**Assignment No-3**

**Aim:** Simulate a cloud scenario using CloudSim and run a scheduling algorithm that is not present in CloudSim.

# Objective:

* To Simulate a cloud scenario using CloudSim.
* To develop and run a scheduling algorithm that is not present in CloudSim.
* Analyze various cloud programming models and apply them to solve problems on the cloud.

# Theory:

Cloud Computing has completely transformed how modern-day applications are developed and maintained with high scalability and low latency.

CloudSim is an open-source framework, which is used to simulate cloud computing infrastructure and services. It is developed by the CLOUDS Lab organization and is written entirely in Java. It is used for modelling and simulating a cloud computing environment as a means for evaluating a hypothesis prior to software development in order to reproduce tests and results. CloudSim simulation toolkit that supports the modeling and simulation of the core functionality of cloud, like job/task queue, processing of events, creation of cloud entities(datacenter, datacenter brokers, etc), communication between different entities, implementation of broker policies, etc. This toolkit allows to:

* Test application services in a repeatable and controllable environment.
* Tune the system bottlenecks before deploying apps in an actual cloud.
* Experiment with different workload mix and resource performance scenarios on simulated infrastructure for developing and testing adaptive application provisioning techniques.

**Task1: Download and install Java installer**

You need to first install Java Platform, Standard Edition Development Kit (JDK). The JDK is a development environment for building applications and components through Java programming language. The JDK includes tools for developing and testing programs written in the Java programming language and running on the Java platform.

# Pre-­Requisites: Java SE Development Kit 18.0.1

# JRE Installer

# 

**Step 1: Download and install JDK**

Goto JDK (or Java SE) download site @ <https://www.oracle.com/java/technologies/downloads/#jdk18-windows>

1. Under "Java SE Development Kit 18.0.1{x} downloads".
2. Select "Windows" ⇒ Download the "x64 Installer" (e.g., "https://download.oracle.com/java/18/latest/jdk-18\_windows-x64\_bin.exe.sha256 - about 153.24MB).

Step 2: Running the JDK installer

You must have administrator privilage to install the JDK on Microsoft Windows.

**Step 2: To run the JDK installer**

1. Start the JDK 10 installer by double-clicking the installer's icon or file name in the download location.
2. Follow the instructions provided by the Installation wizard.

The JDK includes the JavaFX SDK, a private JRE, and the Java Mission Control tools suite. The installer integrates the JavaFX SDK into the JDK installation directory.

1. After the installation is complete, delete the downloaded file to recover the disk space

**Step 3: Setting the PATH Environment Variable**

1. It is useful to set the PATH variable permanently for JDK 10 so that it is persistent after rebooting.
2. If you do not set the PATH variable, then you must specify the full path to the executable file every time that you run it. For example:
3. C:\> "C:\Program Files\Java\jdk-10\bin\javac" MyClass.java
4. To set the PATH variable permanently, add the full path of the jdk-10\bin directory to the PATH variable. Typically, the full path is:
5. C:\Program Files\Java\jdk-10\bin
6. To set the PATH variable on Microsoft Windows:
7. Select **Control Panel** and then **System**.
8. Click **Advanced** and then **Environment Variables**.
9. Add the location of the bin folder of the JDK installation to the PATH variable in **System Variables**.
10. The following is a typical value for the PATH variable:
11. C:\WINDOWS\system32;C:\WINDOWS;"C:\Program Files\Java\jdk-10\bin"

**Step 4: Downloading the JRE Installer**

The JRE Installer is located on the Java SE Runtime Environment 10 Downloads page.

1. In a browser, go to the Java SE Runtime Environment 10 Downloads page.

The following JRE installers are available for you to download:

* + Windows :   jre-10.0.2\_windows-x64\_bin.exe

1. Download the JRE installer according to your requirement.
2. Click **Accept License Agreement**, and then, under the **Download** menu, click the link that corresponds to the installer for your version of Windows.
3. Note the file size specified on the download page and, after the download has completed, verify that you have downloaded the complete file.
4. Running the JRE Installer
5. You must have Administrative privileges in order to install the JRE on Microsoft Windows.
6. To run the JRE installer:
7. Start the JRE 10 Installer by double-clicking the installer's icon or file name in the download location.
8. Follow the instructions provided by the Installation wizard.
9. The installer notifies you if Java content is disabled in web browsers and provides instructions for enabling it. If you previously chose to hide some of the security prompts for applets and Java Web Start applications, then the installer provides an option for restoring the prompts.
10. After the installation is complete, delete the downloaded file to recover disk space.

# Task 2: Download and Install Install Eclipse:

# Eclipse is an *open-source* Integrated Development Environment (IDE) supported by IBM. Eclipse is popular for Java application development (Java SE and Java EE) and Android apps. It also supports C/C++, PHP, Python, Perl, and other web project developments via extensible plug-ins. Eclipse is cross-platform and runs under Windows, Linux and macOS.

# Download Eclipse from <https://www.eclipse.org/downloads/packages/>. Choose "Eclipse IDE for Java Developers" and "Windows x86\_64" (e.g., "eclipse-java-2021-12-R-win32-x86\_64.zip" - about 313MB) ⇒ Download.

# Step 2: Unzip

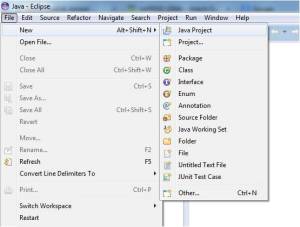
# To install Eclipse, simply unzip the downloaded file into a directory of your choice (e.g., "c:\myProject").

**TASK 3: Installation and running the CloudSim Toolkit**

1. Go to website: http://www.cloudbus.org/cloudsim/
2. Cloudsim project source code, as well as compiled jars files, are published through the [GitHub project page](https://github.com/Cloudslab/cloudsim): <https://github.com/Cloudslab/cloudsim>.
3. By default, the project page displays the source code(based on [maven](https://maven.apache.org/)build tool) of the current release, Release Version 3.0
4. Cloudsim setup is very easy, You just need to unpack the CloudSim file to install.
5. For this you may follow the following link: [cloudsim-setup-using-eclipse/](https://www.cloudsimtutorials.online/cloudsim-setup-using-eclipse/). This link describes in detail all the steps that are required to successfully configure the Cloudsim 3.0.3 version. The cloudsim 3.0.3 version is best to start with that once you understand the basic working and architecture then you can move to any latest version.

**TASK 4:** [**STEP BY STEP INSTALLATION OF CLOUD SIM INTO ECLIPSE**](https://researchcloudcomputing.wordpress.com/2015/01/19/step-by-step-installation-of-cloudsim-into-eclipse/)

1. Open up Eclipse and go to Menu Section, then click File, keep on clicking New and finally select java project. It is shown as in the Figure1

[](https://researchcloudcomputing.files.wordpress.com/2015/01/fig11.jpg)

Open eclipse and select java project

Open up Eclipse and Click on java project

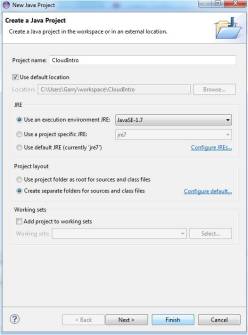
2. A new window will get open. Put a foot on to the following steps:-

2.1.Enter project name. (I have named it as CloudIntro)

2.2 In the next line you will see the path where your project will be created as it as shown in the Figure2.

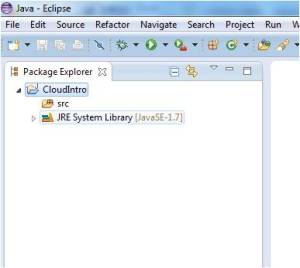
2.3 Next You need to select the JRE environment.

2.4 Finally Click Finish

[](https://researchcloudcomputing.files.wordpress.com/2015/01/fig2.jpg)

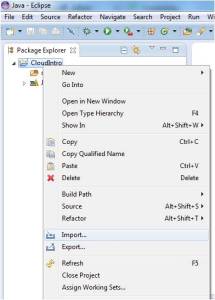
Give project Name and select run time environment and Finish

1. Once you hit finish. An empty project named CloudIntro will be created in the project List as shown in the Figure3.

[](https://researchcloudcomputing.files.wordpress.com/2015/01/fig3.jpg)

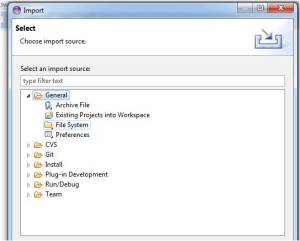
Project Folder Location

4. Next step is to go the project CloudIntro, right click on it. Click Import as shown in the   Figure4.

[](https://researchcloudcomputing.files.wordpress.com/2015/01/fig4.jpg)

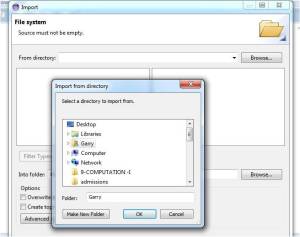
Import cloud sim tool files and subsequent folders

5. A new window will get open, now click File System as demonstrated in the Figure5.

[](https://researchcloudcomputing.files.wordpress.com/2015/01/fig5.jpg)

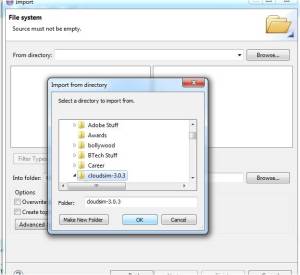
Next to select is File System

6. Next Step is to go to the directory where you have extracted your cloud sim tool. Figure6 is shown to guide you to get into the directory where your cloudsim folder is located.

[](https://researchcloudcomputing.files.wordpress.com/2015/01/fig6.jpg)

Go to Directory to select Cloudsim (My system searching)

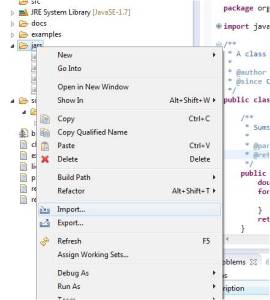
7. Select the cloudsim and click Finish as shown in the Figure7.

[](https://researchcloudcomputing.files.wordpress.com/2015/01/fig7.jpg)

Select Cloudsim and Hit finish

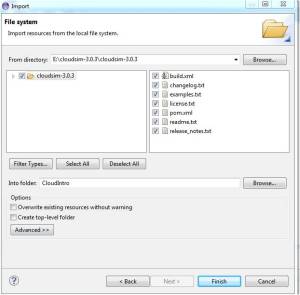
8. Now go to the link[**http://commons.apache.org/proper/commonsmath/download\_math.cgi**](http://commons.apache.org/proper/commonsmath/download_math.cgi)**. Download the file named as “**commons-math3-3.4.1-bin.zip”. Unzip this file. We need jar files for math functions.

9. Now go to the left side of the eclipse tool in the project bar. Go to jar and right click on it. Click import as shown in the Figure8.

[](https://researchcloudcomputing.files.wordpress.com/2015/01/fig10.jpg)

Import jar files for math calculations

10. Now go to the folder where you have placed the downloaded and extracted file as described by point 8. Then all you have to do is select that jar file and hit finish as shown by the Figure9**.**

[](https://researchcloudcomputing.files.wordpress.com/2015/01/fig9.jpg)

Import only jar

11. Finally the cloud sim is installed into your Eclipse environment.

**TASK 5: Implementing cloudsim simulation scenario**

Once you have completed your installation/setup and understand the basic working of the cloudsim, the next step is to implement your own custom scenario. Any simulation will go through the following steps:

* The main() method is the pointer from where the execution of this example starts as follows:

public static void main(String[] args)

* There are eleven steps that are followed in each example with some variation in them, specified as follows:
* Set the Number of users for the current simulation. This user count is directly proportional to a number of brokers in the current simulation.

int num\_user = 1; // number of cloud users Calendar calendar = Calendar.getInstance();

boolean trace\_flag = false;

* Initialize the CloudSim with the current clock time and this will also initialize the core CloudInformationService entity.

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

* Create Datacenter(s) as Datacenters are the resource providers in CloudSim. We need at list one of them to run a CloudSim simulation.

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

* The createDatacenter() method itself initializes the various datacenter characteristics along with the host list. This is the most important entity without this there is no way the simulation of hosting the virtual machine is applicable.

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)));

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)

) );

String arch = "x86";

String os = "Linux";

String vmm = "Xen";

double time\_zone = 10.0;

double cost = 3.0;

double costPerMem = 0.05;

double costPerStorage = 0.001;

double costPerBw = 0.0;

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; }

* Create a Datacenter Broker to simulate the user workload scheduling as well as virtual machine allocation and placements.

DatacenterBroker broker = createBroker();

int brokerId = broker.getId();

* Where the createBroker() method initializes the entity object from DatacenterBroker class

private static DatacenterBroker createBroker() {

DatacenterBroker broker = null;

try { broker = new DatacenterBroker("Broker");

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

return null; }

return broker; }

* Create one/more virtual machine and submit to the broker for further submitting it to the respective DataCenters for its placement and execution management during the simulation run.

vmlist = new ArrayList<Vm>();

int vmid = 0;

int mips = 1000;

long size = 10000;

int ram = 512;

long bw = 1000;

int pesNumber = 1;

String vmm = "Xen";

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

vmlist.add(vm);

* Submit Virtual Machine to Datacenter broker.

broker.submitVmList(vmlist);

* Create one/more Cloudlet Create Cloudlet(s) by specifying their characteristics.

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);

* Submit the cloudlet list to datacenter broker for further task scheduling on the active virtual machines for its processing during the simulation run.

broker.submitCloudletList(cloudletList);

* Starts the simulation, this will initiate all the entities and components created above and put them into execution for supporting various simulation operations.
* Stop the simulation, concludes simulation and flush all the entities & components before the exit of a simulation run.
* Print results when the simulation is over, where you will be able to display which cloudlet executed on which virtual machine along with how much time it spent in execution, its start time as well as its finish time.

### TASK 6: Implementing scheduling algorithm that is not present in CloudSim.

1. Install any of the IDE for running JAVA applications (eclipse recommended)
2. Install JDK and JRE for the same (Explained in TASK 1)
3. Add the jdk\bin path to the environment variables Open environment variables window, add the following to the path variable

Do include your bin path wherever you have installed JDK

For example:

> C:\Program Files\Java\jdk-14.0.1\bin

1. Open eclipse in your confined workspace
2. Click on new and open a new JAVA Project, give it a name
3. Create a package inside he src folder.
4. Dump in all the files [here](https://github.com/suyash-more/Cloud-Computing-Projects/tree/master/Scheduling-Algorithm-in-CloudSim/src) inside the package.
5. Now right click on the project and choose configure build path.
6. Click on the libraries section and add external jars
7. Extract the Cloudsim.tar file you downloaded
8. Now import the jar files in that Cloudsim.tar into the external jars.
9. Do remeber to change the name of the package in all the source files.
10. Now right click on the project and run the project as JAVA Application.
11. Select the SJF\_Scheduler.java file present in the list

**Conclusion:** Thus, we have studied and understood the process to simulate a cloud scenario and run a scheduling algorithm using CloudSim.