**Task – 4: Implement CloudSim using Eclipse ( Example -1 )**

**DESCRIPTION :** Using CloudSim simulate a data enter with one host and run one cloudlet on it. CloudSim is a framework for modelling and simulation of cloud computing infrastructures and services. Originally built primarily at the Cloud Computing and Distributed Systems (CLOUDS) Laboratory, The University of Melbourne, Australia, CloudSim has become one of the most popular open source cloud simulators in the research and academia. CloudSim is completely written in Java.

**AIM :** To implement CloudSim using Eclipse.

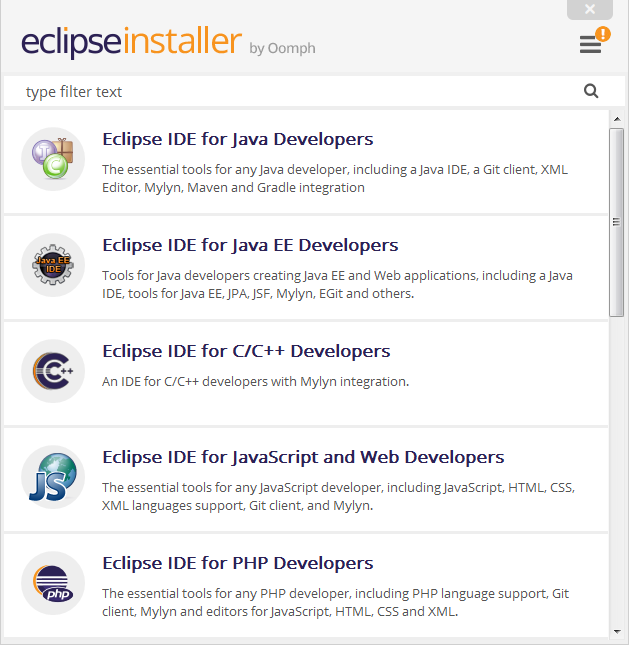
**PROCEDURE :**

The sequence of steps to implement a CloudSim using Eclipse...

**Step-1:**Download cloudsim-3.0.3from the link <https://code.google.com/archive/p/cloudsim/downloads> and extract it to a directory.

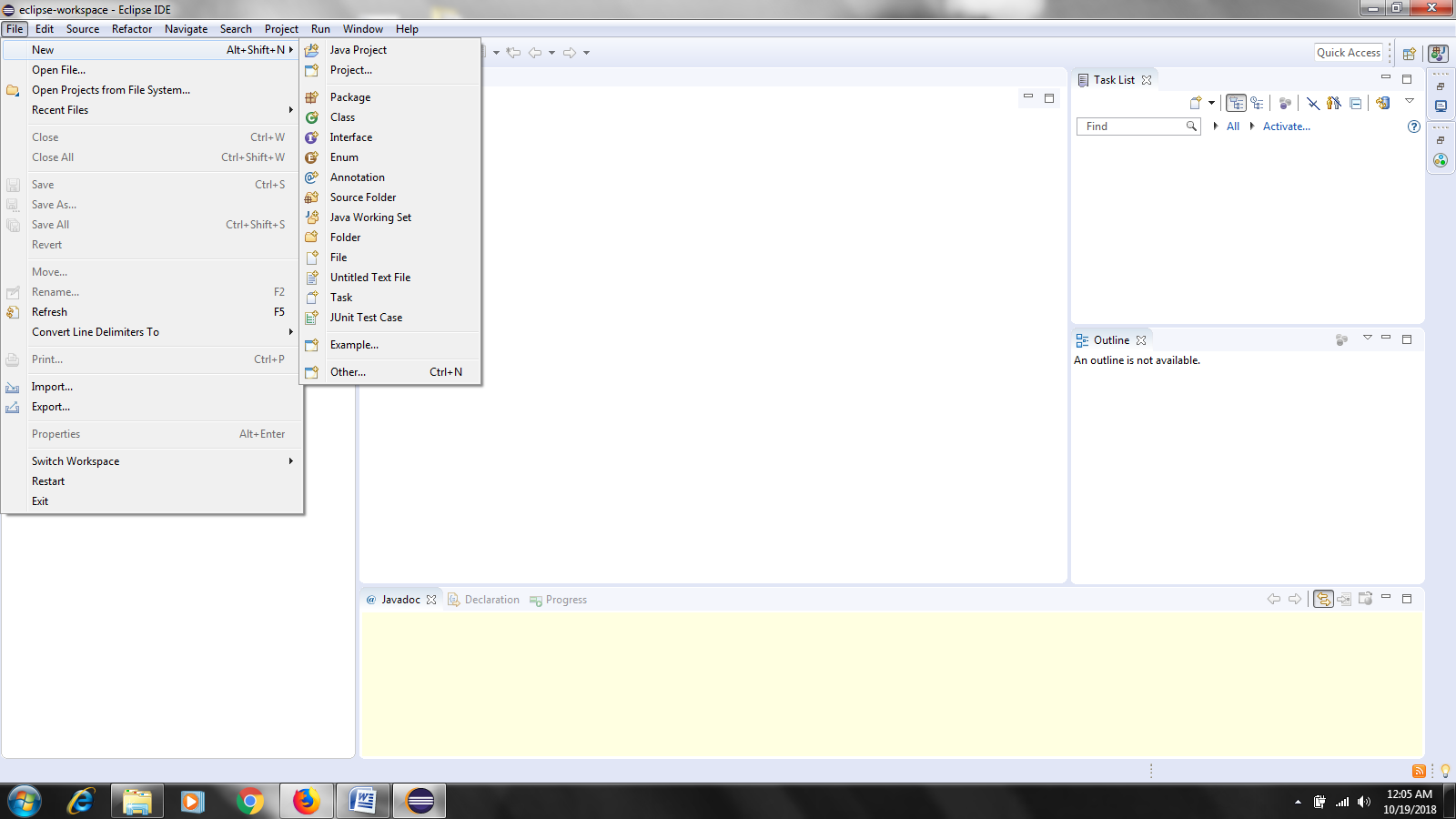
**Step-2:** Download Eclipse from <https://www.eclipse.org/downloads/> and

Run the exe file and choose Eclipse IDE for Java Developers.

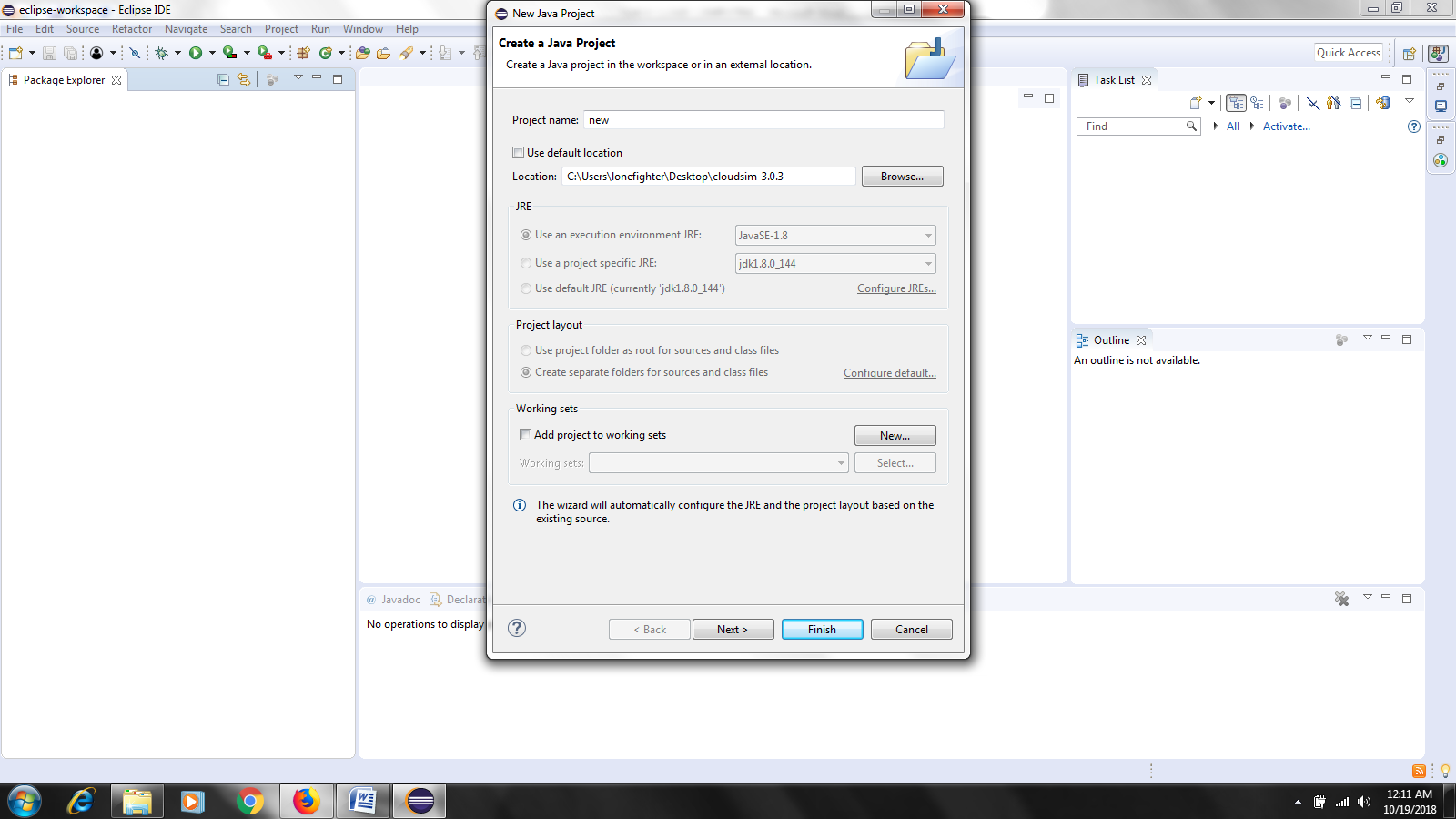


**Step-3:** Create a Java Project and name the project.

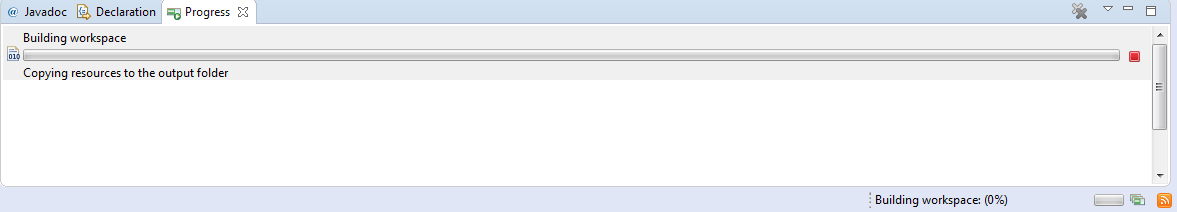
Click **File** -> **New** -> **JavaProject**

****

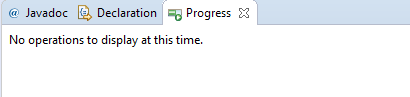
**Step-4:** Name the project and uncheck default location. Set the location to installation path of cloudsim-3.0.3.



Now we can see the progress at the bottom during the launching of the project in Eclipse IDE.

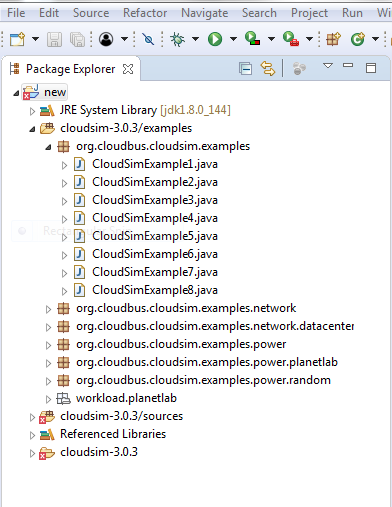


After Building the workspace we can see the following which results in further process of the project.



**Step-5:** Open the CloudSimExample1.java and run the program as Java Application.

Open the Example as follows.



Click on your project which is at the left side of the window.

**Expand your project** -> **Expand cloudsim Examples folder** -> **Expand org.cloudbus-cloudsim.examples** **folder and then select the Example 1 and double click it.**

The program is as follows :

CloudSimExample1.java

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;

public class CloudSimExample1 {

private static List<Cloudlet> cloudletList;

private static List<Vm> vmlist;

@SuppressWarnings("unused")

public static void main(String[] args) {

Log.printLine("Starting CloudSimExample1...");

try {

int num\_user = 1; // number of cloud users

Calendar calendar = Calendar.getInstance();

boolean trace\_flag = false; // mean trace events

CloudSim.init(num\_user, calendar, trace\_flag);

Datacenter datacenter0 = createDatacenter("Datacenter\_0");

DatacenterBroker broker = createBroker();

int brokerId = broker.getId();

vmlist = new ArrayList<Vm>();

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

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

CloudletSchedulerTimeShared());

vmlist.add(vm);

broker.submitVmList(vmlist);

cloudletList = new ArrayList<Cloudlet>();

int id = 0;

long length = 400000;

long fileSize = 300;

long outputSize = 300;

UtilizationModel utilizationModel = new UtilizationModelFull();

Cloudlet cloudlet = new Cloudlet(id, length, pesNumber, fileSize, outputSize,

utilizationModel, utilizationModel, utilizationModel);

cloudlet.setUserId(brokerId);

cloudlet.setVmId(vmid);

cloudletList.add(cloudlet);

broker.submitCloudletList(cloudletList);

CloudSim.startSimulation();

CloudSim.stopSimulation();

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

printCloudletList(newList);

Log.printLine("CloudSimExample1 finished!");

} catch (Exception e) {

e.printStackTrace();

Log.printLine("Unwanted errors happen");

}

}

private static Datacenter createDatacenter(String name) {

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

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

int mips = 1000;

peList.add(new Pe(0, new PeProvisionerSimple(mips))); // need to store Pe id and MIPS

//Rating

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

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

double costPerBw = 0.0; // the cost of using bw in this resource

LinkedList<Storage> storageList = new LinkedList<Storage>(); // we are not adding SAN

DatacenterCharacteristics characteristics = new DatacenterCharacteristics(

arch, os, vmm, hostList, time\_zone, cost, costPerMem,

costPerStorage, costPerBw);

Datacenter datacenter = null;

try {

datacenter = new Datacenter(name, characteristics, new

VmAllocationPolicySimple(hostList), storageList, 0);

} catch (Exception e) {

e.printStackTrace();

}

return datacenter;

}

private static DatacenterBroker createBroker() {

DatacenterBroker broker = null;

try {

broker = new DatacenterBroker("Broker");

} catch (Exception e) {

e.printStackTrace();

return null;

}

return broker;

}

private static void printCloudletList(List<Cloudlet> list) {

int size = list.size();

Cloudlet cloudlet;

String indent = " ";

Log.printLine();

Log.printLine("========== OUTPUT ==========");

Log.printLine("Cloudlet ID" + indent + "STATUS" + indent

+ "Data center ID" + indent + "VM ID" + indent + "Time" + indent

+ "Start Time" + indent + "Finish Time");

DecimalFormat dft = new DecimalFormat("###.##");

for (int i = 0; i < size; i++) {

cloudlet = list.get(i);

Log.print(indent + cloudlet.getCloudletId() + indent + indent);

if (cloudlet.getCloudletStatus() == Cloudlet.SUCCESS) {

Log.print("SUCCESS");

Log.printLine(indent + indent + cloudlet.getResourceId()

+ indent + indent + indent + cloudlet.getVmId()

+ indent + indent

+ dft.format(cloudlet.getActualCPUTime()) + indent

+ indent + dft.format(cloudlet.getExecStartTime())

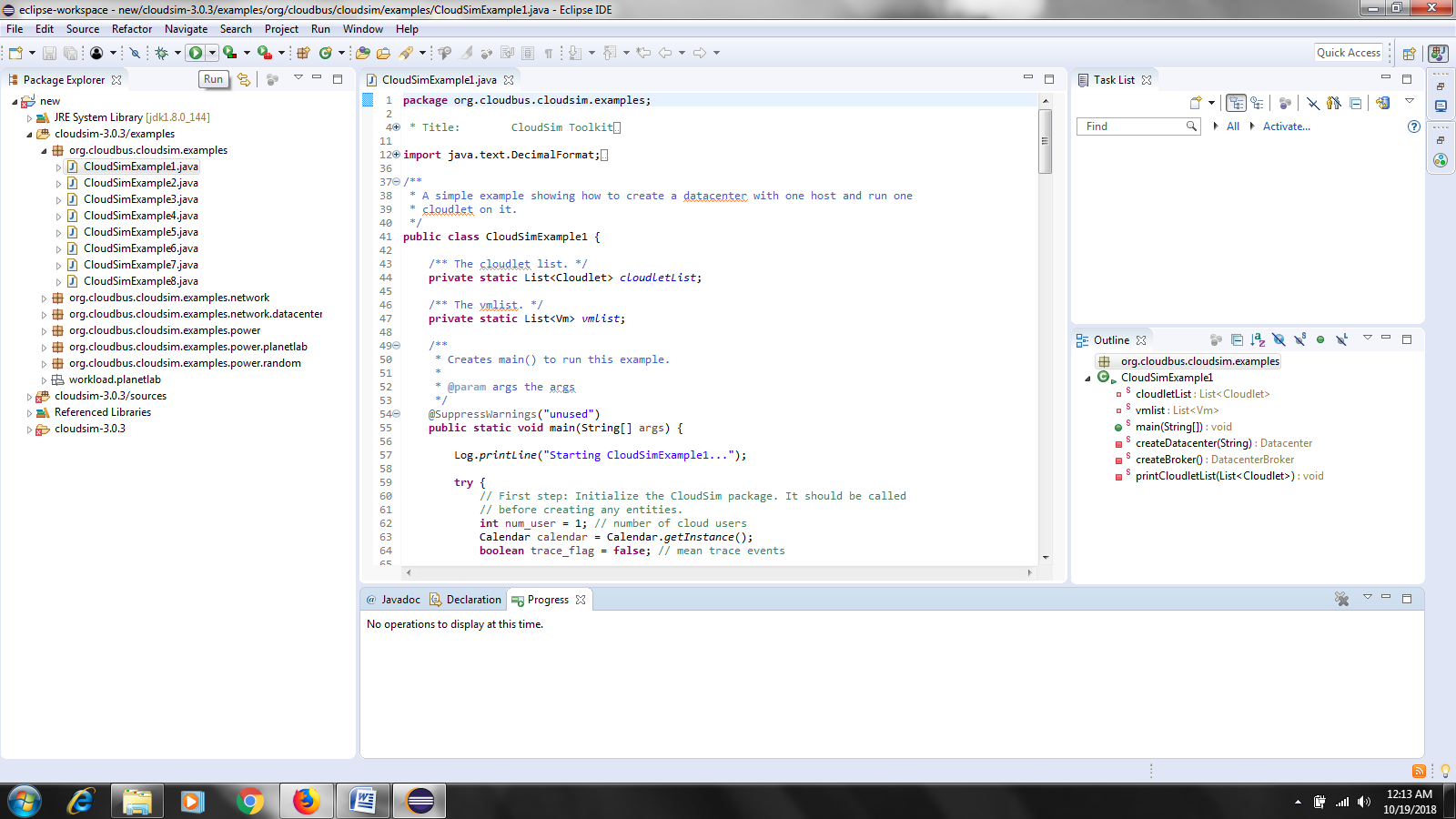
+ indent + indent

+ dft.format(cloudlet.getFinishTime()));}

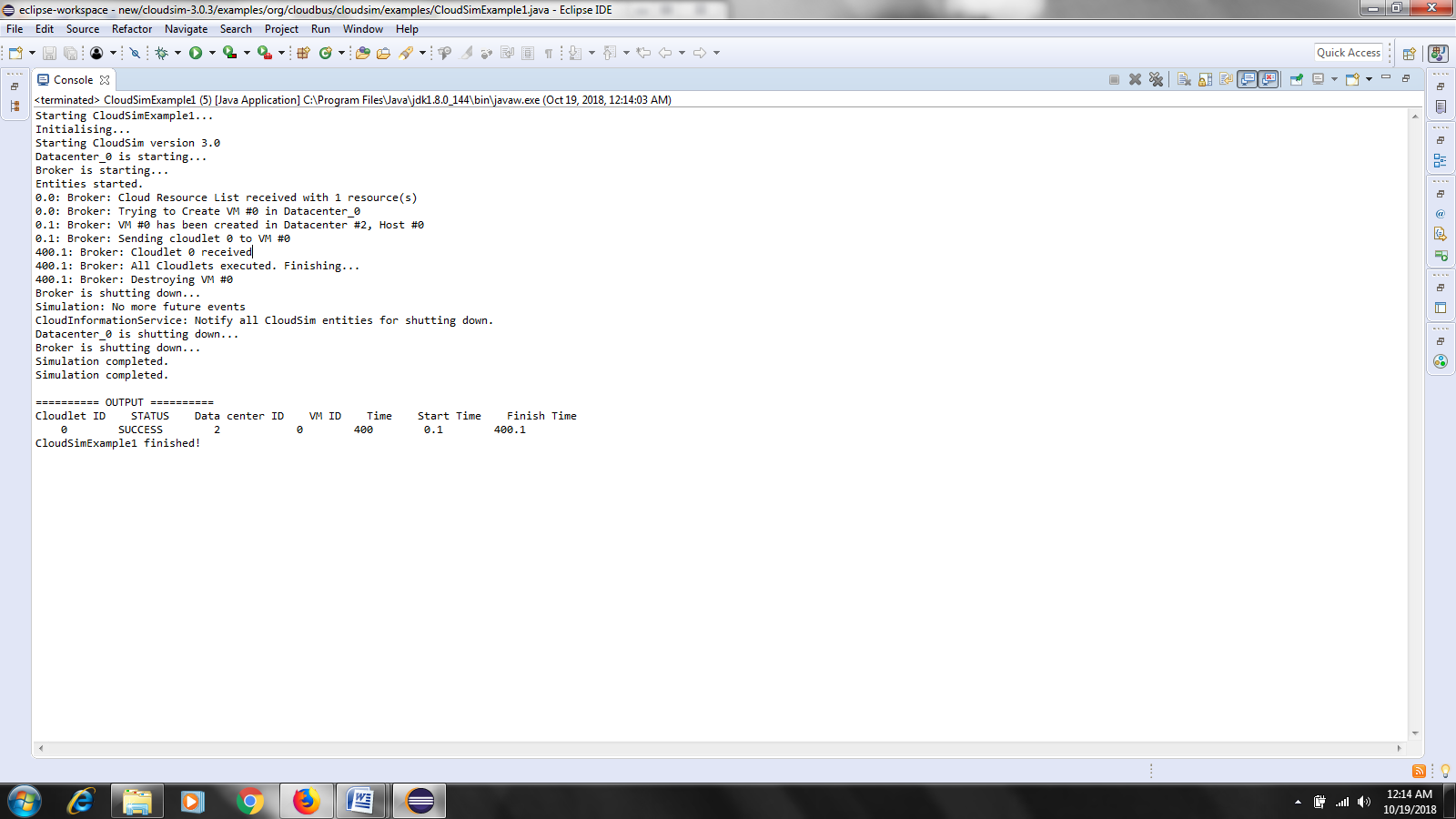
}

}

}

Now Run the Program by clicking on the Run Button proceed if it shows errors :

The output of the above program is…



**RESULT :** Hence**,** implementation of CloudSim using Eclipse is done successfully.