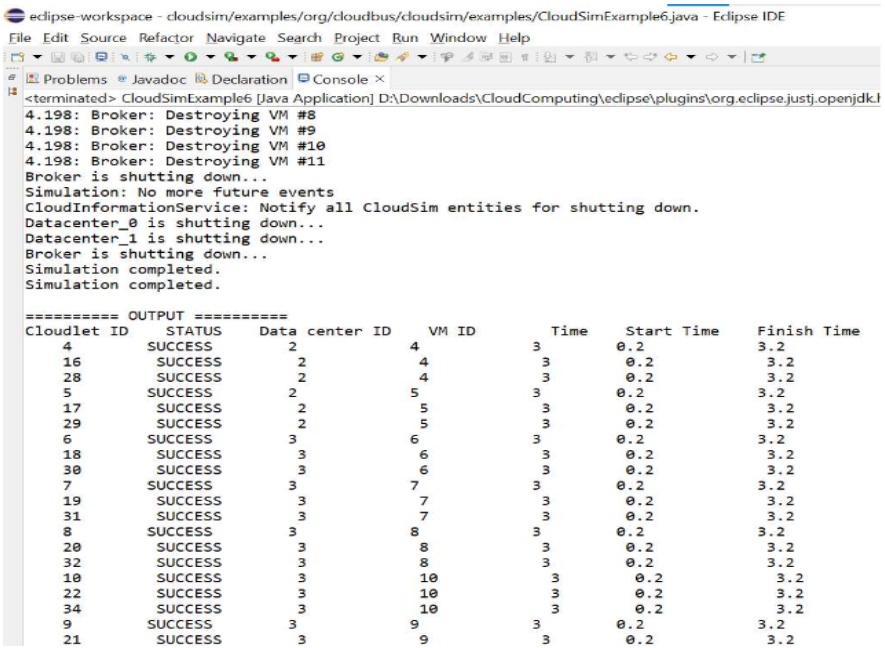

eclipse-workspace - cloudsim/examples/org/cloudbus/cloudsim/examples/CloudSimExample6.java - Eclips
File Edit Source Refactor Navigate Search Project Run Window Help
Problems Javadoc Declaration Console
<terminated> CloudSimExample6 [Java Application] D:\Downloads\CloudComputing\eclipse\plugins\org.e
[VmScheduler.vmCreate] Allocation of VM #9 to Host #1 failed by MIPS
[VmScheduler.vmCreate] Allocation of VM #10 to Host #0 failed by RAM
[VmScheduler.vmCreate] Allocation of VM #10 to Host #1 failed by MIPS
[VmScheduler.vmCreate] Allocation of VM #11 to Host #0 failed by RAM
[VmScheduler.vmCreate] Allocation of VM #11 to Host #1 failed by MIPS
[VmScheduler.vmCreate] Allocation of VM #12 to Host #0 failed by RAM
[VmScheduler.vmCreate] Allocation of VM #12 to Host #1 failed by MIPS
[VmScheduler.vmCreate] Allocation of VM #13 to Host #0 failed by RAM
[VmScheduler.vmCreate] Allocation of VM #13 to Host #1 failed by MIPS
[VmScheduler.vmCreate] Allocation of VM #14 to Host #0 failed by RAM
[VmScheduler.vmCreate] Allocation of VM #14 to Host #1 failed by MIPS
[VmScheduler.vmCreate] Allocation of VM #15 to Host #0 failed by RAM
[VmScheduler.vmCreate] Allocation of VM #15 to Host #1 failed by MIPS
[VmScheduler.vmCreate] Allocation of VM #16 to Host #0 failed by RAM
[VmScheduler.vmCreate] Allocation of VM #16 to Host #1 failed by MIPS
[VmScheduler.vmCreate] Allocation of VM #17 to Host #0 failed by RAM
[VmScheduler.vmCreate] Allocation of VM #17 to Host #1 failed by MIPS
[VmScheduler.vmCreate] Allocation of VM #18 to Host #0 failed by RAM
[VmScheduler.vmCreate] Allocation of VM #18 to Host #1 failed by MIPS
[VmScheduler.vmCreate] Allocation of VM #19 to Host #0 failed by RAM
[VmScheduler.vmCreate] Allocation of VM #19 to Host #1 failed by MIPS
0.1: Broker: VM #0 has been created in Datacenter #2, Host #0
0.1: Broker: VM #1 has been created in Datacenter #2, Host #0
0.1: Broker: VM #2 has been created in Datacenter #2, Host #0
0.1: Broker: VM #3 has been created in Datacenter #2, Host #1
0.1: Broker: VM #4 has been created in Datacenter #2, Host #0
0.1: Broker: VM #5 has been created in Datacenter #2, Host #1
0.1: Broker: Creation of VM #6 failed in Datacenter #2
0.1: Broker: Creation of VM #7 failed in Datacenter #2
0.1: Broker: Creation of VM #8 failed in Datacenter #2
0.1: Broker: Creation of VM #9 failed in Datacenter #2
0.1: Broker: Creation of VM #10 failed in Datacenter #2
0.1: Broker: Creation of VM #11 failed in Datacenter #2
0.1: Broker: Creation of VM #12 failed in Datacenter #2
0.1: Broker: Creation of VM #13 failed in Datacenter #2


eclipse-workspace - cloudsim/examples/org/cloudbus/cloudsim/examples/CloudSimExample6.jav
File Edit Source Refactor Navigate Search Project Run Window Help
Problems Javadoc Declaration Console
<terminated> CloudSimExample6 [Java Application] D:\Downloads\CloudComputing\eclipse\plugi
0.1: Broker: Creation of VM #12 failed in Datacenter #2
0.1: Broker: Creation of VM #13 failed in Datacenter #2
0.1: Broker: Creation of VM #14 failed in Datacenter #2
0.1: Broker: Creation of VM #15 failed in Datacenter #2
0.1: Broker: Creation of VM #16 failed in Datacenter #2
0.1: Broker: Creation of VM #17 failed in Datacenter #2
0.1: Broker: Creation of VM #18 failed in Datacenter #2
0.1: Broker: Creation of VM #19 failed in Datacenter #2
0.1: Broker: Trying to Create VM #6 in Datacenter_1
0.1: Broker: Trying to Create VM #7 in Datacenter_1
0.1: Broker: Trying to Create VM #8 in Datacenter_1
0.1: Broker: Trying to Create VM #9 in Datacenter_1
0.1: Broker: Trying to Create VM #10 in Datacenter_1
0.1: Broker: Trying to Create VM #11 in Datacenter_1
0.1: Broker: Trying to Create VM #12 in Datacenter_1
0.1: Broker: Trying to Create VM #13 in Datacenter_1
0.1: Broker: Trying to Create VM #14 in Datacenter_1
0.1: Broker: Trying to Create VM #15 in Datacenter_1
0.1: Broker: Trying to Create VM #16 in Datacenter_1
0.1: Broker: Trying to Create VM #17 in Datacenter_1
0.1: Broker: Trying to Create VM #18 in Datacenter_1
0.1: Broker: Trying to Create VM #19 in Datacenter_1
[VmScheduler.vmCreate] Allocation of VM #12 to Host #0 failed by RAM
[VmScheduler.vmCreate] Allocation of VM #12 to Host #1 failed by MIPS
[VmScheduler.vmCreate] Allocation of VM #13 to Host #0 failed by RAM
[VmScheduler.vmCreate] Allocation of VM #13 to Host #1 failed by MIPS
[VmScheduler.vmCreate] Allocation of VM #14 to Host #0 failed by RAM
[VmScheduler.vmCreate] Allocation of VM #14 to Host #1 failed by MIPS
[VmScheduler.vmCreate] Allocation of VM #15 to Host #0 failed by RAM
[VmScheduler.vmCreate] Allocation of VM #15 to Host #1 failed by MIPS
[VmScheduler.vmCreate] Allocation of VM #16 to Host #0 failed by RAM
[VmScheduler.vmCreate] Allocation of VM #16 to Host #1 failed by MIPS
[VmScheduler.vmCreate] Allocation of VM #17 to Host #0 failed by RAM
[VmScheduler.vmCreate] Allocation of VM #17 to Host #1 failed by MIPS
[VmScheduler.vmCreate] Allocation of VM #18 to Host #0 failed by RAM


eclipse-workspace - cloudsim/examples/org/cloudbus/cloudsim/examples/CloudSimExample6.java
File Edit Source Refactor Navigate Search Project Run Window Help
Problems Javadoc Declaration Console
<terminated> CloudSimExample6 [Java Application] D:\Downloads\CloudComputing\eclipse\plugins\
[VmScheduler.vmCreate] Allocation of VM #19 to Host #0 failed by RAM
[VmScheduler.vmCreate] Allocation of VM #19 to Host #1 failed by MIPS
0.2: Broker: VM #6 has been created in Datacenter #3, Host #0
0.2: Broker: VM #7 has been created in Datacenter #3, Host #0
0.2: Broker: VM #8 has been created in Datacenter #3, Host #0
0.2: Broker: VM #9 has been created in Datacenter #3, Host #1
0.2: Broker: VM #10 has been created in Datacenter #3, Host #0
0.2: Broker: VM #11 has been created in Datacenter #3, Host #1
0.2: Broker: Creation of VM #12 failed in Datacenter #3
0.2: Broker: Creation of VM #13 failed in Datacenter #3
0.2: Broker: Creation of VM #14 failed in Datacenter #3
0.2: Broker: Creation of VM #15 failed in Datacenter #3
0.2: Broker: Creation of VM #16 failed in Datacenter #3
0.2: Broker: Creation of VM #17 failed in Datacenter #3
0.2: Broker: Creation of VM #18 failed in Datacenter #3
0.2: Broker: Creation of VM #19 failed in Datacenter #3
0.2: Broker: Sending cloudlet 0 to VM #0
0.2: Broker: Sending cloudlet 1 to VM #1
0.2: Broker: Sending cloudlet 2 to VM #2
0.2: Broker: Sending cloudlet 3 to VM #3
0.2: Broker: Sending cloudlet 4 to VM #4
0.2: Broker: Sending cloudlet 5 to VM #5
0.2: Broker: Sending cloudlet 6 to VM #6
0.2: Broker: Sending cloudlet 7 to VM #7
0.2: Broker: Sending cloudlet 8 to VM #8
0.2: Broker: Sending cloudlet 9 to VM #9
0.2: Broker: Sending cloudlet 10 to VM #10
0.2: Broker: Sending cloudlet 11 to VM #11
0.2: Broker: Sending cloudlet 12 to VM #0
0.2: Broker: Sending cloudlet 13 to VM #1
0.2: Broker: Sending cloudlet 14 to VM #2
0.2: Broker: Sending cloudlet 15 to VM #3
0.2: Broker: Sending cloudlet 16 to VM #4
0.2: Broker: Sending cloudlet 17 to VM #5
0.2: Broker: Sending cloudlet 18 to VM #6


eclipse-workspace - cloudsim/examples/org/cloudbus/cloudsim/example
File Edit Source Refactor Navigate Search Project Run Window Help
Problems Javadoc Declaration Console
<terminated> CloudSimExample6 [Java Application] D:\Downloads\CloudC
0.2: Broker: Sending cloudlet 19 to VM #7
0.2: Broker: Sending cloudlet 20 to VM #8
0.2: Broker: Sending cloudlet 21 to VM #9
0.2: Broker: Sending cloudlet 22 to VM #10
0.2: Broker: Sending cloudlet 23 to VM #11
0.2: Broker: Sending cloudlet 24 to VM #0
0.2: Broker: Sending cloudlet 25 to VM #1
0.2: Broker: Sending cloudlet 26 to VM #2
0.2: Broker: Sending cloudlet 27 to VM #3
0.2: Broker: Sending cloudlet 28 to VM #4
0.2: Broker: Sending cloudlet 29 to VM #5
0.2: Broker: Sending cloudlet 30 to VM #6
0.2: Broker: Sending cloudlet 31 to VM #7
0.2: Broker: Sending cloudlet 32 to VM #8
0.2: Broker: Sending cloudlet 33 to VM #9
0.2: Broker: Sending cloudlet 34 to VM #10
0.2: Broker: Sending cloudlet 35 to VM #11
0.2: Broker: Sending cloudlet 36 to VM #0
0.2: Broker: Sending cloudlet 37 to VM #1
0.2: Broker: Sending cloudlet 38 to VM #2
0.2: Broker: Sending cloudlet 39 to VM #3
3.1980000000000004: Broker: Cloudlet 4 received
3.1980000000000004: Broker: Cloudlet 16 received
3.1980000000000004: Broker: Cloudlet 28 received
3.1980000000000004: Broker: Cloudlet 5 received
3.1980000000000004: Broker: Cloudlet 17 received
3.1980000000000004: Broker: Cloudlet 29 received
3.1980000000000004: Broker: Cloudlet 6 received
3.1980000000000004: Broker: Cloudlet 18 received
3.1980000000000004: Broker: Cloudlet 30 received
3.1980000000000004: Broker: Cloudlet 7 received
3.1980000000000004: Broker: Cloudlet 19 received
3.1980000000000004: Broker: Cloudlet 31 received
3.1980000000000004: Broker: Cloudlet 8 received
3.1980000000000004: Broker: Cloudlet 20 received


eclipse-workspace - cloudsim/examples/org/cloudbus/cloudsim/examples/CloudSi
File Edit Source Refactor Navigate Search Project Run Window Help
Problems Javadoc Declaration Console
<terminated> CloudSimExample6 [Java Application] D:\Downloads\CloudComputin
3.1980000000000004: Broker: Cloudlet 32 received
3.1980000000000004: Broker: Cloudlet 10 received
3.1980000000000004: Broker: Cloudlet 22 received
3.1980000000000004: Broker: Cloudlet 34 received
3.1980000000000004: Broker: Cloudlet 9 received
3.1980000000000004: Broker: Cloudlet 21 received
3.1980000000000004: Broker: Cloudlet 33 received
3.1980000000000004: Broker: Cloudlet 11 received
3.1980000000000004: Broker: Cloudlet 23 received
3.1980000000000004: Broker: Cloudlet 35 received
4.198: Broker: Cloudlet 0 received
4.198: Broker: Cloudlet 12 received
4.198: Broker: Cloudlet 24 received
4.198: Broker: Cloudlet 36 received
4.198: Broker: Cloudlet 1 received
4.198: Broker: Cloudlet 13 received
4.198: Broker: Cloudlet 25 received
4.198: Broker: Cloudlet 37 received
4.198: Broker: Cloudlet 2 received
4.198: Broker: Cloudlet 14 received
4.198: Broker: Cloudlet 26 received
4.198: Broker: Cloudlet 38 received
4.198: Broker: Cloudlet 3 received
4.198: Broker: Cloudlet 15 received
4.198: Broker: Cloudlet 27 received
4.198: Broker: Cloudlet 39 received
4.198: Broker: All Cloudlets executed. Finishing...
4.198: Broker: Destroying VM #0
4.198: Broker: Destroying VM #1
4.198: Broker: Destroying VM #2
4.198: Broker: Destroying VM #3
4.198: Broker: Destroying VM #4
4.198: Broker: Destroying VM #5
4.198: Broker: Destroying VM #6
4.198: Broker: Destroying VM #7


eclipse-workspace - cloudsim/examples/org/cloudbus/cloudsim/examples/CloudSimExample6.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Problems Javadoc Declaration Console
<terminated> CloudSimExample6 [Java Application] D:\Downloads\CloudComputing\eclipse\plugins\org.eclipse.justj.openjdk.
4.198: Broker: Destroying VM #8
4.198: Broker: Destroying VM #9
4.198: Broker: Destroying VM #10
4.198: Broker: Destroying VM #11
Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Datacenter_1 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.
========== OUTPUT ==========
Cloudlet ID STATUS Data center ID VM ID Time Start Time Finish Time
4 SUCCESS 2 4 3 0.2 3.2
16 SUCCESS 2 4 3 0.2 3.2
28 SUCCESS 2 4 3 0.2 3.2
5 SUCCESS 2 5 3 0.2 3.2
17 SUCCESS 2 5 3 0.2 3.2
29 SUCCESS 2 5 3 0.2 3.2
6 SUCCESS 3 6 3 0.2 3.2
18 SUCCESS 3 6 3 0.2 3.2
30 SUCCESS 3 6 3 0.2 3.2
7 SUCCESS 3 7 3 0.2 3.2
19 SUCCESS 3 7 3 0.2 3.2
31 SUCCESS 3 7 3 0.2 3.2
8 SUCCESS 3 8 3 0.2 3.2
20 SUCCESS 3 8 3 0.2 3.2
32 SUCCESS 3 8 3 0.2 3.2
10 SUCCESS 3 10 3 0.2 3.2
22 SUCCESS 3 10 3 0.2 3.2
34 SUCCESS 3 10 3 0.2 3.2
9 SUCCESS 3 9 3 0.2 3.2
21 SUCCESS 3 9 3 0.2 3.2

Experiment – 8

Aim - Program to pause and resume the simulation, and create simulation entities (a DatacenterBroker in this example) dynamically.

Software Used – Eclipse

Package Used – Cloudsim(Framework)

Code –

```
* Title: CloudSim Toolkit
 * Description: CloudSim (Cloud Simulation) Toolkit for Modeling and Simulation * of Clouds
 * Licence: GPL - http://www.gnu.org/copyleft/gpl.html
 *
 * Copyright (c) 2009, The University of Melbourne, Australia */
package org.cloudbus.cloudsim.examples;
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; /**
 * An example showing how to pause and resume the simulation,
 * and create simulation entities (a DatacenterBroker in this example) * dynamically.
 */
public class CloudSimExample7 {
 /** The cloudlet list. */
 private static List<Cloudlet> cloudletList;
 /** The vmlist. */
```

```
 private static List<Vm> vmlist;
 private static List<Vm> createVM(int userId, int vms, int idShift) {   //Creates a container to store
VMs. This list is passed to the broker  later
 LinkedList<Vm> list = new LinkedList<Vm>();
 //VM Parameters
 long size = 10000; //image size (MB)
 int ram = 512; //vm memory (MB)
 int mips = 250;
 long bw = 1000;
 int pesNumber = 1; //number of cpus
 String vmm = "Xen"; //VMM name
 //create VMs
 Vm[] vm = new Vm[vms];
 for(int i=0;i<vms;i++){
 vm[i] = new Vm(idShift + i, userId, mips, pesNumber, ram, bw, size,  vmm, new
CloudletSchedulerTimeShared());
 list.add(vm[i]);
 }
 return list;
 }
 private static List<Cloudlet> createCloudlet(int userId, int cloudlets, int idShift){
 // Creates a container to store Cloudlets
 LinkedList<Cloudlet> list = new LinkedList<Cloudlet>();   //cloudlet parameters
 long length = 40000;
 long fileSize = 300;
 long outputSize = 300;
 int pesNumber = 1;
 UtilizationModel utilizationModel = new UtilizationModelFull();   Cloudlet[] cloudlet = new
Cloudlet[cloudlets];
 for(int i=0;i<cloudlets;i++){
 cloudlet[i] = new Cloudlet(idShift + i, length, pesNumber, fileSize,  outputSize, utilizationModel,
utilizationModel, utilizationModel);   // setting the owner of these Cloudlets
 cloudlet[i].setUserId(userId);
 list.add(cloudlet[i]);
 }
 return list;
 }
 /////////////////////////// STATIC METHODS ///////////////////////  /**
 * Creates main() to run this example
 */
 public static void main(String[] args) {
 Log.printLine("Starting CloudSimExample7...");
 try {
 // First step: Initialize the CloudSim package. It should be called  // before creating any entities.
 int num_user = 2; // number of grid users
 Calendar calendar = Calendar.getInstance();
 boolean trace_flag = false; // mean trace events
 // Initialize the CloudSim library
```

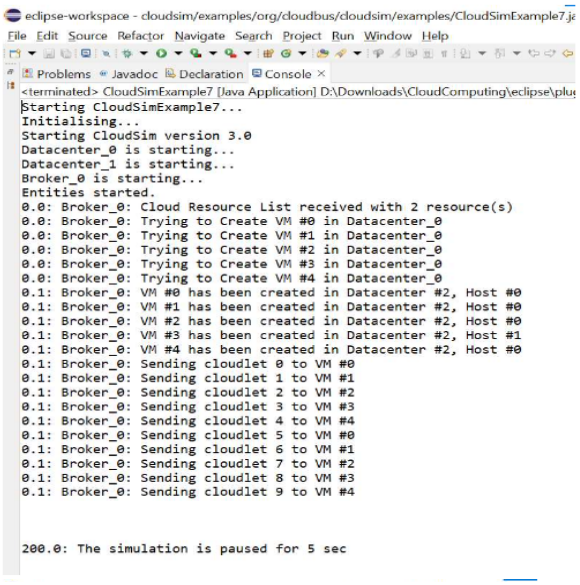
```
CloudSim.init(num_user, calendar, trace_flag);
// Second step: Create Datacenters
//Datacenters are the resource providers in CloudSim. We need at list  one of them to run a
CloudSim simulation
@SuppressWarnings("unused")
Datacenter datacenter0 = createDatacenter("Datacenter_0");
@SuppressWarnings("unused")
Datacenter datacenter1 = createDatacenter("Datacenter_1");   //Third step: Create
Broker
DatacenterBroker broker = createBroker("Broker_0");   int brokerId =
broker.getId();
//Fourth step: Create VMs and Cloudlets and send them to broker  vmlist =
createVM(brokerId, 5, 0); //creating 5 vms  cloudletList = createCloudlet(brokerId, 10, 0); //
creating 10  cloudlets
broker.submitVmList(vmlist);
broker.submitCloudletList(cloudletList);
// A thread that will create a new broker at 200 clock time  Runnable monitor = new
Runnable() {
@Override
public void run() {
CloudSim.pauseSimulation(200);
while (true) {
if (CloudSim.isPaused()) {
break;
}
try {
Thread.sleep(100);
} catch (InterruptedException e) {
e.printStackTrace();
}
}
Log.printLine("\n\n\n" + CloudSim.clock() + ": The simulation  is paused for 5 sec \n\n");
try {
Thread.sleep(5000);
} catch (InterruptedException e) {
e.printStackTrace();
}
DatacenterBroker broker = createBroker("Broker_1");   int brokerId = broker.getId();
//Create VMs and Cloudlets and send them to broker  vmlist = createVM(brokerId, 5, 100); //creating
5 vms  cloudletList = createCloudlet(brokerId, 10, 100); // creating  10 cloudlets
broker.submitVmList(vmlist);
broker.submitCloudletList(cloudletList);
CloudSim.resumeSimulation();
}
};
new Thread(monitor).start();
Thread.sleep(1000);
// Fifth step: Starts the simulation
```

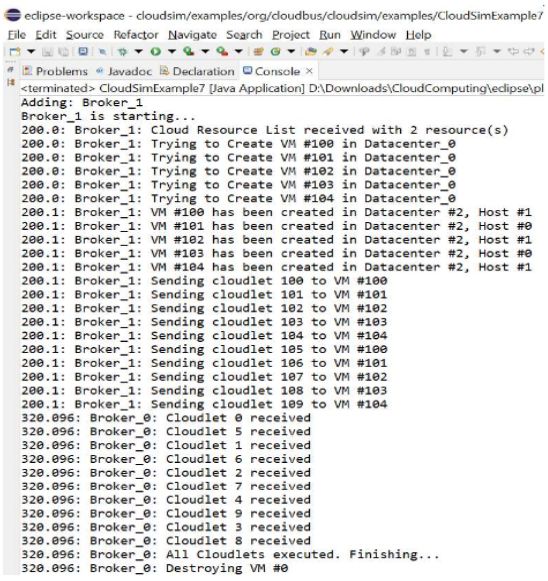
```
CloudSim.startSimulation();
// Final step: Print results when simulation is over  List<Cloudlet> newList =
broker.getCloudletReceivedList();   CloudSim.stopSimulation();
printCloudletList(newList);
Log.printLine("CloudSimExample7 finished!");
}
catch (Exception e)
{
e.printStackTrace();
Log.printLine("The simulation has been terminated due to an  unexpected error");
}
}
private static Datacenter createDatacenter(String name){
// Here are the steps needed to create a PowerDatacenter:  // 1. We need to
create a list to store one or more
// Machines
List<Host> hostList = new ArrayList<Host>();
// 2. A Machine contains one or more PEs or CPUs/Cores. Therefore, should  // create a list to store
these PEs before creating
// a Machine.
List<Pe> peList1 = new ArrayList<Pe>();
int mips = 1000;
// 3. Create PEs and add these into the list.
//for a quad-core machine, a list of 4 PEs is required:  peList1.add(new Pe(0, new
PeProvisionerSimple(mips))); // need to store  Pe id and MIPS Rating
 peList1.add(new Pe(1, new PeProvisionerSimple(mips)));    peList1.add(new
Pe(2, new PeProvisionerSimple(mips)));         peList1.add(new Pe(3, new
PeProvisionerSimple(mips)));   //Another list, for a dual-core machine
List<Pe> peList2 = new ArrayList<Pe>();
peList2.add(new Pe(0, new PeProvisionerSimple(mips)));   peList2.add(new Pe(1, new
PeProvisionerSimple(mips)));   //4. Create Hosts with its id and list of PEs and add them to the list of
machines
int hostId=0;
int ram = 16384; //host memory (MB)
long storage = 1000000; //host storage
int bw = 10000;
hostList.add(
new Host(
hostId,
new RamProvisionerSimple(ram),
new BwProvisionerSimple(bw),
storage,
peList1,
new VmSchedulerTimeShared(peList1)
)
); // This is our first machine
hostId++;
hostList.add(
```

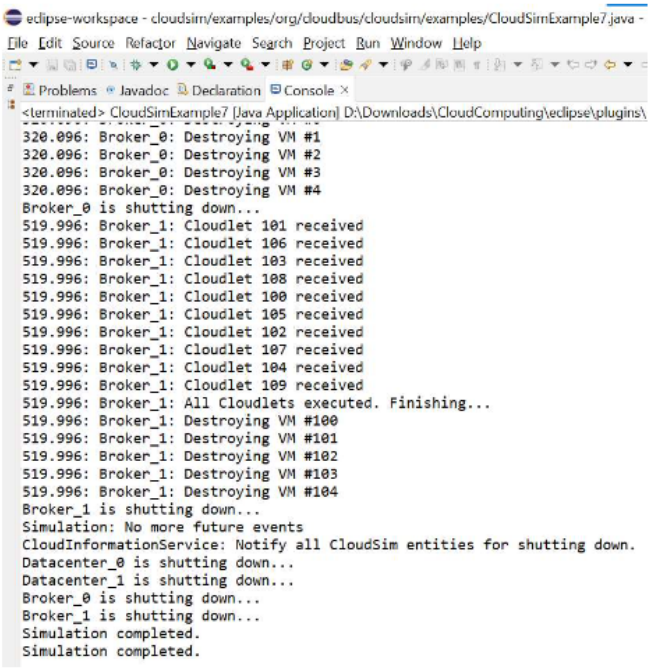
```
 new Host(
 hostId,
 new RamProvisionerSimple(ram),
 new BwProvisionerSimple(bw),
 storage,
 peList2,
 new VmSchedulerTimeShared(peList2)
 )
 ); // Second machine
 // 5. Create a DatacenterCharacteristics object that stores the  // properties of a data center:
architecture, OS, list of  // Machines, allocation policy: time- or space-shared, time zone  //
and its price (G$/Pe time unit).
 String arch = "x86"; // system architecture
 String os = "Linux"; // operating system
 String vmm = "Xen";
 double time_zone = 10.0; // time zone this resource located  double cost = 3.0; // the cost of using
processing in this  resource
 double costPerMem = 0.05; // the cost of using memory in this  resource
 double costPerStorage = 0.1; // the cost of using storage in this  resource
 double costPerBw = 0.1; // the cost of using bw in this resource  LinkedList<Storage> storageList =
new LinkedList<Storage>(); //we are  not adding SAN devices by now
 DatacenterCharacteristics characteristics = new
DatacenterCharacteristics(
 arch, os, vmm, hostList, time_zone, cost, costPerMem,  costPerStorage, costPerBw);
 // 6. Finally, we need to create a PowerDatacenter object.  Datacenter datacenter
= null;
 try {
 datacenter = new Datacenter(name, characteristics, new
VmAllocationPolicySimple(hostList), storageList, 0);
 } catch (Exception e) {
 e.printStackTrace();
 }
 return datacenter;
 }
 //We strongly encourage users to develop their own broker policies, to submit  vms and cloudlets
according
 //to the specific rules of the simulated scenario
 private static DatacenterBroker createBroker(String name){
DatacenterBroker broker = null;
 try {
 broker = new DatacenterBroker(name);
 } catch (Exception e) {
 e.printStackTrace();
 return null;
 }
 return broker;
 }
 /**
```

```
 * Prints the Cloudlet objects
 * @param list list of Cloudlets
 */
 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 + 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 + indent +
dft.format(cloudlet.getActualCPUTime()) +
 indent + indent +
dft.format(cloudlet.getExecStartTime())+ indent + indent + indent +
dft.format(cloudlet.getFinishTime()));
 }
 }
 }
}
```

Output –








```
========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID        Time    Start Time    Finish Time
    0         SUCCESS        2             0           320        0.1           320.1
    5         SUCCESS        2             0           320        0.1           320.1
    1         SUCCESS        2             1           320        0.1           320.1
    6         SUCCESS        2             1           320        0.1           320.1
    2         SUCCESS        2             2           320        0.1           320.1
    7         SUCCESS        2             2           320        0.1           320.1
    4         SUCCESS        2             4           320        0.1           320.1
    9         SUCCESS        2             4           320        0.1           320.1
    3         SUCCESS        2             3           320        0.1           320.1
    8         SUCCESS        2             3           320        0.1           320.1
CloudSimExample7 finished!
```

Experiment – 9

Aim - Program to create simulation entities (a DatacenterBroker in this example) in run-time using a global manager entity (GlobalBroker).entities (a DatacenterBroker in this example) dynamically.

Software Used – Eclipse

Package Used – Cloudsim(Framework)

Code –

```
* Title: CloudSim Toolkit
 * Description: CloudSim (Cloud Simulation) Toolkit for Modeling and Simulation * of Clouds
 * Licence: GPL - http://www.gnu.org/copyleft/gpl.html
 *
 * Copyright (c) 2009, The University of Melbourne, Australia */
package org.cloudbus.cloudsim.examples;
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.core.SimEntity;
import org.cloudbus.cloudsim.core.SimEvent;
import org.cloudbus.cloudsim.provisioners.BwProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.PeProvisionerSimple; import
org.cloudbus.cloudsim.provisioners.RamProvisionerSimple; /**
 * An example showing how to create simulation entities
 * (a DatacenterBroker in this example) in run-time using
 * a globar manager entity (GlobalBroker).
 */
```

```
public class CloudSimExample8 {
 /** The cloudlet list. */
 private static List<Cloudlet> cloudletList;
 /** The vmList. */
 private static List<Vm> vmList;
 private static List<Vm> createVM(int userId, int vms, int idShift) {   //Creates a container to store
VMs. This list is passed to the broker  later
 LinkedList<Vm> list = new LinkedList<Vm>();
 //VM Parameters
 long size = 10000; //image size (MB)
 int ram = 512; //vm memory (MB)
 int mips = 250;
 long bw = 1000;
 int pesNumber = 1; //number of cpus
 String vmm = "Xen"; //VMM name
 //create VMs
 Vm[] vm = new Vm[vms];
 for(int i=0;i<vms;i++){
 vm[i] = new Vm(idShift + i, userId, mips, pesNumber, ram, bw, size,  vmm, new
CloudletSchedulerTimeShared());
 list.add(vm[i]);
 }
 return list;
 }
 private static List<Cloudlet> createCloudlet(int userId, int cloudlets, int idShift){
 // Creates a container to store Cloudlets
 LinkedList<Cloudlet> list = new LinkedList<Cloudlet>();   //cloudlet parameters
 long length = 40000;
 long fileSize = 300;
 long outputSize = 300;
 int pesNumber = 1;
 UtilizationModel utilizationModel = new UtilizationModelFull();   Cloudlet[] cloudlet = new
Cloudlet[cloudlets];
 for(int i=0;i<cloudlets;i++){
 cloudlet[i] = new Cloudlet(idShift + i, length, pesNumber, fileSize,  outputSize, utilizationModel,
utilizationModel, utilizationModel);   // setting the owner of these Cloudlets
 cloudlet[i].setUserId(userId);
 list.add(cloudlet[i]);
 }
 return list;
 }
 ////////////////////////// STATIC METHODS ////////////////////// /**
 * Creates main() to run this example
 */
 public static void main(String[] args) {
 Log.printLine("Starting CloudSimExample8...");
 try {
 // First step: Initialize the CloudSim package. It should be called  // before creating any entities.
```

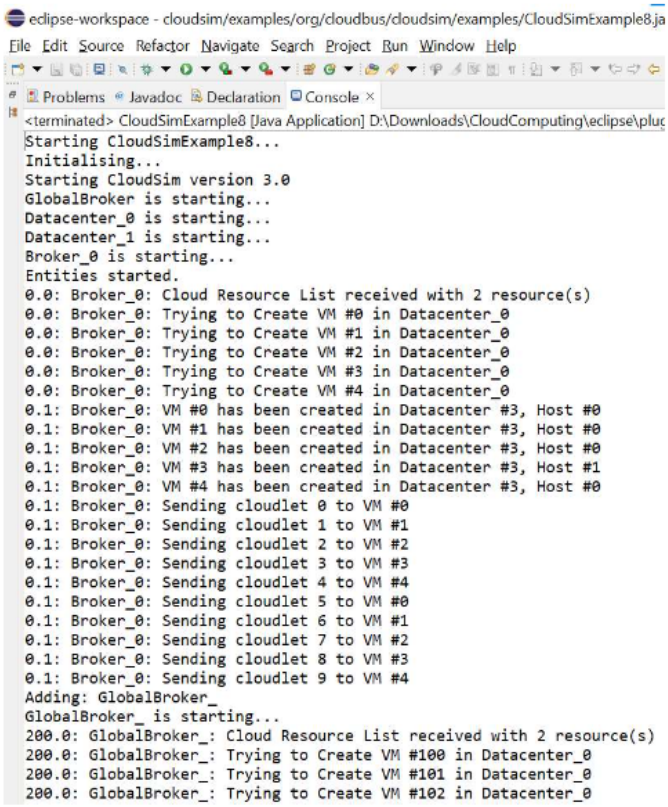
```
 int num_user = 2; // number of grid users
 Calendar calendar = Calendar.getInstance();
 boolean trace_flag = false; // mean trace events
 // Initialize the CloudSim library
 CloudSim.init(num_user, calendar, trace_flag);
 GlobalBroker globalBroker = new GlobalBroker("GlobalBroker");   // Second step: Create
Datacenters
 //Datacenters are the resource providers in CloudSim. We need at list  one of them to run a
CloudSim simulation
 @SuppressWarnings("unused")
 Datacenter datacenter0 = createDatacenter("Datacenter_0");
@SuppressWarnings("unused")
 Datacenter datacenter1 = createDatacenter("Datacenter_1");   //Third step: Create
Broker
 DatacenterBroker broker = createBroker("Broker_0");   int brokerId =
broker.getId();
 //Fourth step: Create VMs and Cloudlets and send them to broker  vmList =
createVM(brokerId, 5, 0); //creating 5 vms  cloudletList = createCloudlet(brokerId, 10, 0); //
creating 10  cloudlets
 broker.submitVmList(vmList);
 broker.submitCloudletList(cloudletList);
 // Fifth step: Starts the simulation
 CloudSim.startSimulation();
 // Final step: Print results when simulation is over  List<Cloudlet> newList =
broker.getCloudletReceivedList();
newList.addAll(globalBroker.getBroker().getCloudletReceivedList());   CloudSim.stopSimulation();
 printCloudletList(newList);
 Log.printLine("CloudSimExample8 finished!");
 }
 catch (Exception e)
 {
 e.printStackTrace();
 Log.printLine("The simulation has been terminated due to an  unexpected error");
 }
 }
 private static Datacenter createDatacenter(String name){
 // Here are the steps needed to create a PowerDatacenter:  // 1. We need to
create a list to store one or more
 // Machines
 List<Host> hostList = new ArrayList<Host>();
 // 2. A Machine contains one or more PEs or CPUs/Cores. Therefore, should  // create a list to store
these PEs before creating  // a Machine.
 List<Pe> peList1 = new ArrayList<Pe>();
 int mips = 1000;
 // 3. Create PEs and add these into the list.
 //for a quad-core machine, a list of 4 PEs is required:  peList1.add(new Pe(0, new
PeProvisionerSimple(mips))); // need to store  Pe id and MIPS Rating
  peList1.add(new Pe(1, new PeProvisionerSimple(mips)));    peList1.add(new
Pe(2,   new   PeProvisionerSimple(mips)));         peList1.add(new   Pe(3,   new
```

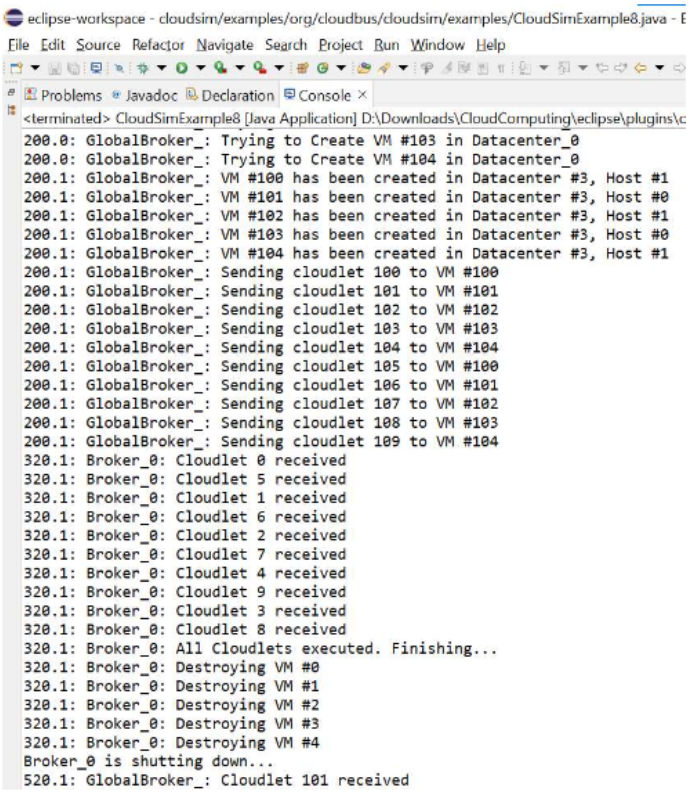
```
PeProvisionerSimple(mips)));   //Another list, for a dual-core machine
 List<Pe> peList2 = new ArrayList<Pe>();
 peList2.add(new Pe(0, new PeProvisionerSimple(mips)));   peList2.add(new Pe(1, new
PeProvisionerSimple(mips)));   //4. Create Hosts with its id and list of PEs and add them to the list of
machines
 int hostId=0;
 int ram = 16384; //host memory (MB)
 long storage = 1000000; //host storage
 int bw = 10000;
 hostList.add(
 new Host(
 hostId,
 new RamProvisionerSimple(ram),
 new BwProvisionerSimple(bw),
 storage,
 peList1,
 new VmSchedulerTimeShared(peList1)
 )
 ); // This is our first machine
 hostId++;
 hostList.add(
 new Host(
 hostId,
 new RamProvisionerSimple(ram),
 new BwProvisionerSimple(bw),
 storage,
 peList2,
 new VmSchedulerTimeShared(peList2)
 )
 ); // Second machine
 // 5. Create a DatacenterCharacteristics object that stores the  // properties of a data center:
architecture, OS, list of  // Machines, allocation policy: time- or space-shared, time zone  //
and its price (G$/Pe time unit).
 String arch = "x86"; // system architecture
 String os = "Linux"; // operating system
 String vmm = "Xen";
 double time_zone = 10.0; // time zone this resource located  double cost = 3.0; // the cost of using
processing in this  resource
 double costPerMem = 0.05; // the cost of using memory in this  resource
 double costPerStorage = 0.1; // the cost of using storage in this  resource
 double costPerBw = 0.1; // the cost of using bw in this resource  LinkedList<Storage> storageList =
new LinkedList<Storage>(); //we are  not adding SAN devices by now
 DatacenterCharacteristics characteristics = new
DatacenterCharacteristics(
 arch, os, vmm, hostList, time_zone, cost, costPerMem,  costPerStorage, costPerBw);
 // 6. Finally, we need to create a PowerDatacenter object.  Datacenter datacenter
= null;
 try {
 datacenter = new Datacenter(name, characteristics, new
```

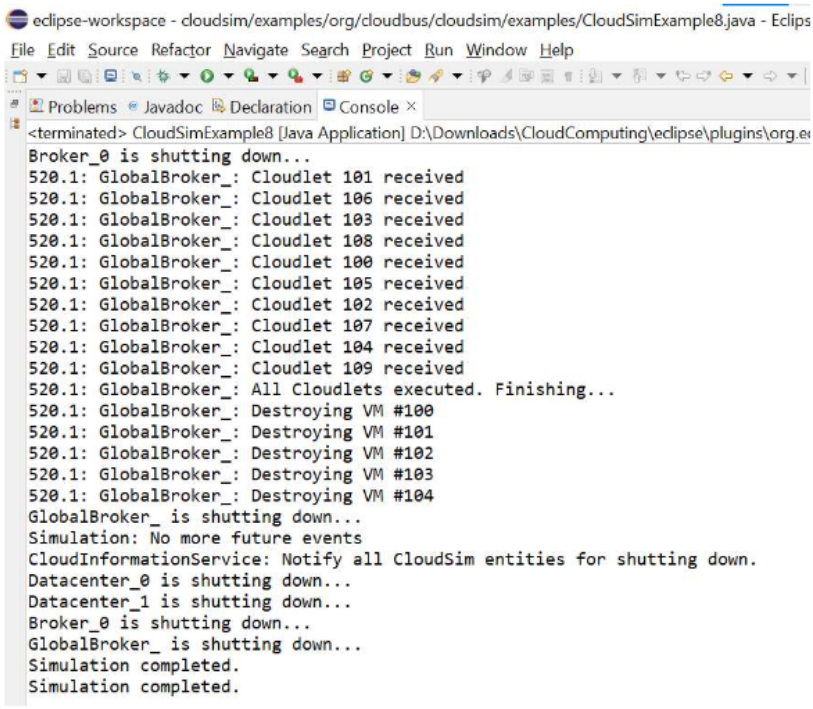
```
VmAllocationPolicySimple(hostList), storageList, 0);
 } catch (Exception e) {
 e.printStackTrace();
 }
 return datacenter;
 }
 //We strongly encourage users to develop their own broker policies, to submit  vms and cloudlets
according
 //to the specific rules of the simulated scenario
 private static DatacenterBroker createBroker(String name){
DatacenterBroker broker = null;
 try {
 broker = new DatacenterBroker(name);
 } catch (Exception e) {
 e.printStackTrace();
 return null;
 }
 return broker;
 }
 /**
 * Prints the Cloudlet objects
 * @param list list of Cloudlets
 */
 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 + 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 + indent +
dft.format(cloudlet.getActualCPUTime()) +
 indent + indent +
dft.format(cloudlet.getExecStartTime())+ indent + indent + indent +
dft.format(cloudlet.getFinishTime()));
 }
 }
 }
 public static class GlobalBroker extends SimEntity {
 private static final int CREATE_BROKER = 0;
 private List<Vm> vmList;
 private List<Cloudlet> cloudletList;
```

```
 private DatacenterBroker broker;
 public GlobalBroker(String name) {
 super(name);
 }
 @Override
 public void processEvent(SimEvent ev) {
 switch (ev.getTag()) {
 case CREATE_BROKER:
 setBroker(createBroker(super.getName()+"_"));   //Create VMs and Cloudlets and send them to
broker  setVmList(createVM(getBroker().getId(), 5, 100)); //creating 5  vms
 setCloudletList(createCloudlet(getBroker().getId(), 10, 100)); //  creating 10 cloudlets
 broker.submitVmList(getVmList());
 broker.submitCloudletList(getCloudletList());   CloudSim.resumeSimulation();
 break;
 default:
 Log.printLine(getName() + ": unknown event type");   break;
 }
 }
 @Override
 public void startEntity() {
 Log.printLine(super.getName()+" is starting...");   schedule(getId(), 200,
CREATE_BROKER);
 }
 @Override
 public void shutdownEntity() {
 }
 public List<Vm> getVmList() {
 return vmList;
 }
 protected void setVmList(List<Vm> vmList) {
 this.vmList = vmList;
 }
 public List<Cloudlet> getCloudletList() {
 return cloudletList;
 }
 protected void setCloudletList(List<Cloudlet> cloudletList) {   this.cloudletList =
cloudletList;
 }
 public DatacenterBroker getBroker() {
 return broker;
 }
 protected void setBroker(DatacenterBroker broker) {   this.broker =
broker;
 }
 }
 }
```

Output –


eclipse-workspace - cloudsim/examples/org/cloudbus/cloudsim/examples/CloudSimExample8.ja
File Edit Source Refactor Navigate Search Project Run Window Help
Problems Javadoc Declaration Console
<terminated> CloudSimExample8 [Java Application] D:\Downloads\CloudComputing\eclipse\plug
Starting CloudSimExample8...
Initialising...
Starting CloudSim version 3.0
GlobalBroker is starting...
Datacenter_0 is starting...
Datacenter_1 is starting...
Broker_0 is starting...
Entities started.
0.0: Broker_0: Cloud Resource List received with 2 resource(s)
0.0: Broker_0: Trying to Create VM #0 in Datacenter_0
0.0: Broker_0: Trying to Create VM #1 in Datacenter_0
0.0: Broker_0: Trying to Create VM #2 in Datacenter_0
0.0: Broker_0: Trying to Create VM #3 in Datacenter_0
0.0: Broker_0: Trying to Create VM #4 in Datacenter_0
0.1: Broker_0: VM #0 has been created in Datacenter #3, Host #0
0.1: Broker_0: VM #1 has been created in Datacenter #3, Host #0
0.1: Broker_0: VM #2 has been created in Datacenter #3, Host #0
0.1: Broker_0: VM #3 has been created in Datacenter #3, Host #1
0.1: Broker_0: VM #4 has been created in Datacenter #3, Host #0
0.1: Broker_0: Sending cloudlet 0 to VM #0
0.1: Broker_0: Sending cloudlet 1 to VM #1
0.1: Broker_0: Sending cloudlet 2 to VM #2
0.1: Broker_0: Sending cloudlet 3 to VM #3
0.1: Broker_0: Sending cloudlet 4 to VM #4
0.1: Broker_0: Sending cloudlet 5 to VM #0
0.1: Broker_0: Sending cloudlet 6 to VM #1
0.1: Broker_0: Sending cloudlet 7 to VM #2
0.1: Broker_0: Sending cloudlet 8 to VM #3
0.1: Broker_0: Sending cloudlet 9 to VM #4
Adding: GlobalBroker_
GlobalBroker_ is starting...
200.0: GlobalBroker_: Cloud Resource List received with 2 resource(s)
200.0: GlobalBroker_: Trying to Create VM #100 in Datacenter_0
200.0: GlobalBroker_: Trying to Create VM #101 in Datacenter_0
200.0: GlobalBroker_: Trying to Create VM #102 in Datacenter_0


eclipse-workspace - cloudsim/examples/org/cloudbus/cloudsim/examples/CloudSimExample8.java - E
File Edit Source Refactor Navigate Search Project Run Window Help
Problems Javadoc Declaration Console
<terminated> CloudSimExample8 [Java Application] D:\Downloads\CloudComputing\eclipse\plugins\c
200.0: GlobalBroker_: Trying to Create VM #103 in Datacenter_0
200.0: GlobalBroker_: Trying to Create VM #104 in Datacenter_0
200.1: GlobalBroker_: VM #100 has been created in Datacenter #3, Host #1
200.1: GlobalBroker_: VM #101 has been created in Datacenter #3, Host #0
200.1: GlobalBroker_: VM #102 has been created in Datacenter #3, Host #1
200.1: GlobalBroker_: VM #103 has been created in Datacenter #3, Host #0
200.1: GlobalBroker_: VM #104 has been created in Datacenter #3, Host #1
200.1: GlobalBroker_: Sending cloudlet 100 to VM #100
200.1: GlobalBroker_: Sending cloudlet 101 to VM #101
200.1: GlobalBroker_: Sending cloudlet 102 to VM #102
200.1: GlobalBroker_: Sending cloudlet 103 to VM #103
200.1: GlobalBroker_: Sending cloudlet 104 to VM #104
200.1: GlobalBroker_: Sending cloudlet 105 to VM #100
200.1: GlobalBroker_: Sending cloudlet 106 to VM #101
200.1: GlobalBroker_: Sending cloudlet 107 to VM #102
200.1: GlobalBroker_: Sending cloudlet 108 to VM #103
200.1: GlobalBroker_: Sending cloudlet 109 to VM #104
320.1: Broker_0: Cloudlet 0 received
320.1: Broker_0: Cloudlet 5 received
320.1: Broker_0: Cloudlet 1 received
320.1: Broker_0: Cloudlet 6 received
320.1: Broker_0: Cloudlet 2 received
320.1: Broker_0: Cloudlet 7 received
320.1: Broker_0: Cloudlet 4 received
320.1: Broker_0: Cloudlet 9 received
320.1: Broker_0: Cloudlet 3 received
320.1: Broker_0: Cloudlet 8 received
320.1: Broker_0: All Cloudlets executed. Finishing...
320.1: Broker_0: Destroying VM #0
320.1: Broker_0: Destroying VM #1
320.1: Broker_0: Destroying VM #2
320.1: Broker_0: Destroying VM #3
320.1: Broker_0: Destroying VM #4
Broker_0 is shutting down...
520.1: GlobalBroker_: Cloudlet 101 received




```
========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID        Time    Start Time    Finish Time
    0        SUCCESS        3              0            320        0.1            320.1
    5        SUCCESS        3              0            320        0.1            320.1
    1        SUCCESS        3              1            320        0.1            320.1
    6        SUCCESS        3              1            320        0.1            320.1
    2        SUCCESS        3              2            320        0.1            320.1
    7        SUCCESS        3              2            320        0.1            320.1
    4        SUCCESS        3              4            320        0.1            320.1
    9        SUCCESS        3              4            320        0.1            320.1
    3        SUCCESS        3              3            320        0.1            320.1
    8        SUCCESS        3              3            320        0.1            320.1
    101        SUCCESS        3              101            320        200.1            520.1
    106        SUCCESS        3              101            320        200.1            520.1
    103        SUCCESS        3              103            320        200.1            520.1
    108        SUCCESS        3              103            320        200.1            520.1
    100        SUCCESS        3              100            320        200.1            520.1
    105        SUCCESS        3              100            320        200.1            520.1
    102        SUCCESS        3              102            320        200.1            520.1
    107        SUCCESS        3              102            320        200.1            520.1
    104        SUCCESS        3              104            320        200.1            520.1
    109        SUCCESS        3              104            320        200.1            520.1
CloudSimExample8 finished!
```

Experiment – 10

Aim - Execute a Python Program for the following in Google-Colab Platform: a. Multiplication of 3 matrices of size 2 X 2.

b. Analyse boston house prices dataset using Matplotlib/Seaborn Library

Software Used – Google Colab

Language Used – Python

a. Multiplication of 3 matrices of size 2 X 2.

Libraries & Global Declarations

```
[1]  import numpy as np

     dim_row = 2
     dim_col = 2
```

Multiplication Function (for 2 Matrices)

```
[2]  def multiply_two(m1, m2):
       row = m1.shape[0]
       col = m2.shape[1]
       res = np.zeros(shape=(row, col))
       for i in range(row):
         for j in range(col):
           sum = 0
           for (l,m) in zip(m1[i, :], m2[:, j]):
             sum += l*m
           res[i][j] = sum
       return res
```

Input

```
[3]  mat1 = np.zeros(shape=(dim_row, dim_col))
     mat2 = np.zeros(shape=(dim_row, dim_col))
     mat3 = np.zeros(shape=(dim_row, dim_col))
     matrices = [mat1, mat2, mat3]

     for i in range(0, 3):
       print(f"Enter values for matrix {i+1}")
       for j in range(0, dim_row * dim_col):
         row = int(j/dim_row)
         col = j%dim_col
         val = float(input(f"val {row + 1}, {col + 1}: "))
         matrices[i][row][col] = val
```

```
Enter values for matrix 1
val 1, 1: 1
val 1, 2: 2
val 2, 1: 3
val 2, 2: 4
Enter values for matrix 2
val 1, 1: 5
val 1, 2: 6
val 2, 1: 7
val 2, 2: 8
Enter values for matrix 3
val 1, 1: 9
val 1, 2: 10
val 2, 1: 11
val 2, 2: 12
```

Result

```
res = multiply_two(mat1, multiply_two(mat2, mat3))
print(res)
```

```
[[ 413.  454.]
 [ 937. 1030.]]
```

b. Analyse boston house prices dataset using Matplotlib/Seaborn Library

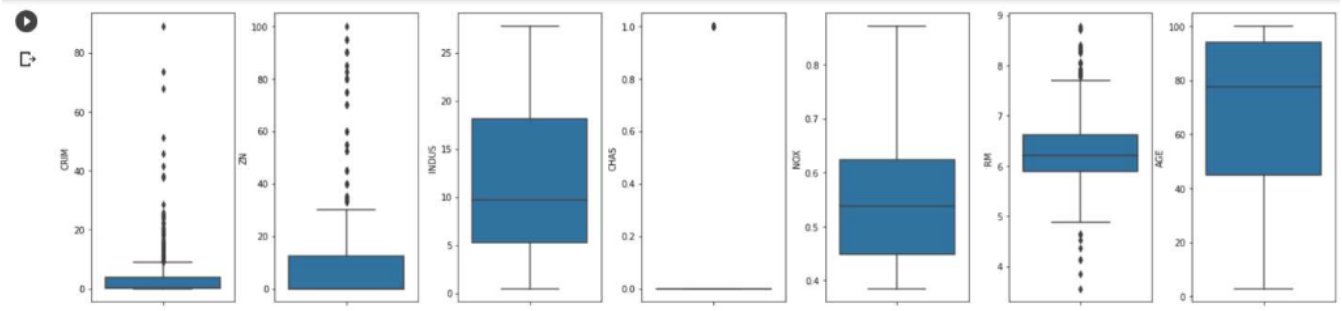
Importing Libraries & Dataset

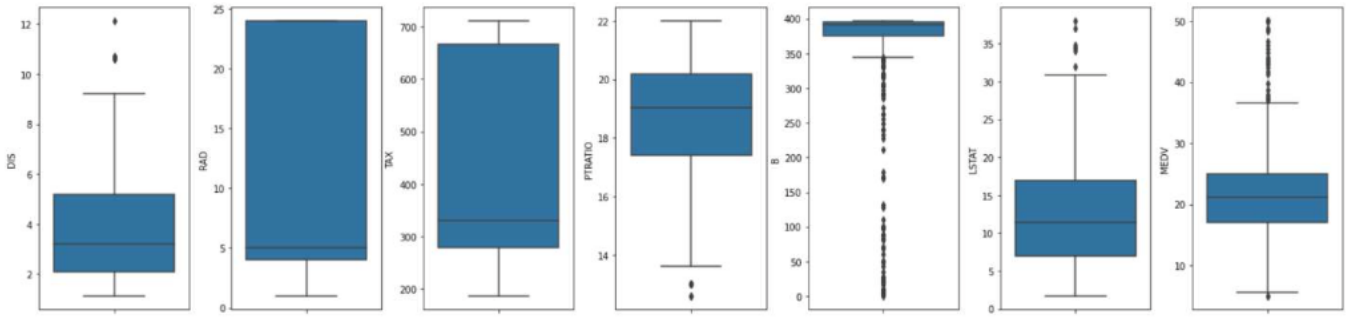
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats

column_names = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX', 'PTRATIO', 'B', 'LSTAT', 'MEDV']
df = pd.read_csv('housing.csv', header=None, delimiter=r"\s+", names=column_names)

df.head()
df.describe()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	L
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.00
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043	9.549407	408.237154	18.455534	356.674032	12.65
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710	8.707259	168.537116	2.164946	91.294864	7.14
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000	12.600000	0.320000	1.73
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4.000000	279.000000	17.400000	375.377500	6.95
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5.000000	330.000000	19.050000	391.440000	11.36
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24.000000	666.000000	20.200000	396.225000	16.95
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	22.000000	396.900000	37.97

Box Plot (Outliers)

```
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats

fig, axs = plt.subplots(ncols=7, nrows=2, figsize=(20, 10))
index = 0
axs = axs.flatten()
for k,v in df.items():
    sns.boxplot(y=k, data=df, ax=axs[index])
    index += 1
plt.tight_layout(pad=0.4, w_pad=0.5, h_pad=5.0)
```





MEDV (Median value of owner-occupied homes) vs Parameters Scatter Plot

```
from sklearn import preprocessing

min_max_scaler = preprocessing.MinMaxScaler()
column_sels = ['LSTAT', 'INDUS', 'NOX', 'PTRATIO', 'RM', 'TAX', 'DIS', 'AGE']
x = df.loc[:,column_sels]
y = df['MEDV']
x = pd.DataFrame(data=min_max_scaler.fit_transform(x), columns=column_sels)
fig, axs = plt.subplots(ncols=4, nrows=2, figsize=(20, 10))
index = 0
axs = axs.flatten()
colors = ['b', 'g', 'r', 'purple', 'pink', 'y', 'orange', 'brown']
for i, k in enumerate(column_sels):
    sns.regplot(y=y, x=x[k], ax=axs[i], color=colors[i])
plt.tight_layout(pad=0.4, w_pad=0.5, h_pad=5.0)
```