Ex. No. 5

INSTALLATION AND CONFIGURATION OF CLOUDSIM IN ECLIPSE IDE

AIM:

To install and configure the CloudSim in Eclipse IDE and run a java program in it.

PROCEDURE:

1. Java Installation

a. Check java in your system.

b. If java not installed then download Java.

c. Install java setup.

d. Set path variable for java.

2. Download CloudSim and additional jar file

a. Download CloudSim 3.0.3

b. Download Commons math 3 jar file.

3. Eclipse IDE Installation

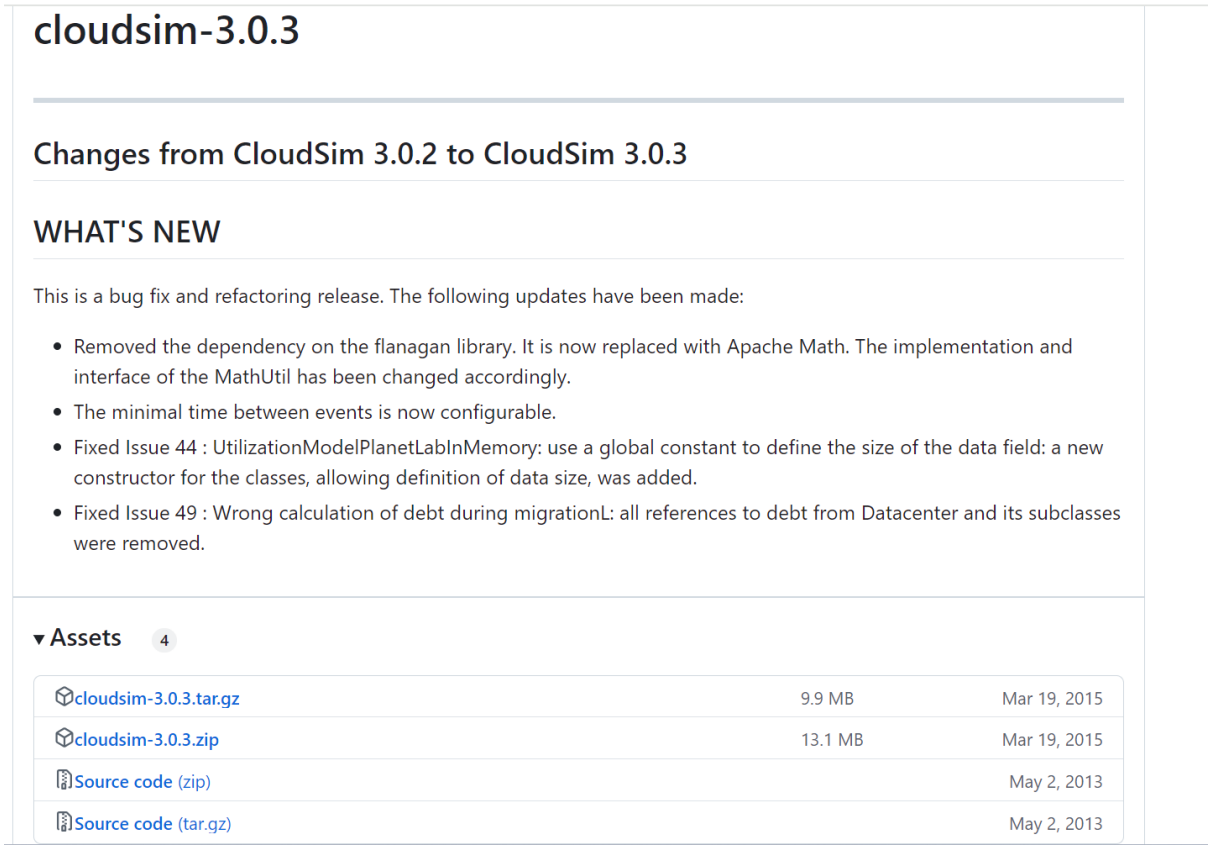
a. If java 64 bit Installed then download 64-bit Eclipse otherwise java 32 bit then download 32 eclipse.

b. Install Eclipse IDE.

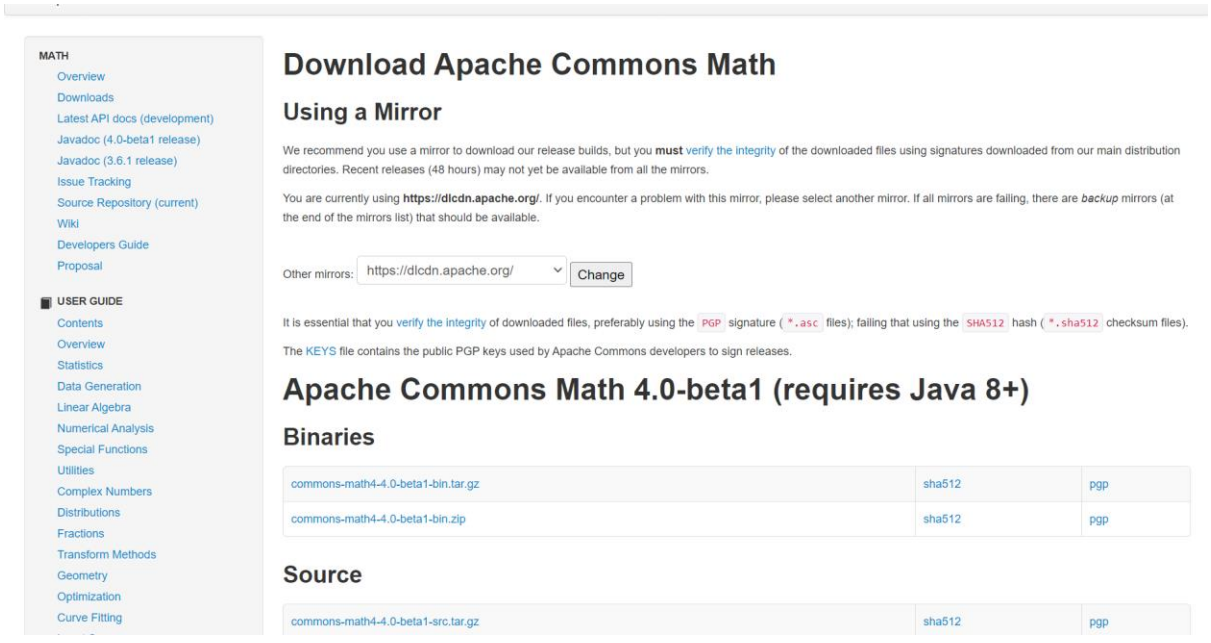
4. Run CloudSim in Eclipse

a. Put up commons-math-3-3.6.1.jar file into jar folder of Cloudsim3.0.3

b. Build a new java project with Cloudsim3.0.3 folder.



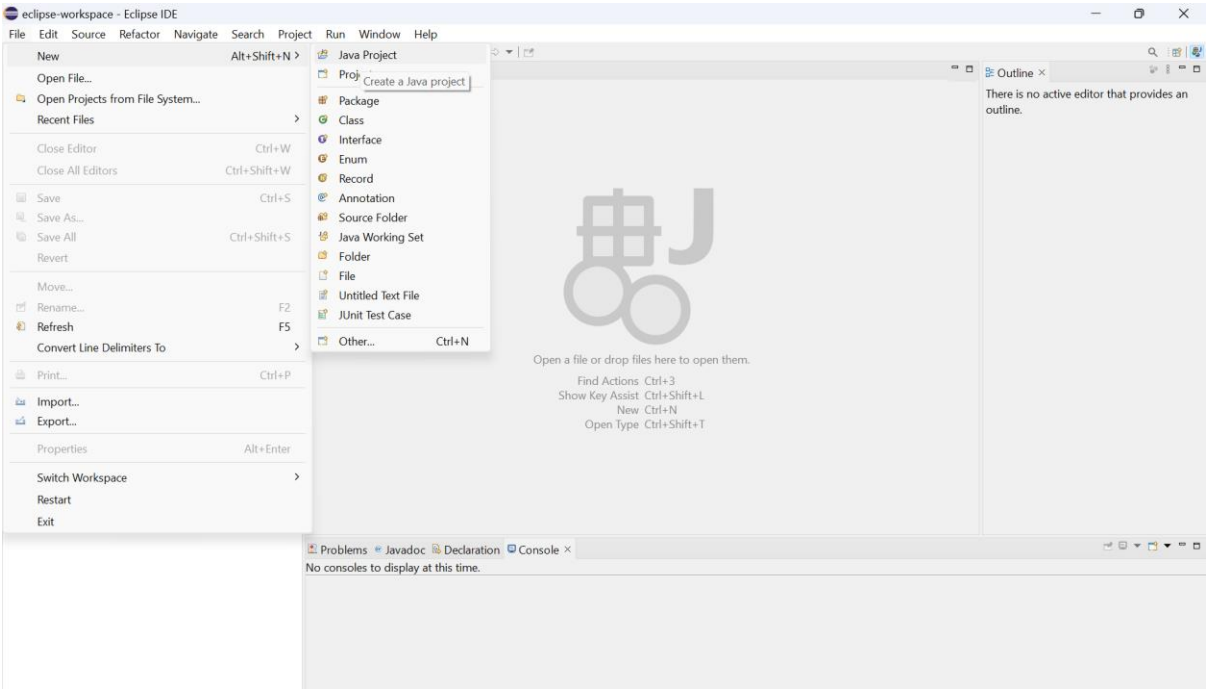
Download the .zip file

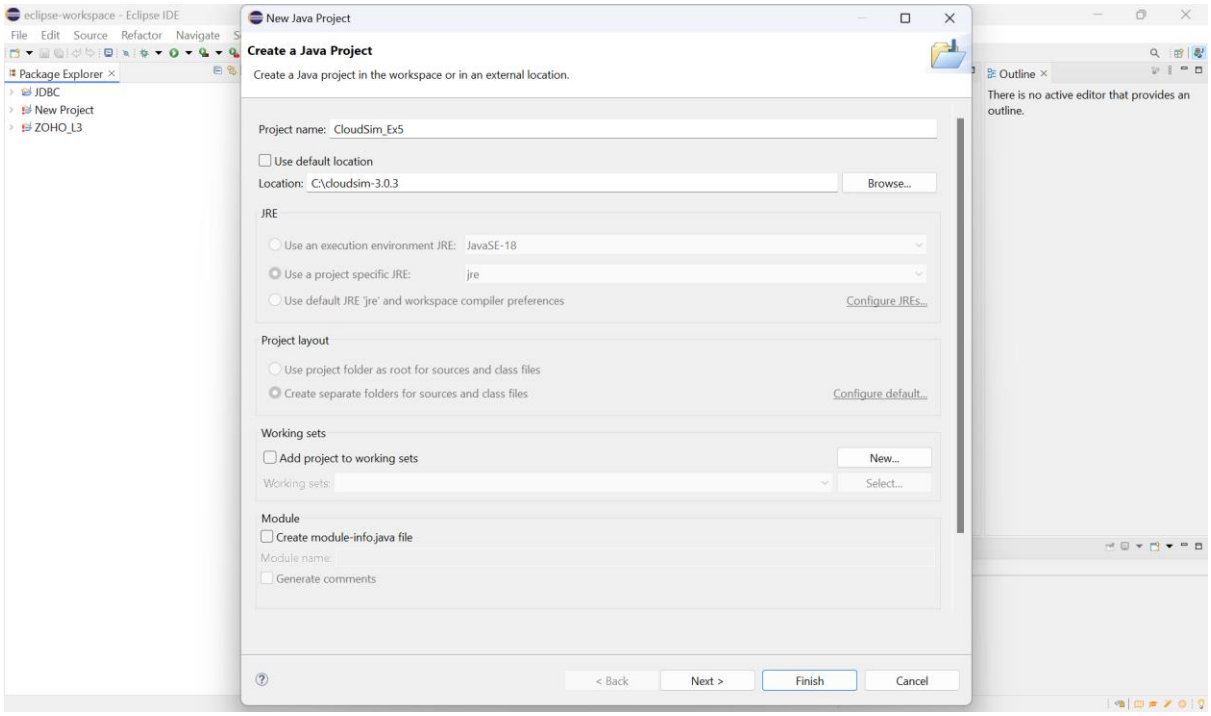


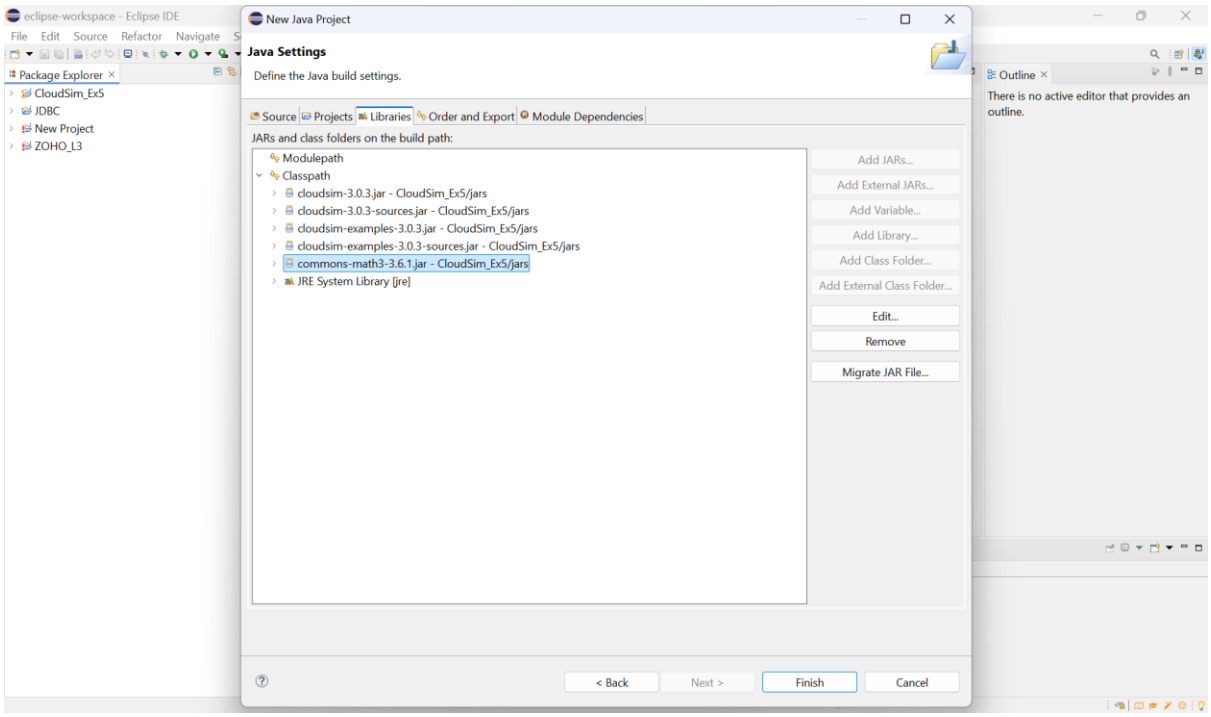
Download the .zip file

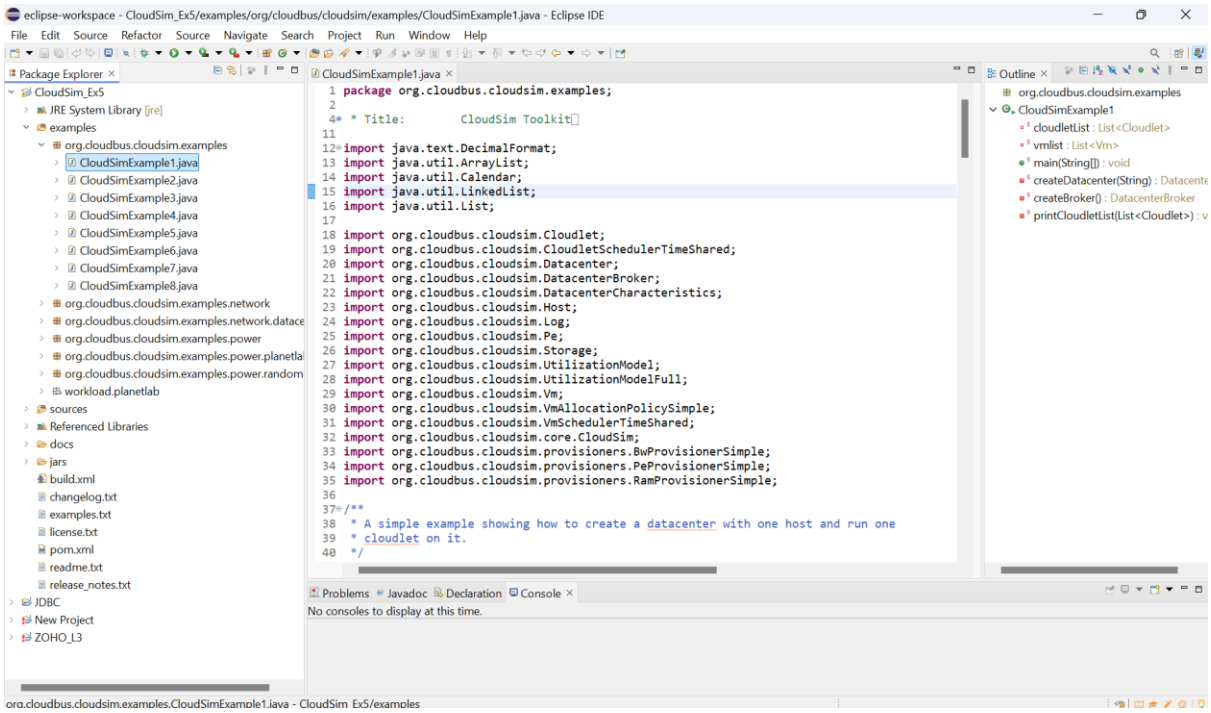
Name	Date modified	Type	Size
docs	17-03-2016 12:59	File folder	
commons-math3-3.6.1	17-03-2016 13:34	Executable Jar File	2,162 KB
commons-math3-3.6.1-javadoc	17-03-2016 13:34	Executable Jar File	6,680 KB
commons-math3-3.6.1-sources	17-03-2016 12:59	Executable Jar File	2,456 KB
commons-math3-3.6.1-tests	17-03-2016 12:59	Executable Jar File	2,906 KB
commons-math3-3.6.1-test-sources	17-03-2016 12:59	Executable Jar File	1,914 KB
commons-math3-3.6.1-tools	17-03-2016 13:27	Executable Jar File	12 KB
LICENSE	19-02-2015 13:38	Text Document	24 KB
NOTICE	16-03-2016 09:50	Text Document	1 KB
RELEASE-NOTES	17-03-2016 11:17	Text Document	2 KB

Name	Date modified	Type	Size
cloudsim-3.0.3	02-05-2013 19:56	Executable Jar File	241 KB
cloudsim-3.0.3-sources	02-05-2013 19:56	Executable Jar File	232 KB
cloudsim-examples-3.0.3	02-05-2013 19:56	Executable Jar File	4,953 KB
cloudsim-examples-3.0.3-sources	02-05-2013 19:57	Executable Jar File	4,945 KB
commons-math3-3.6.1	17-03-2016 13:34	Executable Jar File	2,162 KB




eclipse-workspace - Eclipse IDE
File Edit Source Refactor Navigate
Package Explorer
JDBC
New Project
ZOHO_L3
New Java Project
Create a Java Project
Create a Java project in the workspace or in an external location.
Project name: CloudSim_Ex5
Use default location
Location: C:\cloudsim-3.0.3
Browse...
JRE
Use an execution environment JRE: JavaSE-18
Use a project specific JRE: jre
Use default JRE 'jre' and workspace compiler preferences
Configure JREs...
Project layout
Use project folder as root for sources and class files
Create separate folders for sources and class files
Configure default...
Working sets
Add project to working sets
New...
Working sets:
Select...
Module
Create module-info.java file
Module name:
Generate comments
< Back
Next >
Finish
Cancel
Outline
There is no active editor that provides an outline.


eclipse-workspace - Eclipse IDE
File Edit Source Refactor Navigate
Package Explorer
CloudSim_Ex5
JDBC
New Project
ZOHO_L3
New Java Project
Java Settings
Define the Java build settings.
Source
Projects
Libraries
Order and Export
Module Dependencies
JARs and class folders on the build path:
Modulepath
Classpath
cloudsim-3.0.3.jar - CloudSim_Ex5/jars
cloudsim-3.0.3-sources.jar - CloudSim_Ex5/jars
cloudsim-examples-3.0.3.jar - CloudSim_Ex5/jars
cloudsim-examples-3.0.3-sources.jar - CloudSim_Ex5/jars
commons-math3-3.6.1.jar - CloudSim_Ex5/jars
JRE System Library [jre]
Add JARs...
Add External JARs...
Add Variable...
Add Library...
Add Class Folder...
Add External Class Folder...
Edit...
Remove
Migrate JAR File...
< Back
Next >
Finish
Cancel
Outline
There is no active editor that provides an outline.



PROGRAM:

```
package org.cloudbus.cloudsim.examples;

 * Title:        CloudSim Toolkit

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.
 */
```

```java
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 Cloud

            // 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,
            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

        // 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
                                                // 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 VmAllocationPolicySimp
        } 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:

```
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!
```

RESULT:

Thus, the installation and configuration of CloudSim in Eclipse IDE has been successfully completed.