**Step 1: Set Up Your Environment**

First, let's set up the development environment:

1. **Install Java JDK** (if not already installed)
   * Download and install JDK 8 or newer from Oracle's website
   * Set up JAVA\_HOME environment variable
2. **Install Eclipse IDE**
   * Download Eclipse IDE for Java Developers from eclipse.org
   * Install and launch Eclipse
   * Choose a workspace location when prompted
3. **Create a New Java Project**
   * Go to File > New > Java Project
   * Name your project (e.g., "CloudSimExample")
   * Click "Finish"

**Step 2: Add CloudSim Library**

1. **Download CloudSim**
   * Download the CloudSim toolkit from the official GitHub repository or website
   * Extract the ZIP file to a location on your computer
2. **Add CloudSim JAR Files to Your Project**
   * Right-click on your project in Eclipse
   * Select Properties > Java Build Path > Libraries tab
   * Click "Add External JARs"
   * Navigate to the extracted CloudSim folder and select all JAR files in the "jars" directory
   * Click "Apply and Close"

**Step 3: Create a Simple CloudSim Program**

Let's create a basic simulation that:

1. Creates a datacenter
2. Creates a broker
3. Creates a VM and a cloudlet (task)
4. Submits the VM and cloudlet to the broker
5. Starts the simulation

Here's how to implement it:

1. **Create a new Java class**
   * Right-click on your project > New > Class
   * Name it "CloudSimExample"
   * Check the option to create a main method
   * Click "Finish"
2. **Insert the following code:**

java

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 CloudSimExample {

*/\*\* The cloudlet list. \*/*

private static List<Cloudlet> cloudletList;

*/\*\* The vm list. \*/*

private static List<Vm> vmList;

public static void main(String[] args) {

try {

*// Number of users (we set it to 1)*

int num\_user = 1;

*// Calendar instance*

Calendar calendar = Calendar.getInstance();

*// Indicate if we should trace events*

boolean trace\_flag = false;

*// Initialize CloudSim*

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

*// Create Datacenter*

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

*// Create Broker*

DatacenterBroker broker = createBroker();

int brokerId = broker.getId();

*// Create VM and Cloudlet Lists*

vmList = new ArrayList<Vm>();

cloudletList = new ArrayList<Cloudlet>();

*// VM Parameters*

int vmid = 0;

int mips = 1000;

long size = 10000; *// image size (MB)*

int ram = 512; *// VM memory (MB)*

long bw = 1000; *// bandwidth*

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 VM to list*

vmList.add(vm);

*// Submit VM list to broker*

broker.submitVmList(vmList);

*// Cloudlet parameters*

int id = 0;

long length = 400000; *// 400,000 Million Instructions (MI)*

long fileSize = 300; *// 300 KB*

long outputSize = 300; *// 300 KB*

UtilizationModel utilizationModel = new UtilizationModelFull();

*// Create a cloudlet*

Cloudlet cloudlet = new Cloudlet(id, length, pesNumber, fileSize, outputSize, utilizationModel, utilizationModel, utilizationModel);

cloudlet.setUserId(brokerId);

cloudlet.setVmId(vmid);

*// Add cloudlet to list*

cloudletList.add(cloudlet);

*// Submit cloudlet list to broker*

broker.submitCloudletList(cloudletList);

*// Start the simulation*

CloudSim.startSimulation();

*// Stop the simulation*

CloudSim.stopSimulation();

*// Print results*

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

printCloudletList(newList);

Log.printLine("CloudSimExample finished!");

} catch (Exception e) {

e.printStackTrace();

Log.printLine("The simulation has been terminated due to an unexpected error");

}

}

*/\*\**

\* Creates the datacenter.

\*

*\* @param name the name*

*\* @return the datacenter*

*\*/*

private static Datacenter createDatacenter(String name) {

*// Create a list to store hosts*

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

*// Host parameters*

int hostId = 0;

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

long storage = 1000000; *// host storage (MB)*

int bw = 10000; *// bandwidth*

*// PE (Processing Element) parameters*

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

int mips = 1000; *// MIPS rating*

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

*// Create Host with its characteristics*

Host host = new Host(

hostId,

new RamProvisionerSimple(ram),

new BwProvisionerSimple(bw),

storage,

peList,

new VmSchedulerTimeShared(peList)

);

*// Add host to the list*

hostList.add(host);

*// Datacenter characteristics*

String arch = "x86"; *// architecture*

String os = "Linux"; *// OS*

String vmm = "Xen"; *// VMM*

double time\_zone = 10.0; *// time zone (GMT)*

double cost = 3.0; *// cost of using processing*

double costPerMem = 0.05; *// cost of using memory*

double costPerStorage = 0.001; *// cost of using storage*

double costPerBw = 0.0; *// cost of using bw*

LinkedList<Storage> storageList = new LinkedList<Storage>();

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;

}

*/\*\**

\* 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()));

}

}

}

}

**Step 4: Run the Simulation**

1. **Run the program**
   * Right-click on your CloudSimExample.java file
   * Select Run As > Java Application
2. **Examine the output**
   * You should see the simulation results in the console
   * The output will show details about the cloudlet execution, including its ID, status, datacenter ID, VM ID, execution time, start time, and finish time