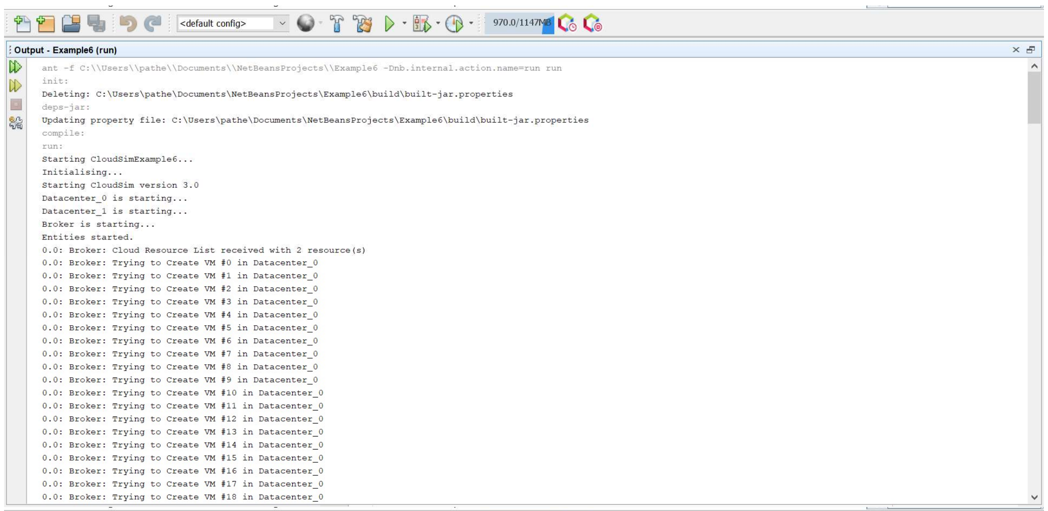
Practical No 3 [Part 2]

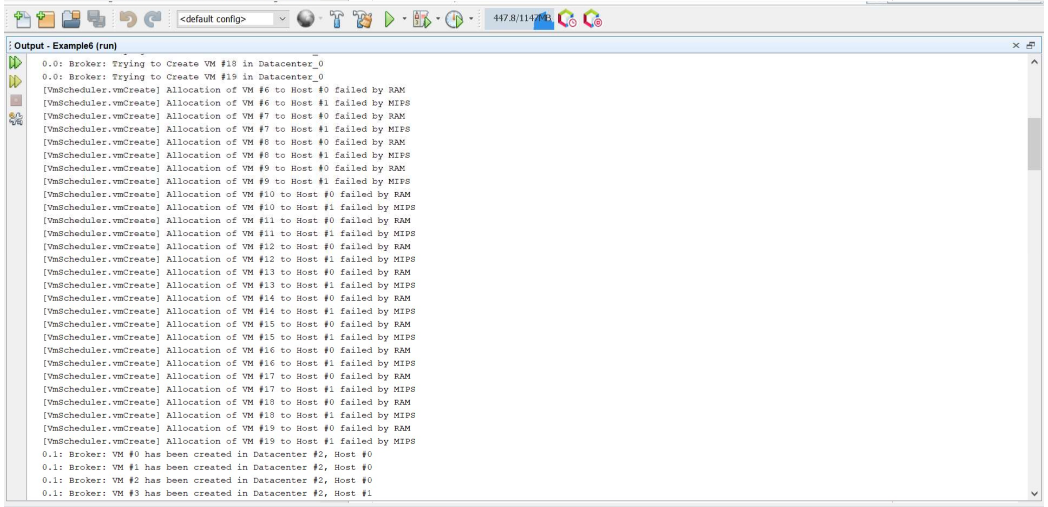
Name: Atharva Paliwal
Roll No: B 40
Date: 17-09-2021

Example 6

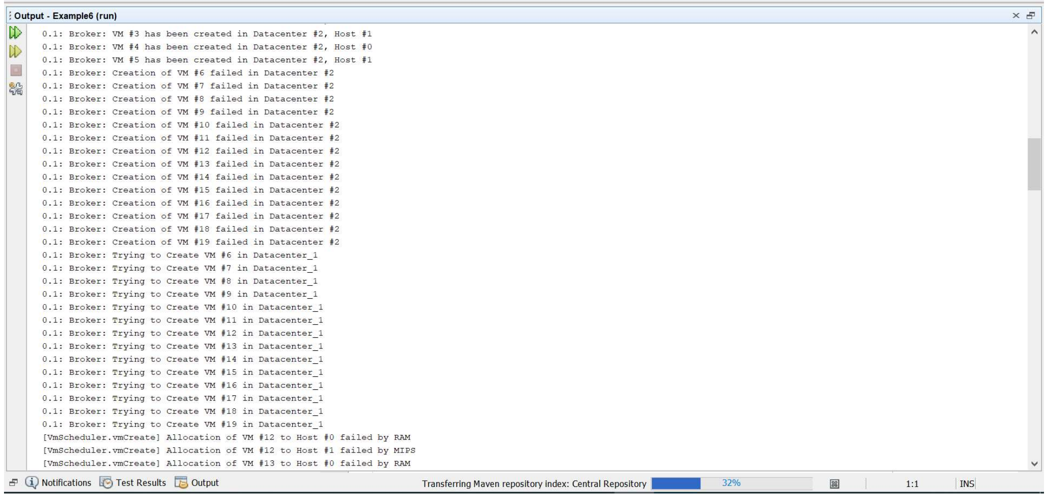
```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package example6;

//public class Example6 {
//
//    /**
//     * @param args the command line arguments
//     */
//    public static void main(String[] args) {
//        // TODO code application logic here
//    }
//
//}

/*
 * 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 Example6 {

    /** 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()));
            }
        }

    }
}
```



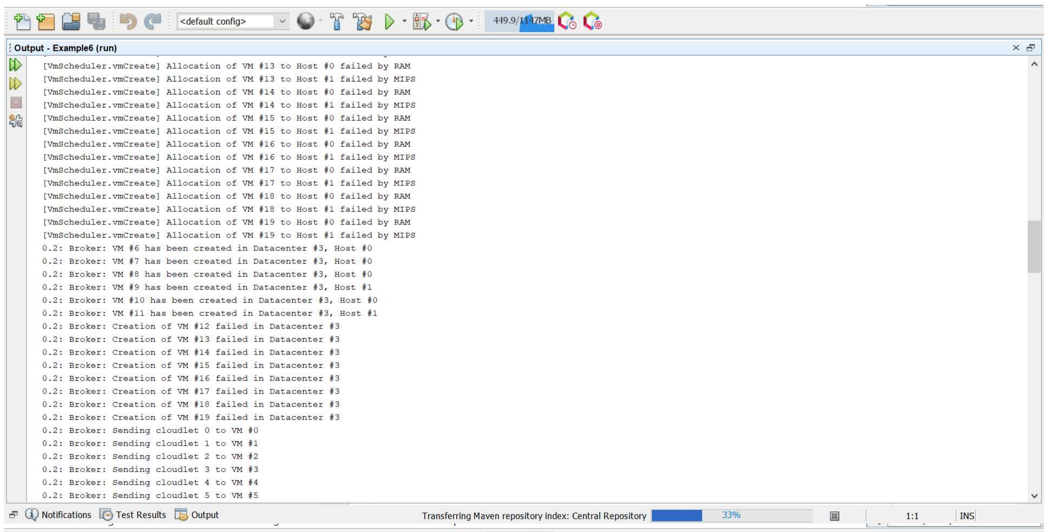
```
ant -f C:\\Users\\pathe\\Documents\\NetBeansProjects\\Example6 -Dnb.internal.action.name=run run
init:
Deleting: C:\Users\pathe\Documents\NetBeansProjects\Example6\build\built-jar.properties
deps-jar:
Updating property file: C:\Users\pathe\Documents\NetBeansProjects\Example6\build\built-jar.properties
compile:
run:
Starting CloudSimExample6...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Datacenter_1 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 2 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.0: Broker: Trying to Create VM #1 in Datacenter_0
0.0: Broker: Trying to Create VM #2 in Datacenter_0
0.0: Broker: Trying to Create VM #3 in Datacenter_0
0.0: Broker: Trying to Create VM #4 in Datacenter_0
0.0: Broker: Trying to Create VM #5 in Datacenter_0
0.0: Broker: Trying to Create VM #6 in Datacenter_0
0.0: Broker: Trying to Create VM #7 in Datacenter_0
0.0: Broker: Trying to Create VM #8 in Datacenter_0
0.0: Broker: Trying to Create VM #9 in Datacenter_0
0.0: Broker: Trying to Create VM #10 in Datacenter_0
0.0: Broker: Trying to Create VM #11 in Datacenter_0
0.0: Broker: Trying to Create VM #12 in Datacenter_0
0.0: Broker: Trying to Create VM #13 in Datacenter_0
0.0: Broker: Trying to Create VM #14 in Datacenter_0
0.0: Broker: Trying to Create VM #15 in Datacenter_0
0.0: Broker: Trying to Create VM #16 in Datacenter_0
0.0: Broker: Trying to Create VM #17 in Datacenter_0
0.0: Broker: Trying to Create VM #18 in Datacenter_0
```



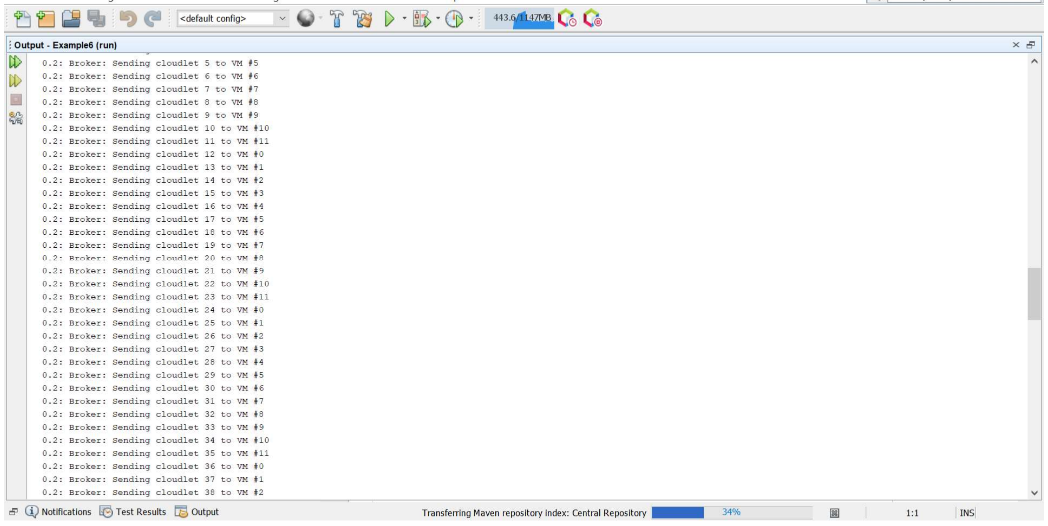
```
0.0: Broker: Trying to Create VM #18 in Datacenter_0
0.0: Broker: Trying to Create VM #19 in Datacenter_0
[VmScheduler.vmCreate] Allocation of VM #6 to Host #0 failed by RAM
[VmScheduler.vmCreate] Allocation of VM #6 to Host #1 failed by MIPS
[VmScheduler.vmCreate] Allocation of VM #7 to Host #0 failed by RAM
[VmScheduler.vmCreate] Allocation of VM #7 to Host #1 failed by MIPS
[VmScheduler.vmCreate] Allocation of VM #8 to Host #0 failed by RAM
[VmScheduler.vmCreate] Allocation of VM #8 to Host #1 failed by MIPS
[VmScheduler.vmCreate] Allocation of VM #9 to Host #0 failed by RAM
[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
```



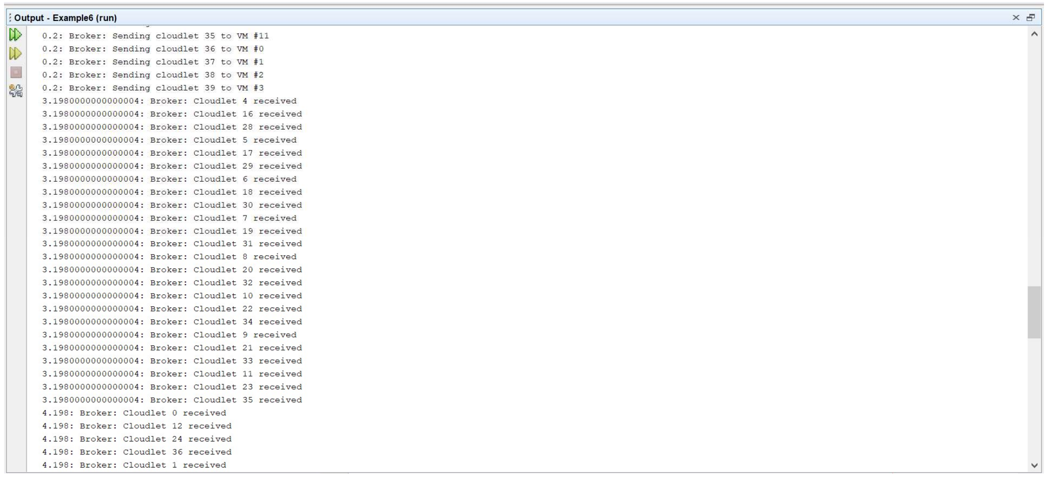
```
0.1: Broker: VM #3 has been created in Datacenter #2, Host #1
0.1: Broker: VM #4 has been created in Datacenter #2, Host #0
0.1: Broker: VM #5 has been created in Datacenter #2, Host #1
0.1: Broker: Creation of VM #6 failed in Datacenter #2
0.1: Broker: Creation of VM #7 failed in Datacenter #2
0.1: 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
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
```



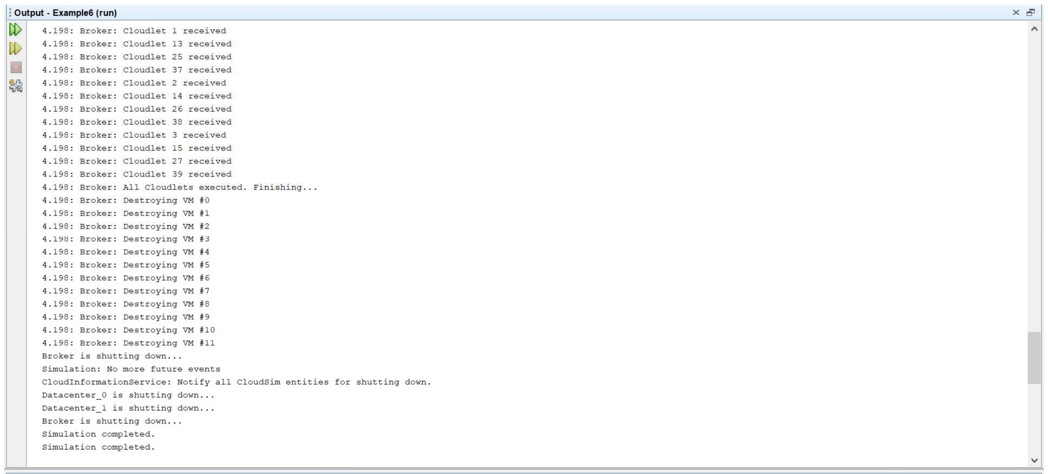
```
[VmScheduler.vmCreate] Allocation of VM #13 to Host #0 failed by RAM
[VmScheduler.vmCreate] Allocation of VM #13 to Host #1 failed by MIPS
[VmScheduler.vmCreate] Allocation of VM #14 to Host #0 failed by RAM
[VmScheduler.vmCreate] Allocation of VM #14 to Host #1 failed by MIPS
[VmScheduler.vmCreate] 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
0.2: 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
```



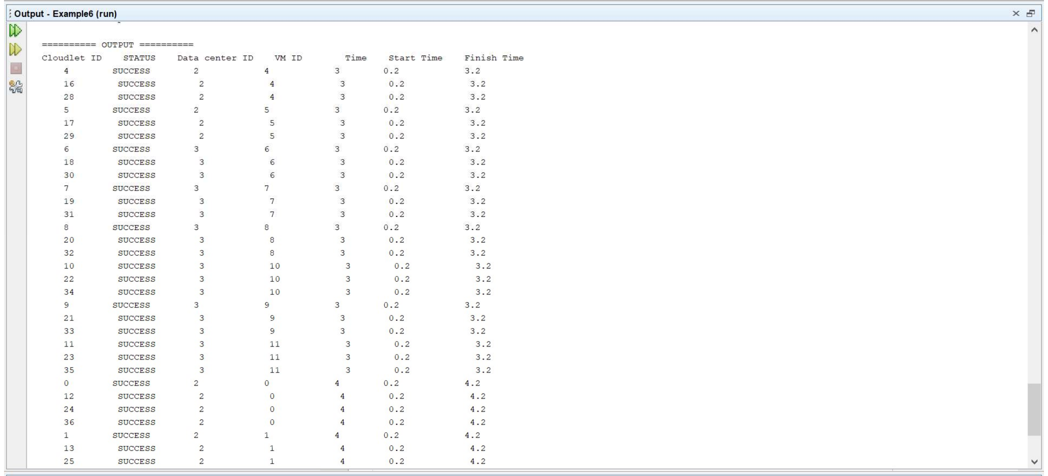
```
0.2: Broker: Sending cloudlet 5 to VM #5
0.2: Broker: Sending cloudlet 6 to VM #6
0.2: Broker: Sending cloudlet 7 to VM #7
0.2: Broker: Sending cloudlet 8 to VM #8
0.2: Broker: Sending cloudlet 9 to VM #9
0.2: Broker: Sending cloudlet 10 to VM #10
0.2: Broker: Sending cloudlet 11 to VM #11
0.2: Broker: 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
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
```



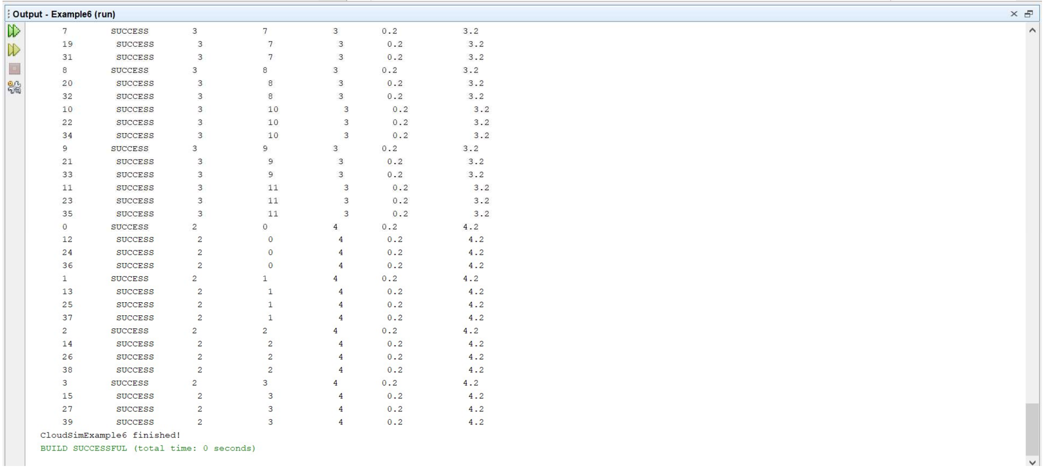
```
0.2: Broker: Sending cloudlet 35 to VM #11
0.2: Broker: Sending cloudlet 36 to VM #0
0.2: Broker: Sending cloudlet 37 to VM #1
0.2: Broker: Sending cloudlet 38 to VM #2
0.2: Broker: Sending cloudlet 39 to VM #3
3.1980000000000004: Broker: Cloudlet 4 received
3.1980000000000004: Broker: Cloudlet 16 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
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
```


Output - Example6 (run)

```
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
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 - Example6 (run)

```
========== 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
    33        SUCCESS        3            9            3        0.2            3.2
    11        SUCCESS        3            11            3        0.2            3.2
    23        SUCCESS        3            11            3        0.2            3.2
    35        SUCCESS        3            11            3        0.2            3.2
    0        SUCCESS        2            0            4        0.2            4.2
    12        SUCCESS        2            0            4        0.2            4.2
    24        SUCCESS        2            0            4        0.2            4.2
    36        SUCCESS        2            0            4        0.2            4.2
    1        SUCCESS        2            1            4        0.2            4.2
    13        SUCCESS        2            1            4        0.2            4.2
    25        SUCCESS        2            1            4        0.2            4.2
```


Output - Example6 (run)

```
    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
    33        SUCCESS        3            9            3        0.2            3.2
    11        SUCCESS        3            11            3        0.2            3.2
    23        SUCCESS        3            11            3        0.2            3.2
    35        SUCCESS        3            11            3        0.2            3.2
    0        SUCCESS        2            0            4        0.2            4.2
    12        SUCCESS        2            0            4        0.2            4.2
    24        SUCCESS        2            0            4        0.2            4.2
    36        SUCCESS        2            0            4        0.2            4.2
    1        SUCCESS        2            1            4        0.2            4.2
    13        SUCCESS        2            1            4        0.2            4.2
    25        SUCCESS        2            1            4        0.2            4.2
    37        SUCCESS        2            1            4        0.2            4.2
    2        SUCCESS        2            2            4        0.2            4.2
    14        SUCCESS        2            2            4        0.2            4.2
    26        SUCCESS        2            2            4        0.2            4.2
    38        SUCCESS        2            2            4        0.2            4.2
    3        SUCCESS        2            3            4        0.2            4.2
    15        SUCCESS        2            3            4        0.2            4.2
    27        SUCCESS        2            3            4        0.2            4.2
    39        SUCCESS        2            3            4        0.2            4.2
CloudSimExample6 finished!
BUILD SUCCESSFUL (total time: 0 seconds)
```

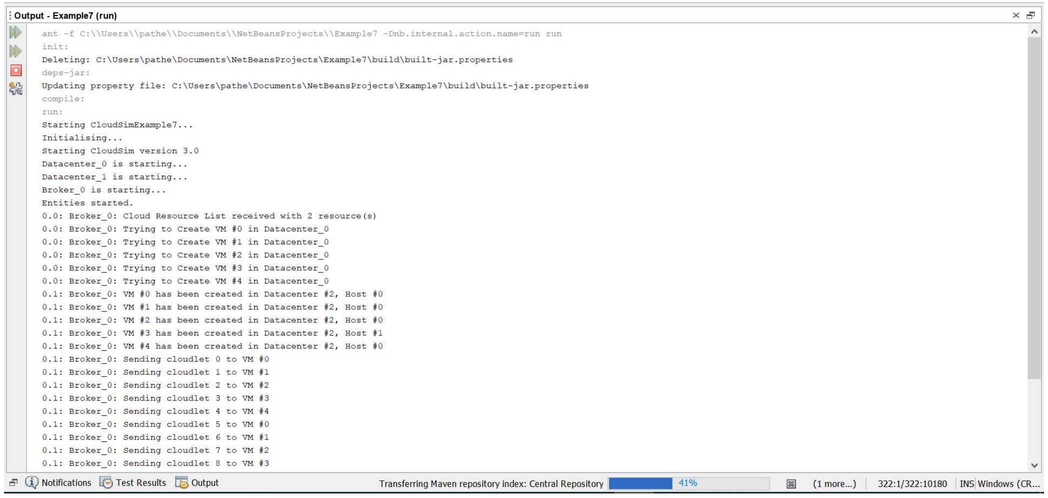
Example 7

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package example7;

//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 Example7 {

    /** 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
```

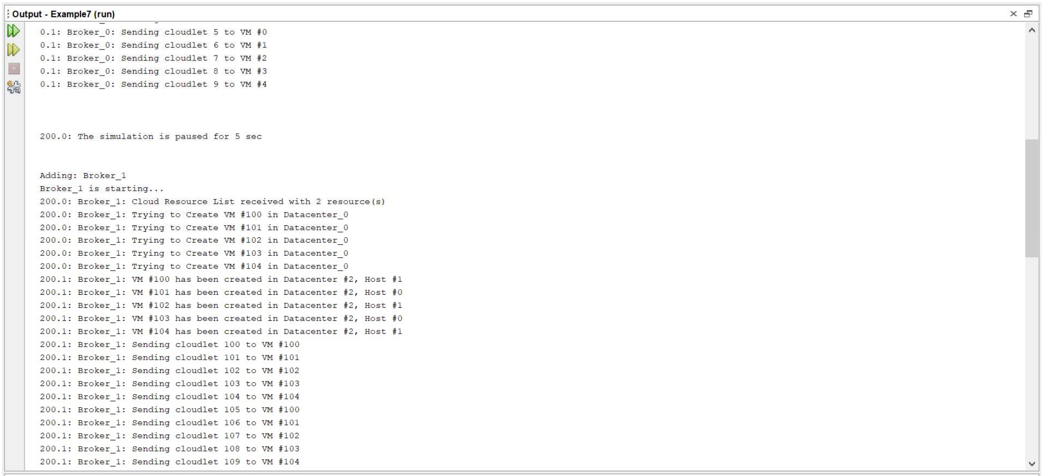
```
        LinkedList<Vm> list = new LinkedList<Vm>();

        //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()));
            }
        }

    }
}
```

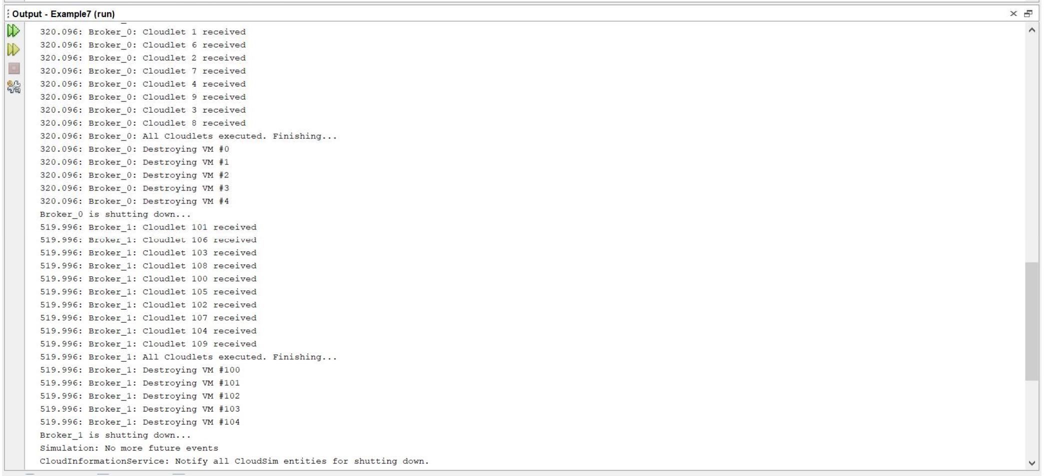

Output - Example7 (run)

```
ant -f C:\\Users\\pathe\\Documents\\NetBeansProjects\\Example7 -Dnb.internal.action.name=run run
init:
Deleting: C:\Users\pathe\Documents\NetBeansProjects\Example7\build\built-jar.properties
deps-jar:
Updating property file: C:\Users\pathe\Documents\NetBeansProjects\Example7\build\built-jar.properties
compile:
run:
Starting CloudSimExample7...
Initialising...
Starting CloudSim version 3.0
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 #2, Host #0
0.1: Broker_0: VM #1 has been created in Datacenter #2, Host #0
0.1: Broker_0: VM #2 has been created in Datacenter #2, Host #0
0.1: Broker_0: VM #3 has been created in Datacenter #2, Host #1
0.1: Broker_0: VM #4 has been created in Datacenter #2, 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
```



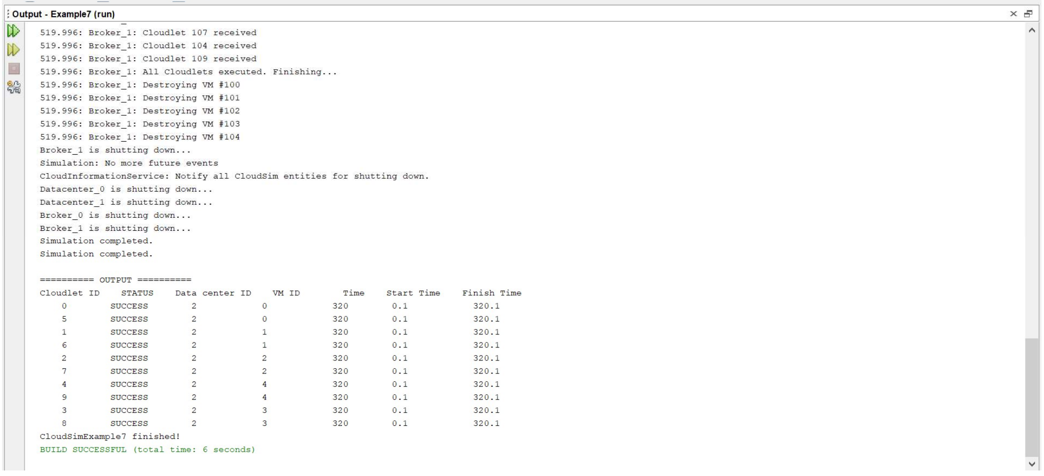
Output - Example7 (run)

```
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



200.0: The simulation is paused for 5 sec


Adding: Broker_1
Broker_1 is starting...
200.0: Broker_1: Cloud Resource List received with 2 resource(s)
200.0: Broker_1: Trying to Create VM #100 in Datacenter_0
200.0: Broker_1: Trying to Create VM #101 in Datacenter_0
200.0: Broker_1: Trying to Create VM #102 in Datacenter_0
200.0: Broker_1: Trying to Create VM #103 in Datacenter_0
200.0: Broker_1: Trying to Create VM #104 in Datacenter_0
200.1: Broker_1: VM #100 has been created in Datacenter #2, Host #1
200.1: Broker_1: VM #101 has been created in Datacenter #2, Host #0
200.1: Broker_1: VM #102 has been created in Datacenter #2, Host #1
200.1: Broker_1: VM #103 has been created in Datacenter #2, Host #0
200.1: Broker_1: VM #104 has been created in Datacenter #2, Host #1
200.1: Broker_1: Sending cloudlet 100 to VM #100
200.1: Broker_1: Sending cloudlet 101 to VM #101
200.1: Broker_1: Sending cloudlet 102 to VM #102
200.1: Broker_1: Sending cloudlet 103 to VM #103
200.1: Broker_1: Sending cloudlet 104 to VM #104
200.1: Broker_1: Sending cloudlet 105 to VM #100
200.1: Broker_1: Sending cloudlet 106 to VM #101
200.1: Broker_1: Sending cloudlet 107 to VM #102
200.1: Broker_1: Sending cloudlet 108 to VM #103
200.1: Broker_1: Sending cloudlet 109 to VM #104
```



Output - Example7 (run)

```
320.096: Broker_0: Cloudlet 1 received
320.096: Broker_0: Cloudlet 6 received
320.096: Broker_0: Cloudlet 2 received
320.096: Broker_0: Cloudlet 7 received
320.096: Broker_0: Cloudlet 4 received
320.096: Broker_0: Cloudlet 9 received
320.096: Broker_0: Cloudlet 3 received
320.096: Broker_0: Cloudlet 8 received
320.096: Broker_0: All Cloudlets executed. Finishing...
320.096: Broker_0: Destroying VM #0
320.096: Broker_0: Destroying VM #1
320.096: Broker_0: Destroying VM #2
320.096: Broker_0: Destroying VM #3
320.096: Broker_0: Destroying VM #4
Broker_0 is shutting down...
519.996: Broker_1: Cloudlet 101 received
519.996: Broker_1: Cloudlet 106 received
519.996: Broker_1: Cloudlet 103 received
519.996: Broker_1: Cloudlet 108 received
519.996: Broker_1: Cloudlet 100 received
519.996: Broker_1: Cloudlet 105 received
519.996: Broker_1: Cloudlet 102 received
519.996: Broker_1: Cloudlet 107 received
519.996: Broker_1: Cloudlet 104 received
519.996: Broker_1: Cloudlet 109 received
519.996: Broker_1: All Cloudlets executed. Finishing...
519.996: Broker_1: Destroying VM #100
519.996: Broker_1: Destroying VM #101
519.996: Broker_1: Destroying VM #102
519.996: Broker_1: Destroying VM #103
519.996: Broker_1: Destroying VM #104
Broker_1 is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
```



Output - Example7 (run)

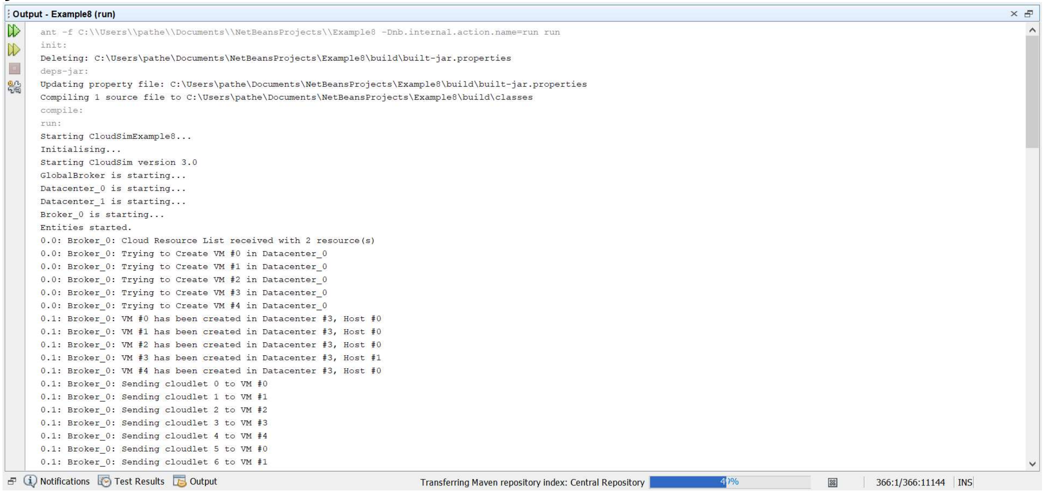
```
519.996: Broker_1: Cloudlet 107 received
519.996: Broker_1: Cloudlet 104 received
519.996: Broker_1: Cloudlet 109 received
519.996: Broker_1: All Cloudlets executed. Finishing...
519.996: Broker_1: Destroying VM #100
519.996: Broker_1: Destroying VM #101
519.996: Broker_1: Destroying VM #102
519.996: Broker_1: Destroying VM #103
519.996: Broker_1: Destroying VM #104
Broker_1 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_0 is shutting down...
Broker_1 is shutting down...
Simulation completed.
Simulation completed.

========== 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!
BUILD SUCCESSFUL (total time: 6 seconds)
```

Example 8

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package example8;
/*
 * 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 global manager entity (GlobalBroker).
 */
```

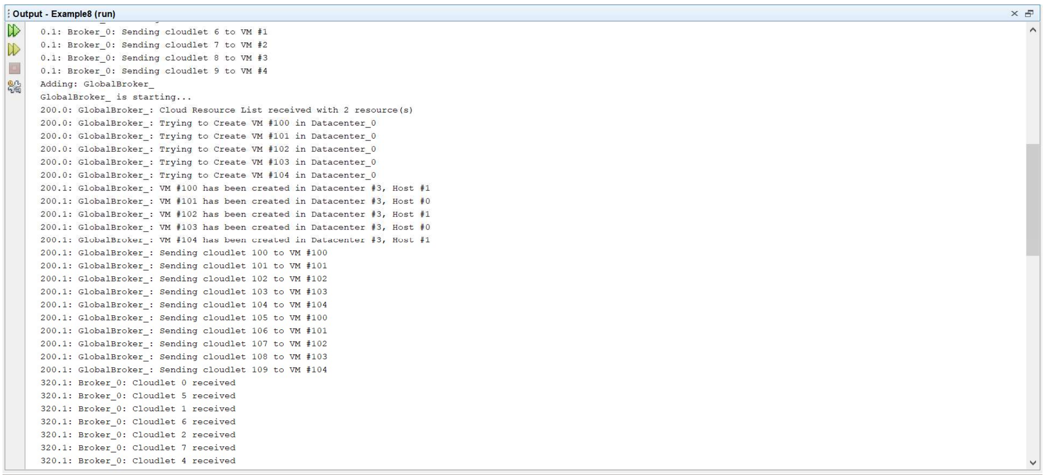
```
public class Example8 {

    /** 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;
       }

   }

}
```

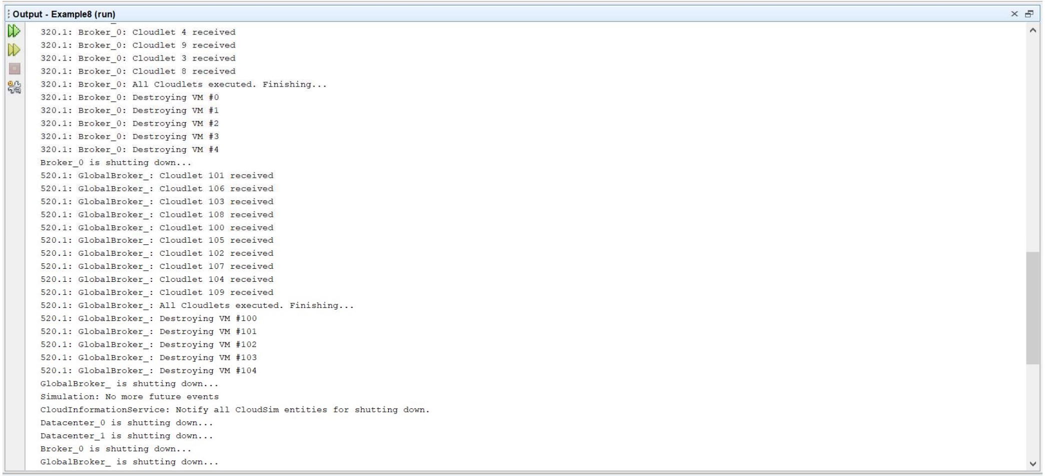


Output - Example8 (run)

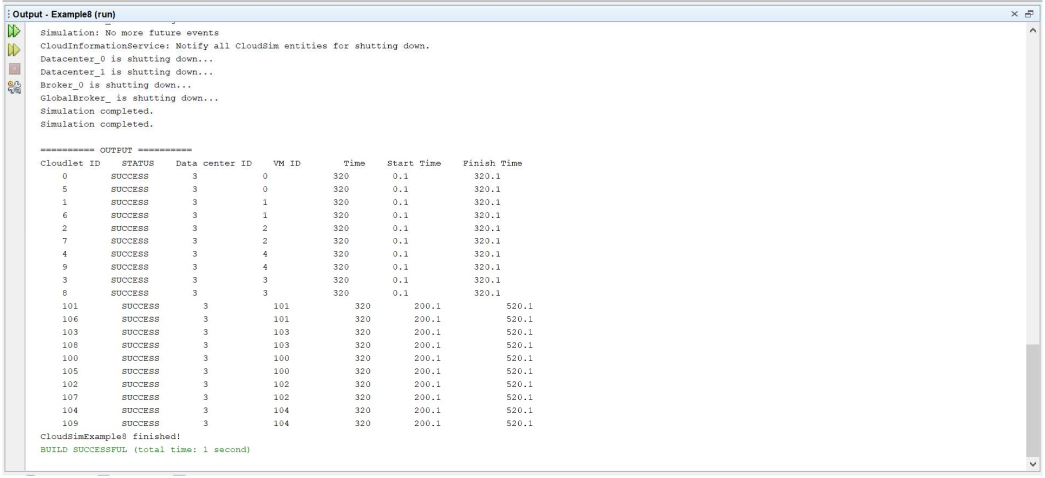
```
ant -f C:\\Users\\pathe\\Documents\\NetBeansProjects\\Example8 -Dnb.internal.action.name=run run
init:
Deleting: C:\Users\pathe\Documents\NetBeansProjects\Example8\build\built-jar.properties
deps-jar:
Updating property file: C:\Users\pathe\Documents\NetBeansProjects\Example8\build\built-jar.properties
Compiling 1 source file to C:\Users\pathe\Documents\NetBeansProjects\Example8\build\classes
compile:
run:
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 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
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 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
520.1: GlobalBroker_: Cloudlet 106 received
520.1: GlobalBroker_: Cloudlet 103 received
520.1: GlobalBroker_: Cloudlet 108 received
520.1: GlobalBroker_: Cloudlet 100 received
520.1: GlobalBroker_: Cloudlet 105 received
520.1: GlobalBroker_: Cloudlet 102 received
520.1: GlobalBroker_: Cloudlet 107 received
520.1: GlobalBroker_: Cloudlet 104 received
520.1: GlobalBroker_: Cloudlet 109 received
520.1: GlobalBroker_: All Cloudlets executed. Finishing...
520.1: GlobalBroker_: Destroying VM #100
520.1: GlobalBroker_: Destroying VM #101
520.1: GlobalBroker_: Destroying VM #102
520.1: GlobalBroker_: Destroying VM #103
520.1: GlobalBroker_: Destroying VM #104
GlobalBroker_ 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_0 is shutting down...
GlobalBroker_ 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_0 is shutting down...
GlobalBroker_ is shutting down...
Simulation completed.
Simulation completed.

========== 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!
BUILD SUCCESSFUL (total time: 1 second)
```