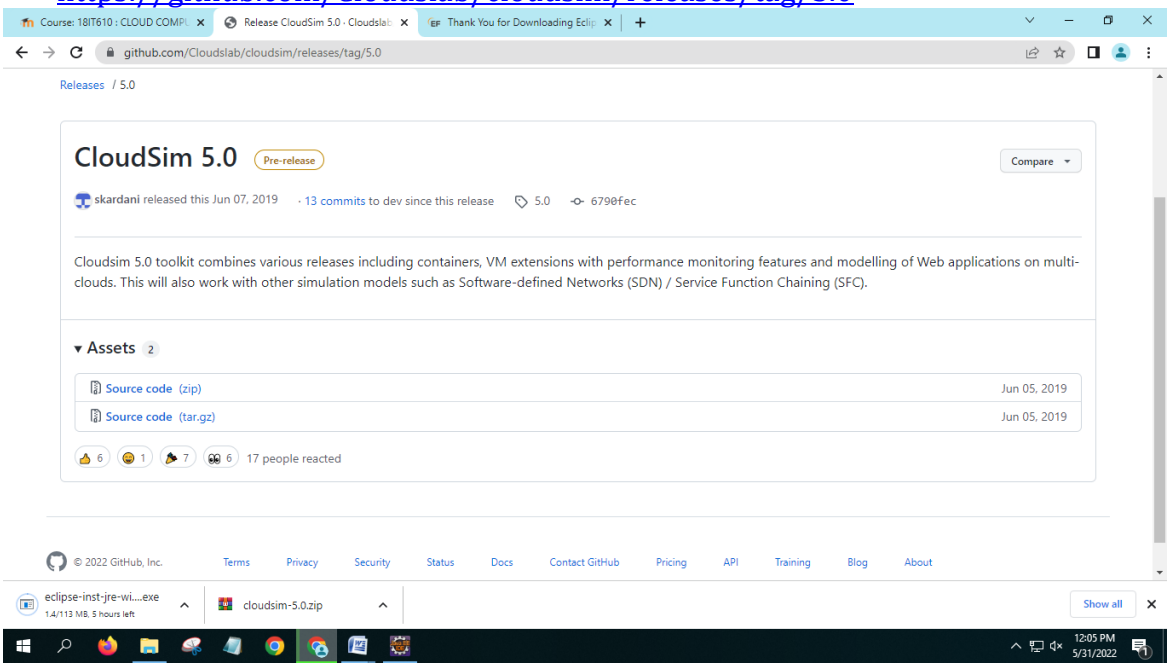
Ex.No.10	Simulation of cloud scenario using cloudsim

AIM:

To implement the simulation of cloud scenario using cloudsim.

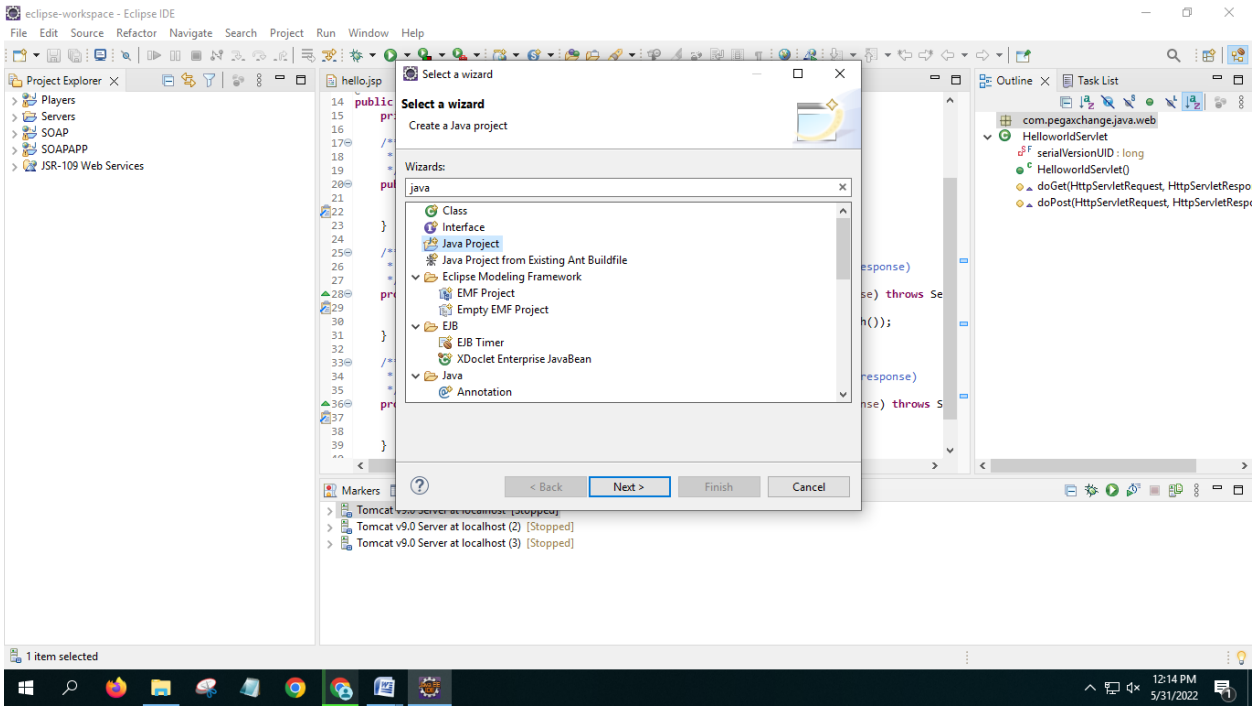
PROCEDURE:

1. Download Cloudsim 5.0 from https://github.com/Cloudslab/cloudsim/releases/tag/5.0

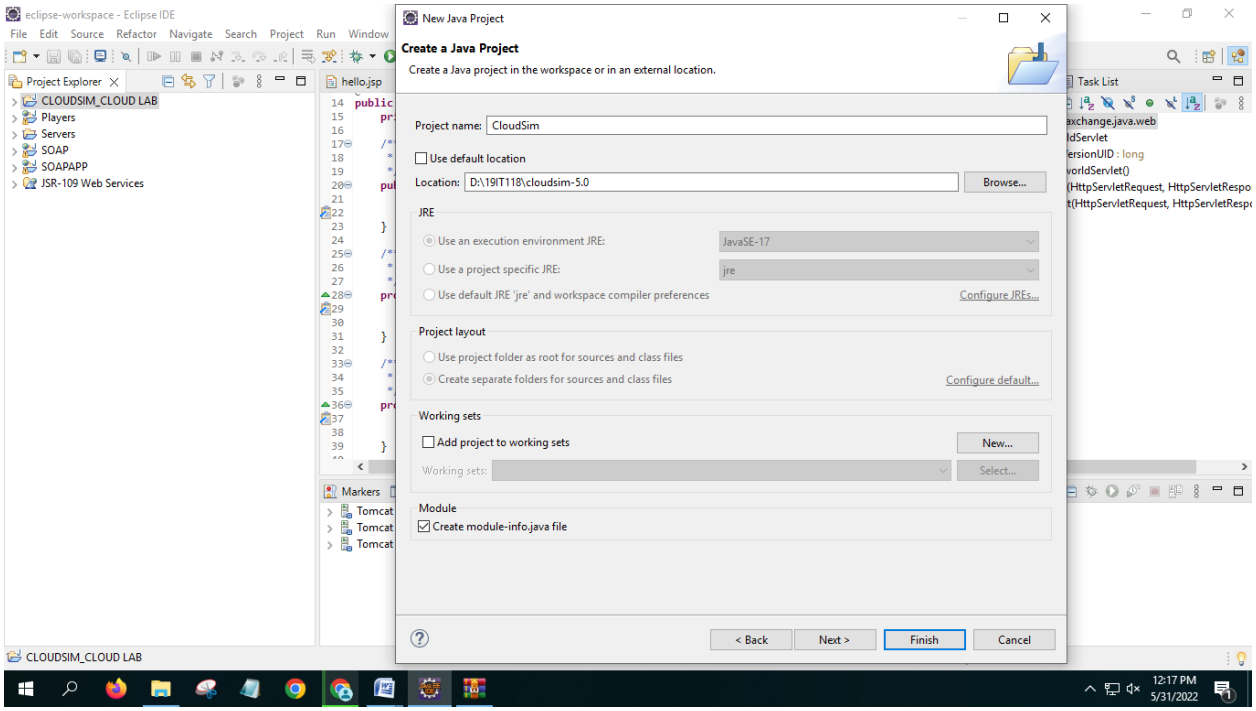


2. Add common math library https://commons.apache.org/proper/commons-math/download_math.cgi

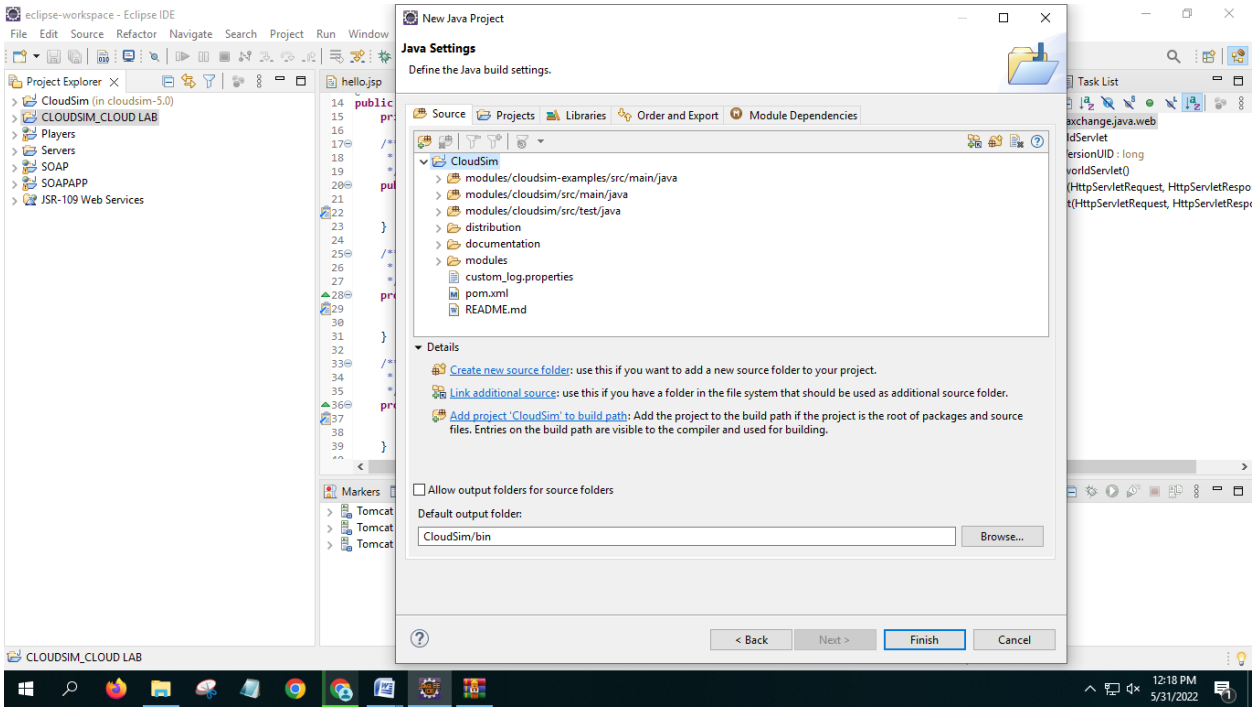
Open Eclipse. Select New > Project > Java Project



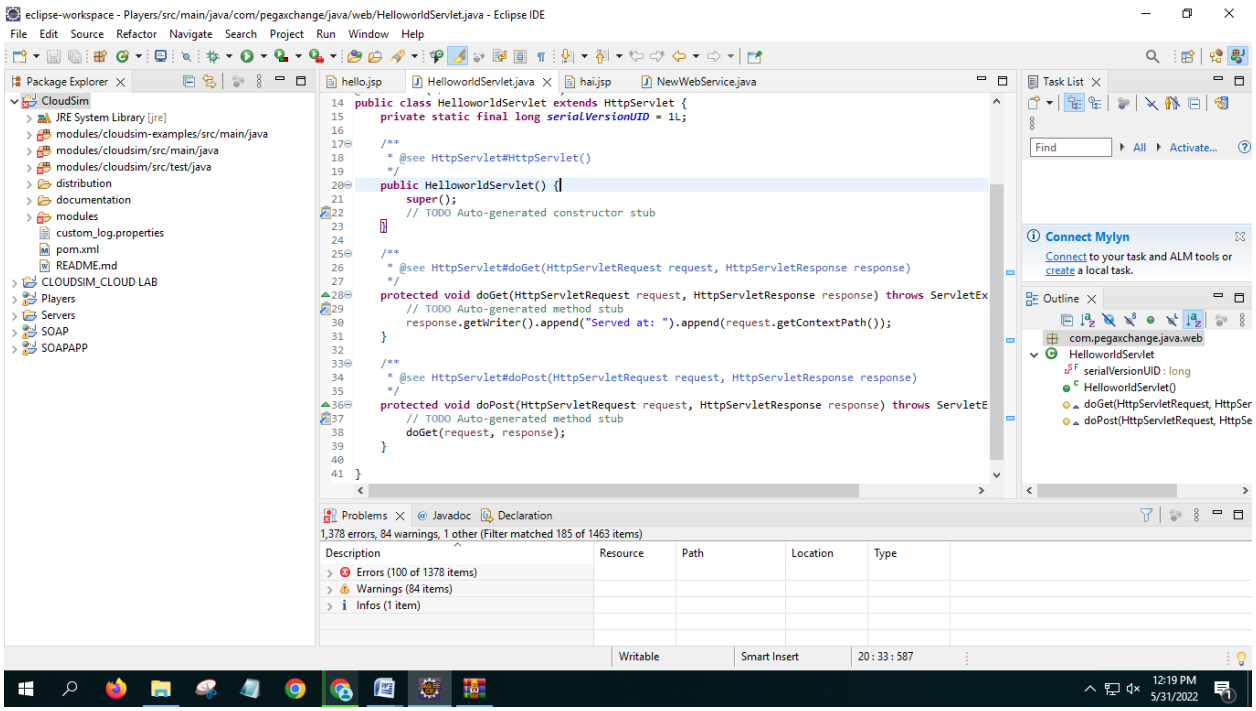
The project is named as Cloudsim and the location of the downloaded folder is specified.



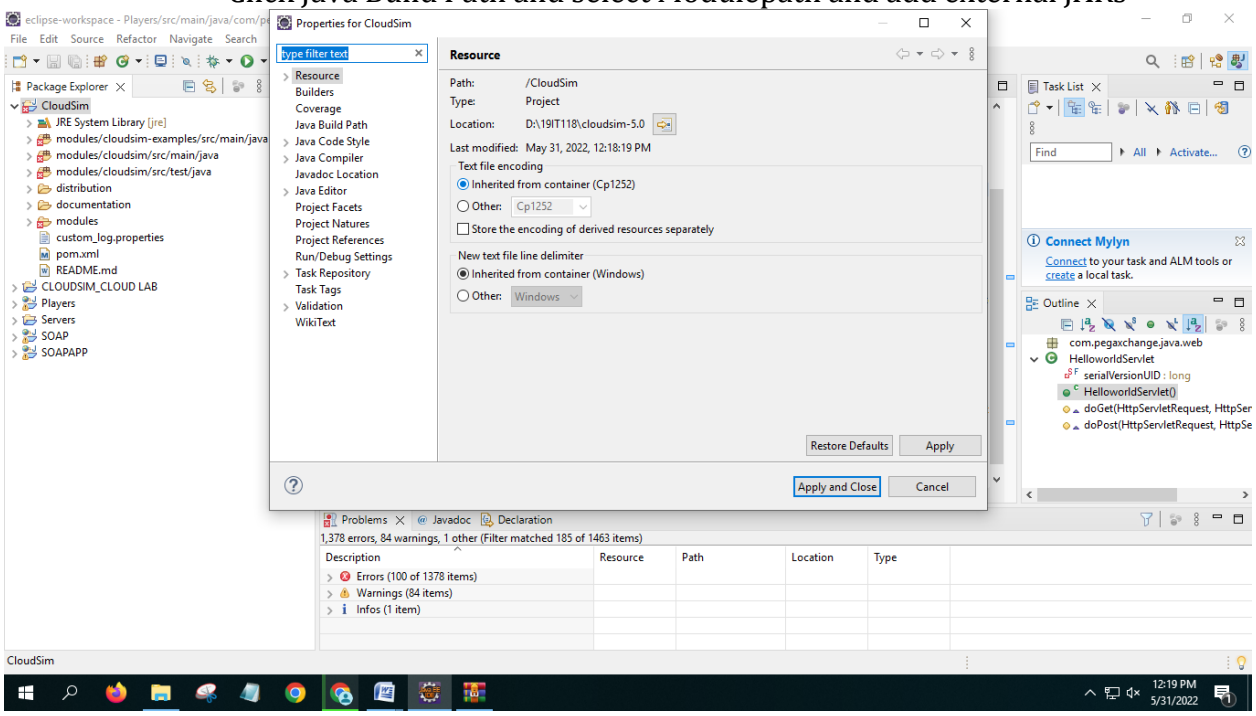
Once you click finish, the cloudsim project becomes viewable in your package explorer as shown below



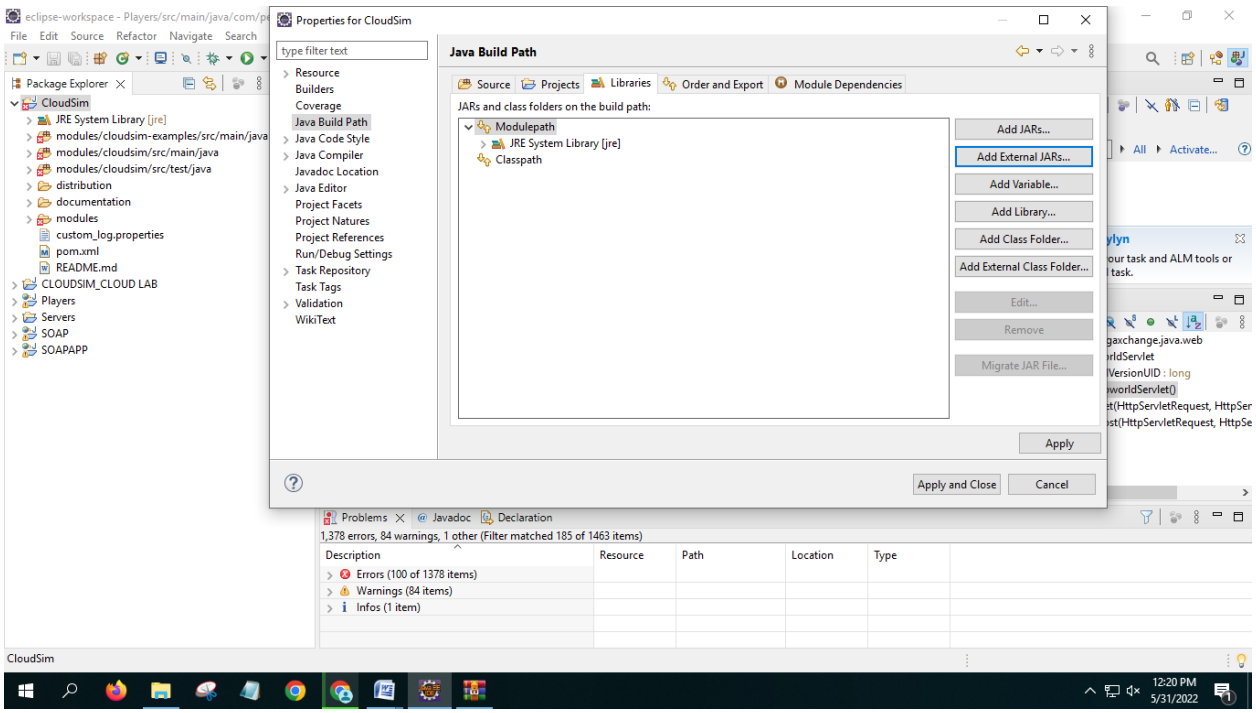
After Loading the code you will be able to see Cloudsim project in Package Explorer



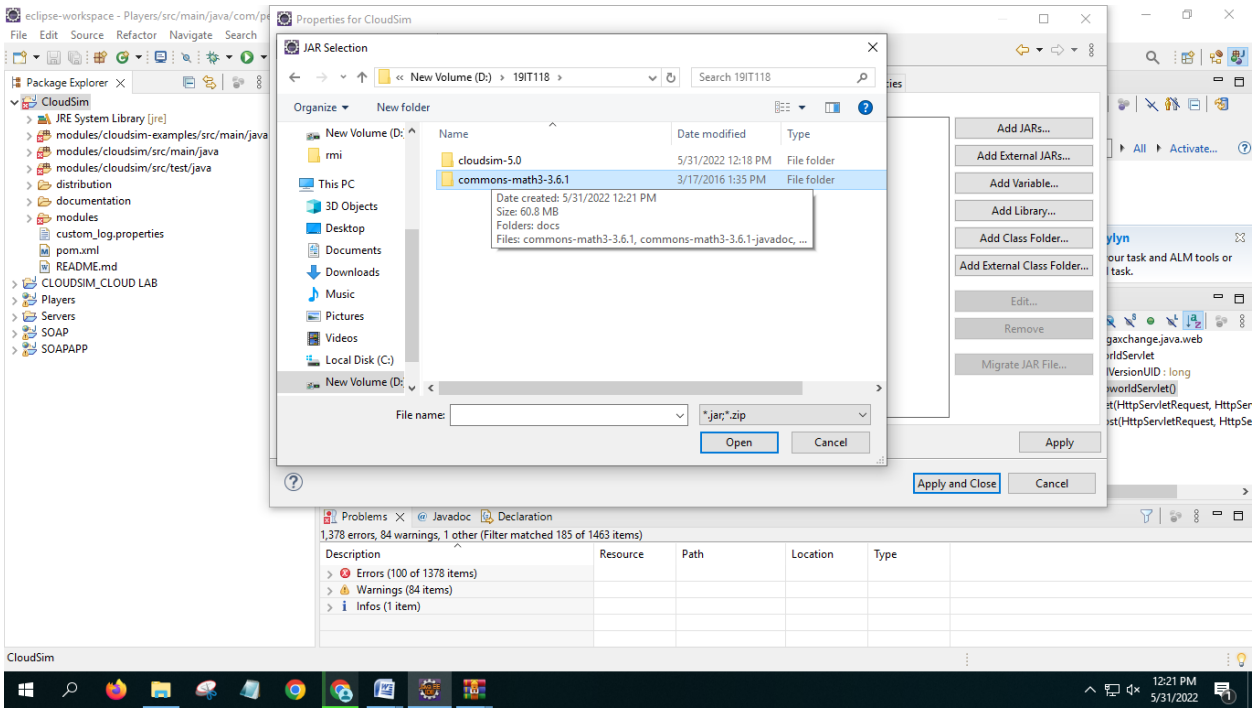
Click Java Build Path and select Modulepath and add external JARs



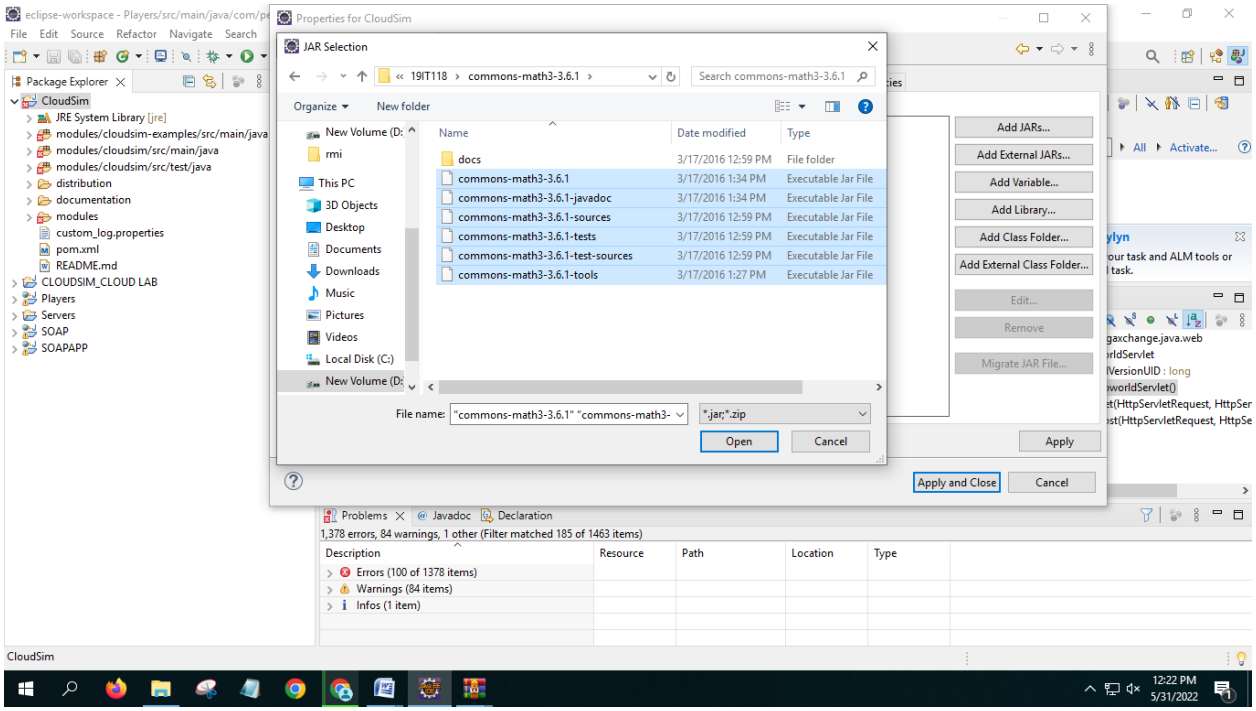
Here Click the Module Path and Select External JARS from options present at the right.



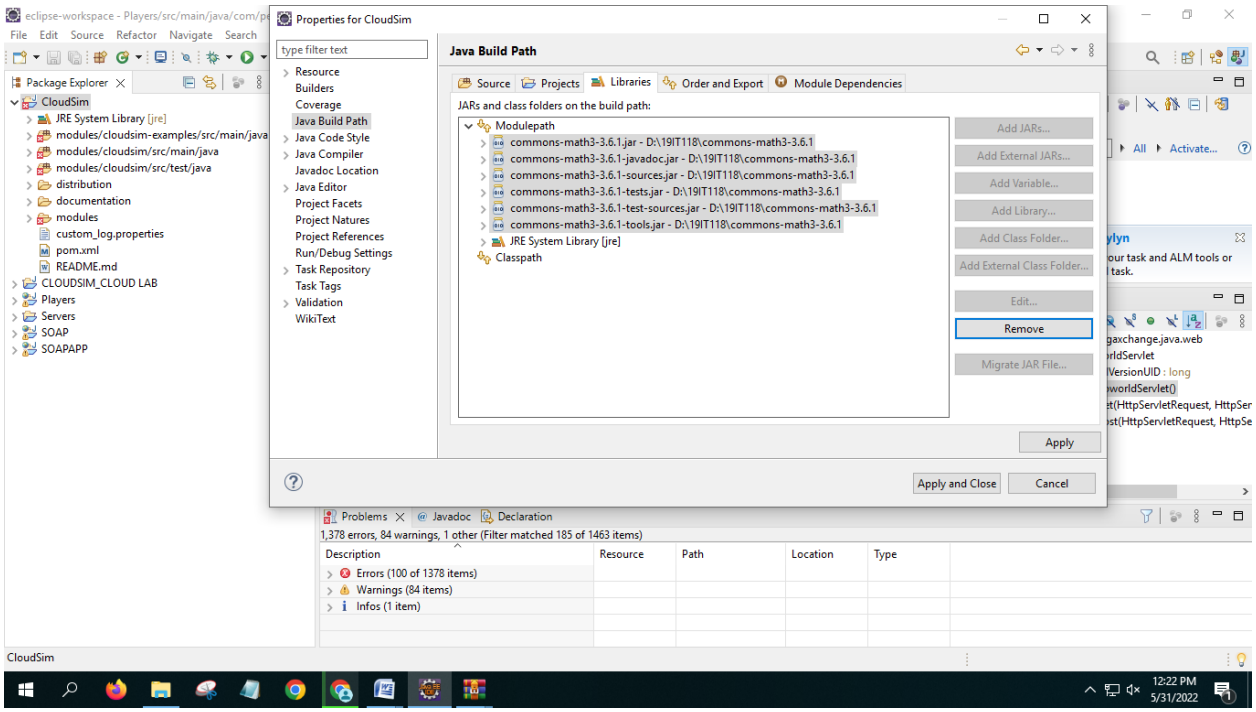
Select the Common math libraries folder that you have downloaded and extracted.



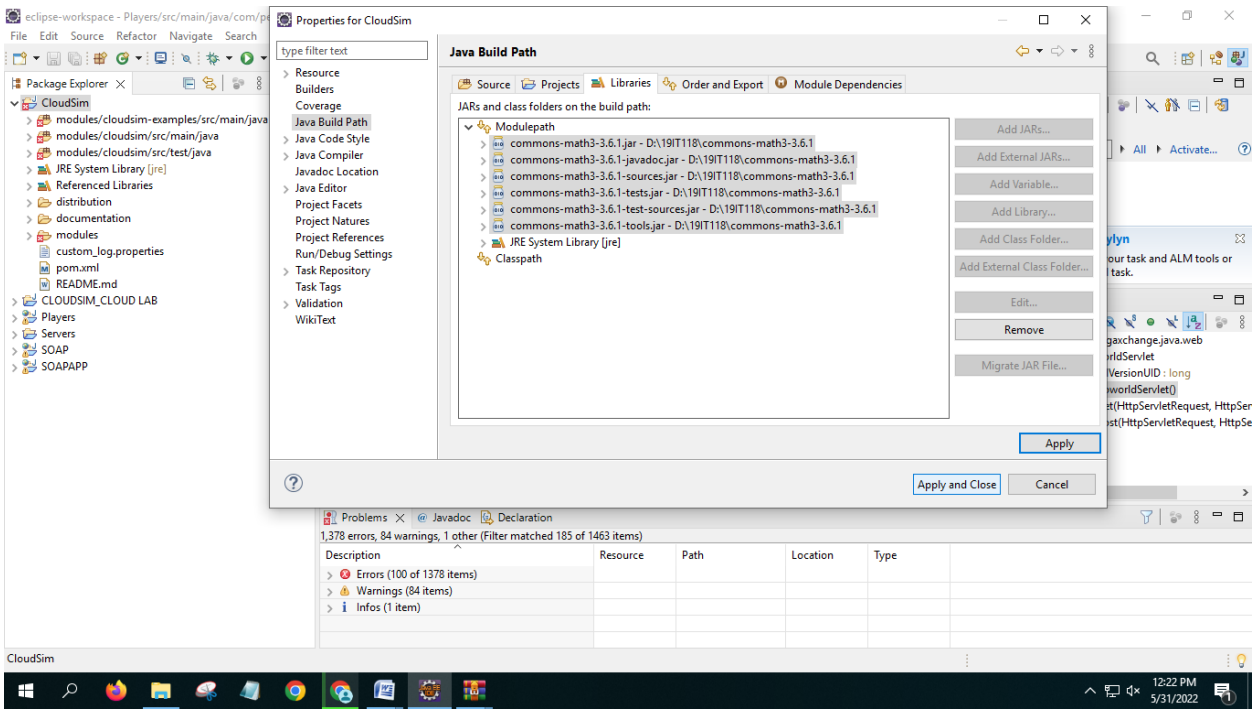
Within that folder select all the files of the format **.jar**



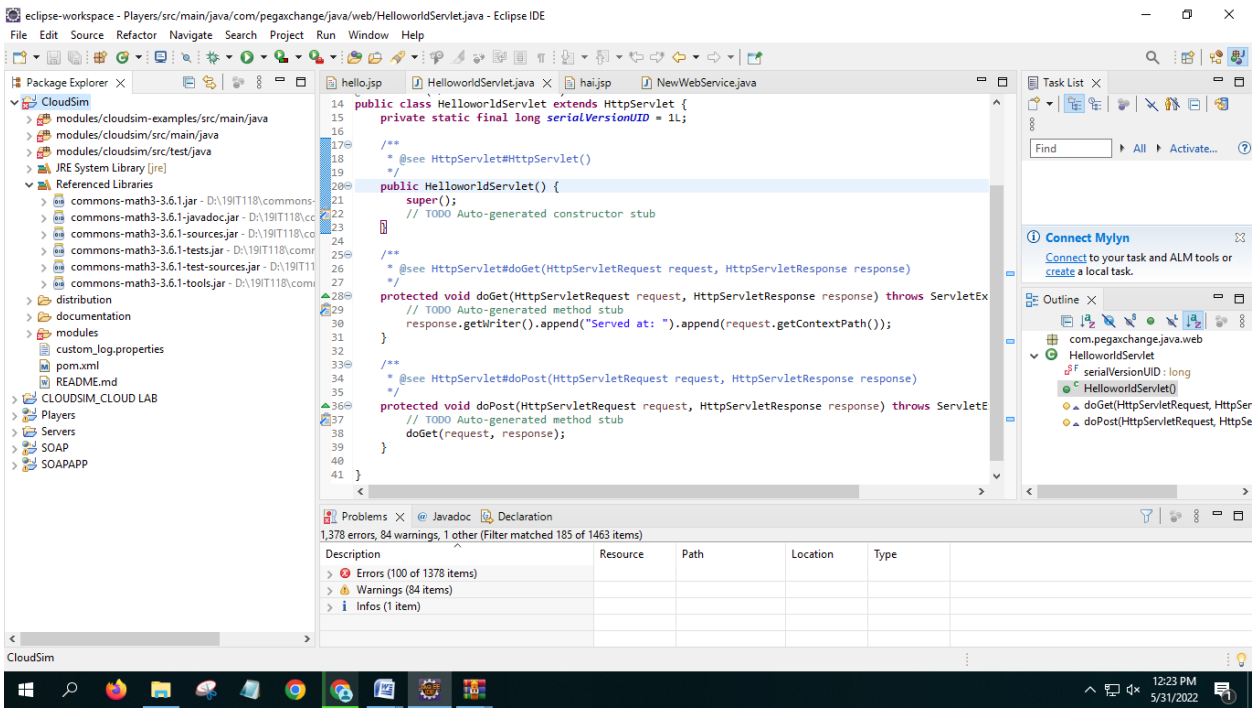
After selecting open, you will be able to see the files being added here.



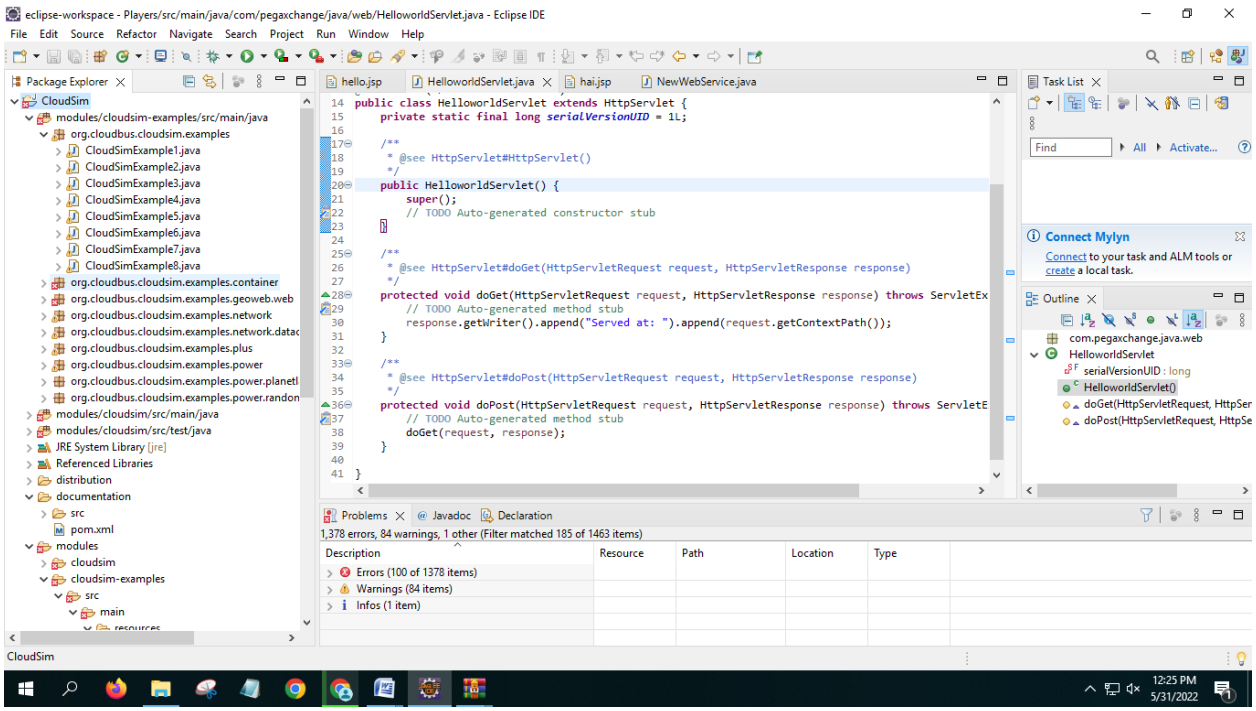
Click Apply and Apply and Close



After including the external jars You will be able to view them in the referenced libraries



Now select modules/clousim-examples/src/main/java. Here you will be able to view the .java files.

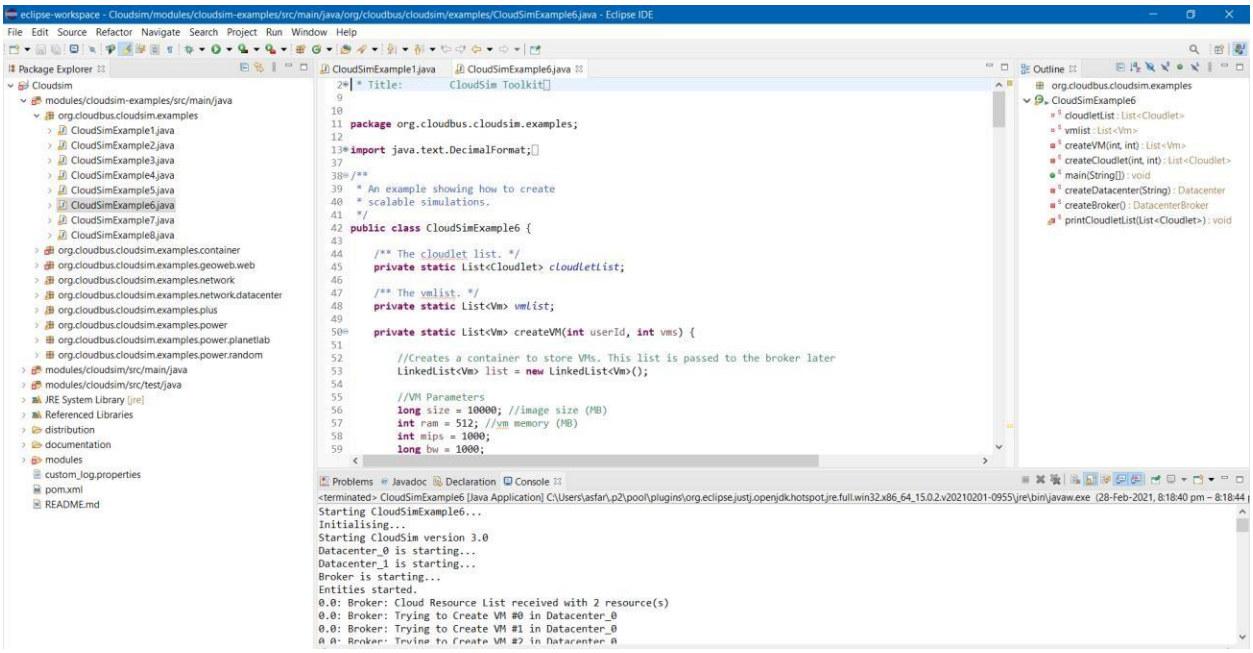


CLOUD SIM EXAMPLE 6:

An example showing how to create scalable simulations.It means varying numbersof cloudlets as well as varying numbers of VMs.

Description:

1. Set the Number of users for the current simulation
2. The createDatacenter() method initializes the various datacenter characteristics along with the host list.
3. The createBroker() method initializes the entity object from DatacenterBroker class
4. Create a Virtual Machines and also create cloudlets.
5. Invoke method to start and stop stimulation.
6. Finally, print the final status of the Simulation.



CODE WITH EXPLANATION:

```
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 CloudSimExample6 {
/** The cloudlet list. */
private static List<Cloudlet> cloudletList;
/** The vmlist. */
private static List<Vm> vmlist;
private static List<Vm> createVM(int userId, int vms) {
//Creates a container to store VMs. This list is passed to the broker later
LinkedList<Vm> list = new LinkedList<Vm>();
//VM Parameters
long size = 10000; //image size (MB)
int ram = 512; //vm memory (MB)
int mips = 1000;
long bw = 1000;
int pesNumber = 1; //number of cpus
```

```
String vmm = "Xen"; //VMM name

//create VMs

Vm[] vm = new Vm[vms];

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

vm[i] = new Vm(i, userId, mips, pesNumber, ram, bw, size, vmm, new
CloudletSchedulerTimeShared());

//for creating a VM with a space shared scheduling policy for cloudlets:

//vm[i] = Vm(i, userId, mips, pesNumber, ram, bw, size, priority, vmm, new
CloudletSchedulerSpaceShared());

list.add(vm[i]);}

return list;}

private static List<Cloudlet> createCloudlet(int userId, int cloudlets){

// Creates a container to store Cloudlets

LinkedList<Cloudlet> list = new LinkedList<Cloudlet>();

//cloudlet parameters

long length = 1000;

long fileSize = 300;

long outputSize = 300;

int pesNumber = 1;

UtilizationModel utilizationModel = new UtilizationModelFull();

Cloudlet[] cloudlet = new Cloudlet[cloudlets];

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

cloudlet[i] = new Cloudlet(i, length, pesNumber, fileSize, outputSize, utilizationModel,
utilizationModel, utilizationModel);

// setting the owner of these Cloudlets

cloudlet[i].setUserId(userId);
```

```
list.add(cloudlet[i]);}

return list;}

////////////////////////// STATIC METHODS ///////////////////////

/*** Creates main() to run this example*/

public static void main(String[] args) {

Log.printLine("Starting CloudSimExample6...");

try {

// First step: Initialize the CloudSim package. It should be called

// before creating any entities.

int num_user = 1;   // number of grid users

Calendar calendar = Calendar.getInstance();

boolean trace_flag = false;  // mean trace events

// Initialize the CloudSim library

CloudSim.init(num_user, calendar, trace_flag);

// Second step: Create Datacenters

//Datacenters are the resource providers in CloudSim. We need at list one of them to run a CloudSim simulation

@SuppressWarnings("unused")

Datacenter datacenter0 = createDatacenter("Datacenter_0");

@SuppressWarnings("unused")

Datacenter datacenter1 = createDatacenter("Datacenter_1");

//Third step: Create Broker

DatacenterBroker broker = createBroker();

int brokerId = broker.getId();
```

```
//Fourth step: Create VMs and Cloudlets and send them to broker
vmlist = createVM(brokerId,20); //creating 20 vms
cloudletList = createCloudlet(brokerId,40); // creating 40 cloudlets
broker.submitVmList(vmlist);
broker.submitCloudletList(cloudletList);
// Fifth step: Starts the simulation
CloudSim.startSimulation();
// Final step: Print results when simulation is over
List<Cloudlet> newList = broker.getCloudletReceivedList();
CloudSim.stopSimulation();
printCloudletList(newList);
Log.printLine("CloudSimExample6 finished!");}
catch (Exception e){
e.printStackTrace();
Log.printLine("The simulation has been terminated due to an unexpected error");}}
private static Datacenter createDatacenter(String name){
// Here are the steps needed to create a PowerDatacenter:
// 1. We need to create a list to store one or more Machines
List<Host> hostList = new ArrayList<Host>();
// 2. A Machine contains one or more PEs or CPUs/Cores. Therefore, should create a
list to store these PEs before creating a Machine.
List<Pe> peList1 = new ArrayList<Pe>();
 int mips = 1000;
// 3. Create PEs and add these into the list.
//for a quad-core machine, a list of 4 PEs is required:
```

```
peList1.add(new Pe(0, new PeProvisionerSimple(mips))); // need to store Pe id and MIPS Rating

peList1.add(new Pe(1, new PeProvisionerSimple(mips)));

peList1.add(new Pe(2, new PeProvisionerSimple(mips)));

peList1.add(new Pe(3, new PeProvisionerSimple(mips)));

//Another list, for a dual-core machine

List<Pe> peList2 = new ArrayList<Pe>();

peList2.add(new Pe(0, new PeProvisionerSimple(mips)));

peList2.add(new Pe(1, new PeProvisionerSimple(mips)));

//4. Create Hosts with its id and list of PEs and add them to the list of machines

int hostId=0;

int ram = 2048; //host memory (MB)

long storage = 1000000; //host storage

int bw = 10000;

hostList.add(

new Host(

hostId,

new RamProvisionerSimple(ram),

new BwProvisionerSimple(bw),

storage,

peList1,

new VmSchedulerTimeShared(peList1))); // This is our first machine

hostId++;

hostList.add(

new Host(hostId,
```

```
new RamProvisionerSimple(ram),
new BwProvisionerSimple(bw),
storage,
peList2,
new VmSchedulerTimeShared(peList2))); // Second machine
```

//To create a host with a space-shared allocation policy for PEs to VMs:

```
//hostList.add(
        //new Host(
        //                  hostId,
        //                  new CpuProvisionerSimple(peList1),
        //                  new RamProvisionerSimple(ram),
        //                  new BwProvisionerSimple(bw),
        //                  storage,
        //                  new VmSchedulerSpaceShared(peList1)
        //              )
        //      );
```

//To create a host with a oportunistic space-shared allocation policy for PEs to VMs:

```
//hostList.add(
        //              new Host(
        //                  hostId,
        //                  new CpuProvisionerSimple(peList1),
        //                  new RamProvisionerSimple(ram),
        //                  new BwProvisionerSimple(bw),
        //                  storage,
        //                  new VmSchedulerOportunisticSpaceShared(peList1)
```

```
//                  )
//       );
```

// 5. Create a DatacenterCharacteristics object that stores the properties of a data center: architecture, OS, list of Machines, allocation policy: time- or space-shared, time zone and its price (G$/Pe time unit).

```
String arch = "x86";      // system architecture
String os = "Linux";          // operating system
String vmm = "Xen";
double time_zone = 10.0;          // time zone this resource located
double cost = 3.0;              // the cost of using processing in this resource
double costPerMem = 0.05;  // the cost of using memory in this resource
double costPerStorage = 0.1;            // the cost of using storage in this resource
double costPerBw = 0.1;     // the cost of using bw in this resource
LinkedList<Storage> storageList = new LinkedList<Storage>(); //we are not adding SAN devices by now
DatacenterCharacteristics characteristics = new DatacenterCharacteristics(arch, os, vmm, hostList, time_zone, cost, costPerMem, costPerStorage, costPerBw);
```

// 6. Finally, we need to create a PowerDatacenter object.

```
Datacenter datacenter = null;
try {
datacenter = new Datacenter(name, characteristics, new VmAllocationPolicySimple(hostList), storageList, 0);} catch (Exception e) {
e.printStackTrace();
}
return datacenter;}
```

//We strongly encourage users to develop their own broker policies, to submit vms and cloudlets according to the specific rules of the simulated scenario

```
private static DatacenterBroker createBroker(){

DatacenterBroker broker = null;

try {

broker = new DatacenterBroker("Broker");

} catch (Exception e) {e.printStackTrace(); return null;}

return broker;}
```

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

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

int size = list.size();

Cloudlet cloudlet;

String indent = "    ";

Log.printLine();

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

Log.printLine("Cloudlet ID" + indent + "STATUS" + indent + "Data center ID" + indent + "VM
ID" + indent + indent + "Time" + indent + "Start Time" + indent + "Finish Time");

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

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

cloudlet = list.get(i);

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

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

Log.print("SUCCESS");

Log.printLine( indent + indent + cloudlet.getResourceId() + indent + indent + indent +
cloudlet.getVmId() + indent + indent + indent + dft.format(cloudlet.getActualCPUTime()) +
indent + indent + dft.format(cloudlet.getExecStartTime())+ indent + indent + indent +
dft.format(cloudlet.getFinishTime()));}}}}
```