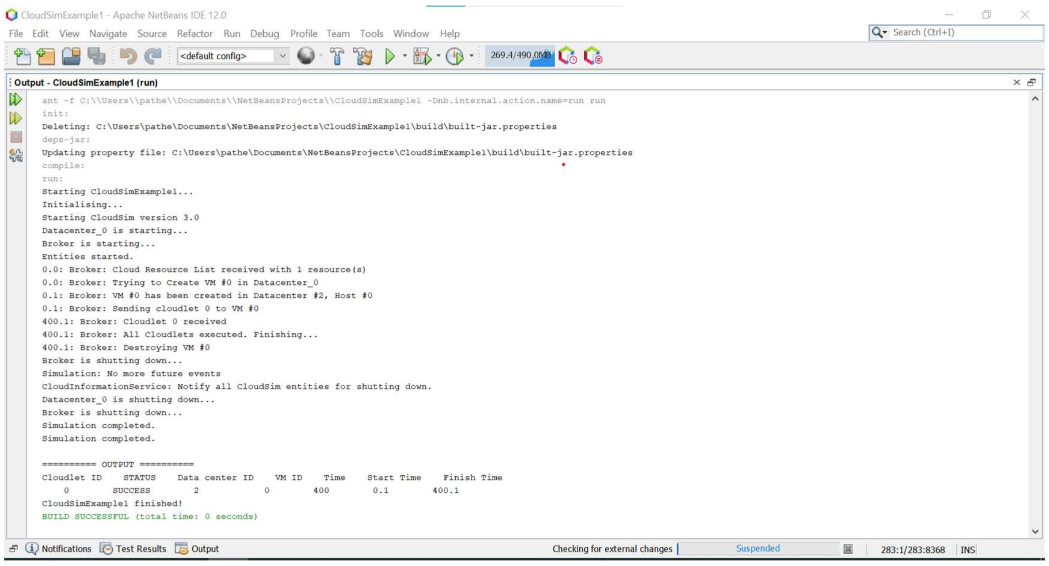
Practical No 3 [Part1]

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

CloudSim Example 1:

```
/*
 * 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 cloudsimexample1;

//public class CloudSimExample1 {
//
//    /**
//     * @param args the command line arguments
//     */
//    public static void main(String[] args) {
//        // TODO code application logic here
//    }
//
//}
//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:
```



```
ant -f C:\\Users\\pathe\\Documents\\NetBeansProjects\\CloudSimExample1 -Dnb.internal.action.name=run run
init:
Deleting: C:\Users\pathe\Documents\NetBeansProjects\CloudSimExample1\build\built-jar.properties
deps-jar:
Updating property file: C:\Users\pathe\Documents\NetBeansProjects\CloudSimExample1\build\built-jar.properties
compile:
run:
Starting CloudSimExample1...
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
0.1: Broker: VM #0 has been created in Datacenter #2, Host #0
0.1: Broker: Sending cloudlet 0 to VM #0
400.1: Broker: Cloudlet 0 received
400.1: Broker: All Cloudlets executed. Finishing...
400.1: Broker: Destroying VM #0
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
    0        SUCCESS        2            0        400        0.1        400.1
CloudSimExample1 finished!
BUILD SUCCESSFUL (total time: 0 seconds)
```

CloudSim Example 2:

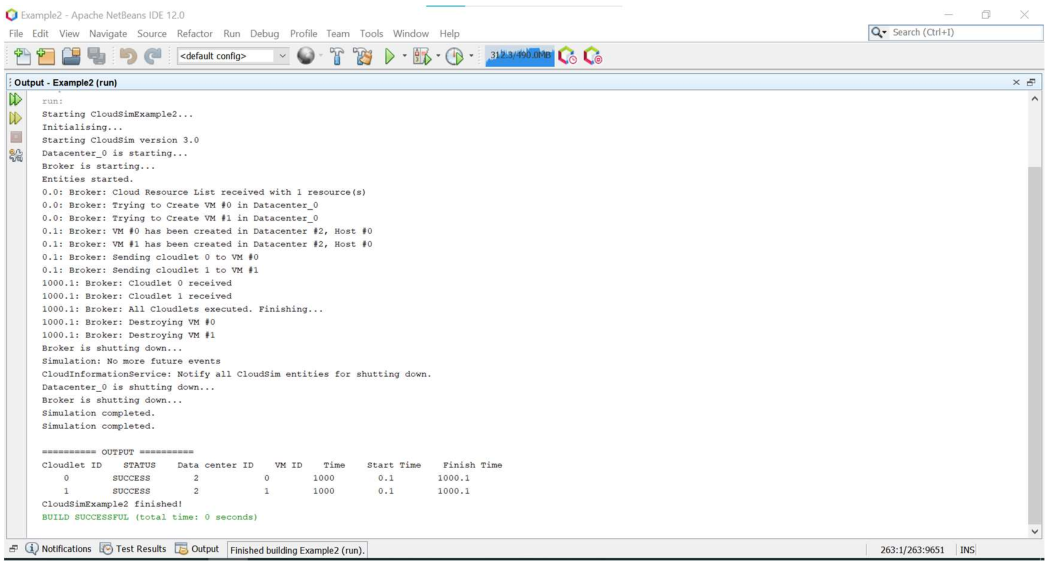
```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package example2;


//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 one host and run two
 * cloudlets on it. The cloudlets run in
 * VMs with the same MIPS requirements.
 * The cloudlets will take the same time to
 * complete the execution.
 */
public class Example2 {

    /** 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 CloudSimExample2...");

            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 = 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;
                 pesNumber=1;
                 long length = 250000;
                 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("CloudSimExample2 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;
            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
        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:


Example2 - Apache NetBeans IDE 12.0
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default config>
Output - Example2 (run)
Notifications
Test Results
Output
Finished building Example2 (run).
263:1/263:9651
INS

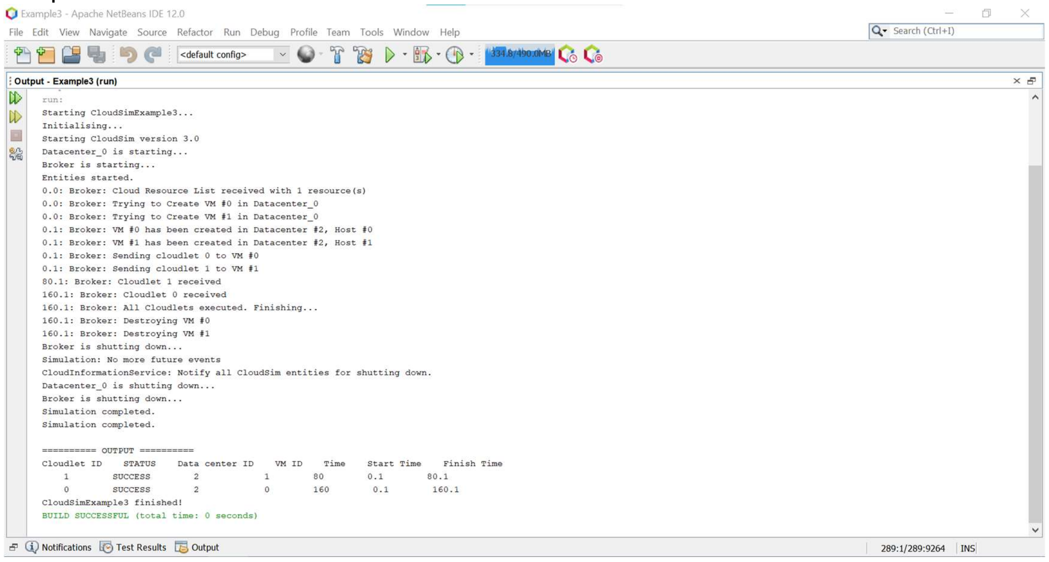
```
run:
Starting CloudSimExample2...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 1 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.0: Broker: Trying to Create VM #1 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 #0
0.1: Broker: Sending cloudlet 0 to VM #0
0.1: Broker: Sending cloudlet 1 to VM #1
1000.1: Broker: Cloudlet 0 received
1000.1: Broker: Cloudlet 1 received
1000.1: Broker: All Cloudlets executed. Finishing...
1000.1: Broker: Destroying VM #0
1000.1: 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
    0        SUCCESS        2            0        1000        0.1        1000.1
    1        SUCCESS        2            1        1000        0.1        1000.1
CloudSimExample2 finished!
BUILD SUCCESSFUL (total time: 0 seconds)
```

CloudSim Example 3:

```
/*
 * 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 example3;

/////*
// * 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 Example3 {

    /** 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:



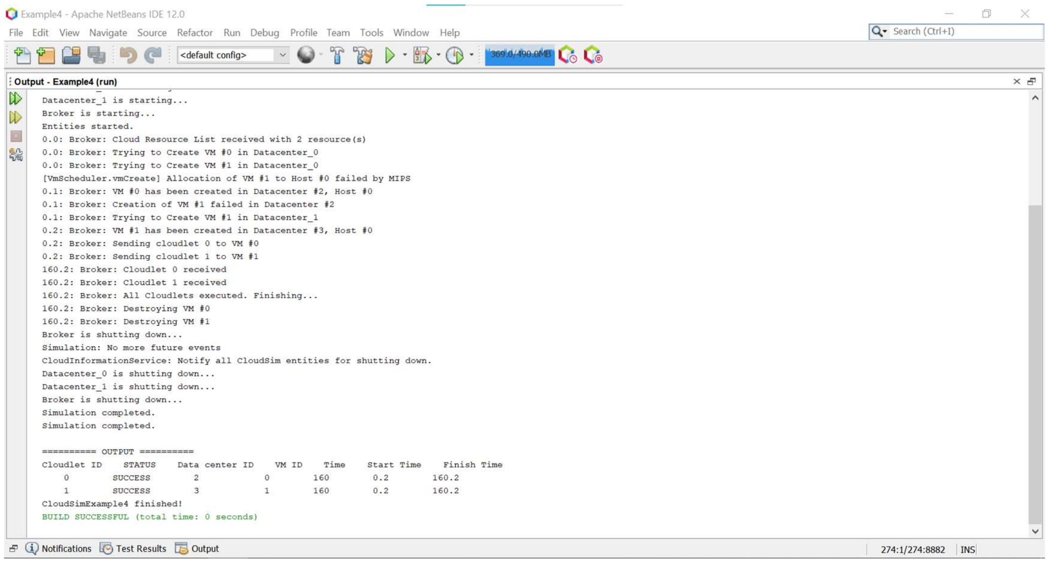
```
run:
Starting CloudSimExample3...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 1 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.0: Broker: Trying to Create VM #1 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!
BUILD SUCCESSFUL (total time: 0 seconds)
```

Example3 - Apache NetBeans IDE 12.0
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default config>
334.8/490.0MB
Output - Example3 (run)
Notifications
Test Results
Output
289:1/289:9264
INS

CloudSim Example 4:

```
/*
 * 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 example4;

* 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 Example4 {
    /** 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:


Example4 - Apache NetBeans IDE 12.0
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default config>
369.0/490.0MB
Output - Example4 (run)
Notifications
Test Results
Output
274:1/274:8882
INS

```
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
[VmScheduler.vmCreate] Allocation of VM #1 to Host #0 failed by MIPS
0.1: Broker: VM #0 has been created in Datacenter #2, Host #0
0.1: Broker: Creation of VM #1 failed in Datacenter #2
0.1: Broker: Trying to Create VM #1 in Datacenter_1
0.2: Broker: VM #1 has been created in Datacenter #3, Host #0
0.2: Broker: Sending cloudlet 0 to VM #0
0.2: Broker: Sending cloudlet 1 to VM #1
160.2: Broker: Cloudlet 0 received
160.2: Broker: Cloudlet 1 received
160.2: Broker: All Cloudlets executed. Finishing...
160.2: Broker: Destroying VM #0
160.2: 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...
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
    0        SUCCESS        2            0        160        0.2        160.2
    1        SUCCESS        3            1        160        0.2        160.2
CloudSimExample4 finished!
BUILD SUCCESSFUL (total time: 0 seconds)
```

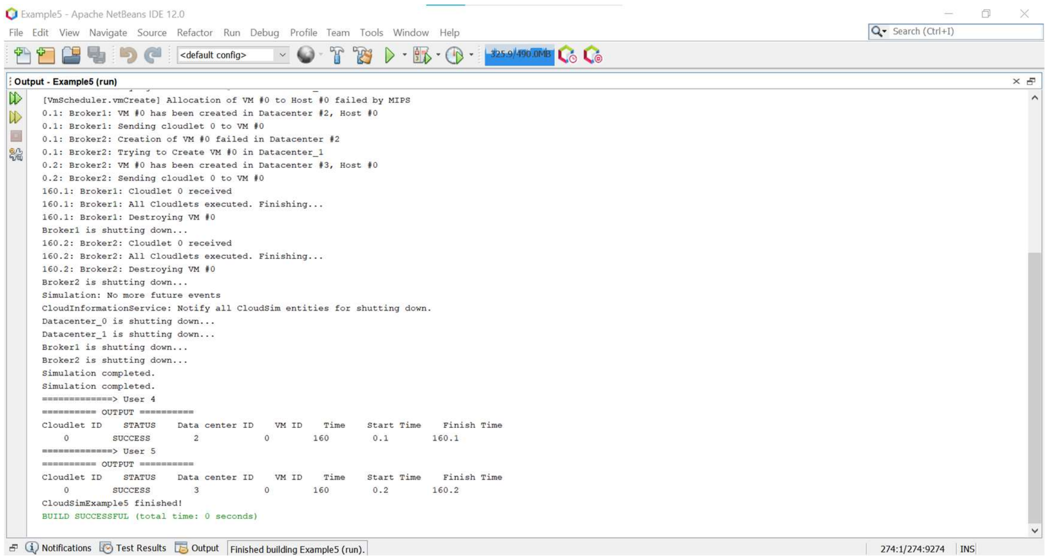
CloudSim Example 5:

```
/*
 * 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 example5;
/*
 * 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 Example5 {
    /** 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:



```
[VmScheduler.vmCreate] Allocation of VM #0 to Host #0 failed by MIPS
0.1: Broker1: VM #0 has been created in Datacenter #2, Host #0
0.1: Broker1: Sending cloudlet 0 to VM #0
0.1: Broker2: Creation of VM #0 failed in Datacenter #2
0.1: Broker2: Trying to Create VM #0 in Datacenter_1
0.2: Broker2: VM #0 has been created in Datacenter #3, Host #0
0.2: Broker2: Sending cloudlet 0 to VM #0
160.1: Broker1: Cloudlet 0 received
160.1: Broker1: All Cloudlets executed. Finishing...
160.1: Broker1: Destroying VM #0
Broker1 is shutting down...
160.2: Broker2: Cloudlet 0 received
160.2: Broker2: All Cloudlets executed. Finishing...
160.2: Broker2: Destroying VM #0
Broker2 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...
Broker1 is shutting down...
Broker2 is shutting down...
Simulation completed.
Simulation completed.
=============> 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!
BUILD SUCCESSFUL (total time: 0 seconds)
```

Analysis: