Ex.No – 05 **Roll no:**210701280

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 the path for Java in Environment Variables.
2. Download Cloud Sim and Additional JAR file:
   a. Download CloudSim 3.0.3
   b. Download common math 3 JAR file
3. Eclipse IDE Installation:
   a. Download the correct version of Eclipse IDE for your system.
   b. Install Eclipse IDE.
4. Run Cloud Sim in Eclipse:
   a. Put the common math 3 JAR file in the JAR folder of CloudSim.
   b. Build a new java project with CloudSim folder.

CODE:

```
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 {
public static void main(String[] args) {
Log.printLine("Starting CloudSimExample1...");
try {
int num_user = 1;
```

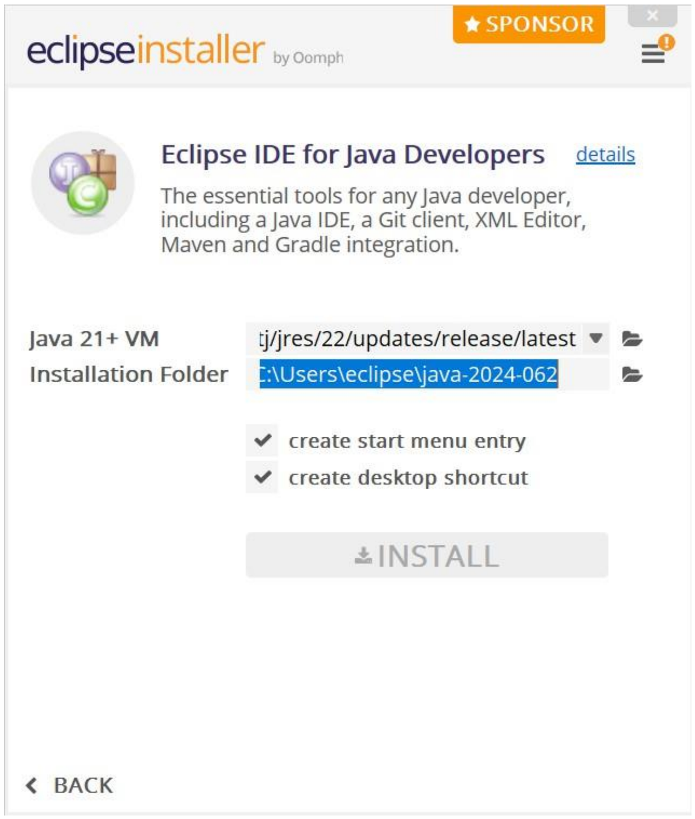
```
Calendar calendar = Calendar.getInstance();
boolean trace_flag = false;
CloudSim.init(num_user, calendar, trace_flag);
Datacenter datacenter0 =
createDatacenter("Datacenter_0");
DatacenterBroker broker = createBroker();
int brokerId = broker.getId();
vmlist = new ArrayList<Vm>();int vmid = 0;
int mips = 1000;
long size = 10000;
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);
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!");
3 catch (Exception e) {
e.printStackTrace();
Log.printLine("Unwanted errors happen");
3
3
private static Datacenter createDatacenter(String
name) {// Create a list to store our machine
List<Host> hostList = new ArrayList<Host>();
// 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;
// Create PEs and add these into a list.
peList.add(new Pe(0, new
PeProvisionerSimple(mips))); // need to store Pe id and
MIPS Rating
// 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
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;
double costPerMem = 0.05;
double costPerStorage = 0.001
double costPerBw = 0.0; // the cost of using bw in
this resource
LinkedList<Storage> storageList = newLinkedList<Storage>();
DatacenterCharacteristics characteristics = new
DatacenterCharacteristics(
arch, os, vmm, hostList, time_zone, cost, costPerMem,
costPerStorage, costPerBw);
// Finally, create a Datacenter object.
Datacenter datacenter = null;
try {
datacenter = new Datacenter(name,
characteristics, new VmAllocationPolicySimple(hostList),
storageList, 0);
3 catch (Exception e) {
e.printStackTrace();
3
return datacenter;
3
/**
* Creates the broker.
```

```
*
* @return the datacenter broker
*/
private static DatacenterBroker createBroker() {
DatacenterBroker broker = null;
try {
broker = new DatacenterBroker("Broker");
3 catch (Exception e) {
e.printStackTrace();
return null;
3
return broker;
3
list list of Cloudlets
*/
list) {
private static void printCloudletList(List<Cloudlet>
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()));
3
3
3
}
```

OUTPUT:



Eclipse IDE for Java Developers

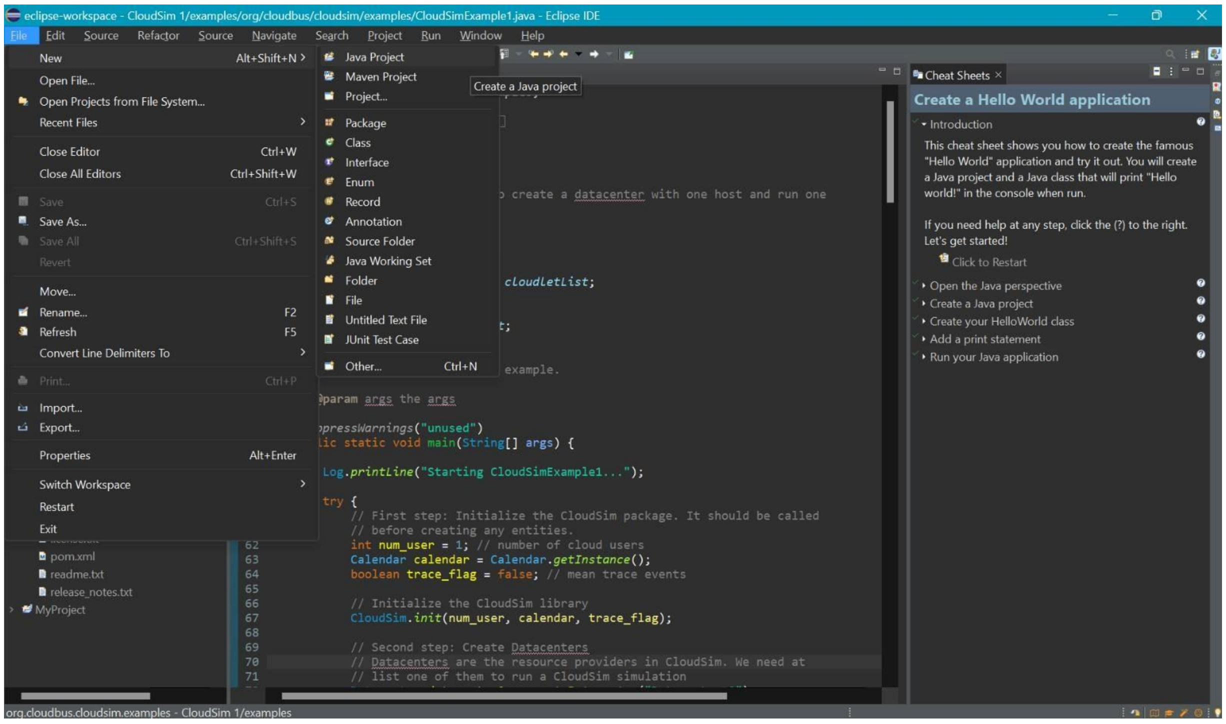
The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Maven and Gradle integration.

Java 21+ VM: tj/jres/22/updates/release/latest

Installation Folder: C:\Users\eclipse\java-2024-062

✓ create start menu entry

✓ create desktop shortcut



```
o create a datacenter with one host and run one

 cloudletList;

t;

 example.

@param args the args

ppressWarnings("unused")
lic 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
```

Create a Hello World application

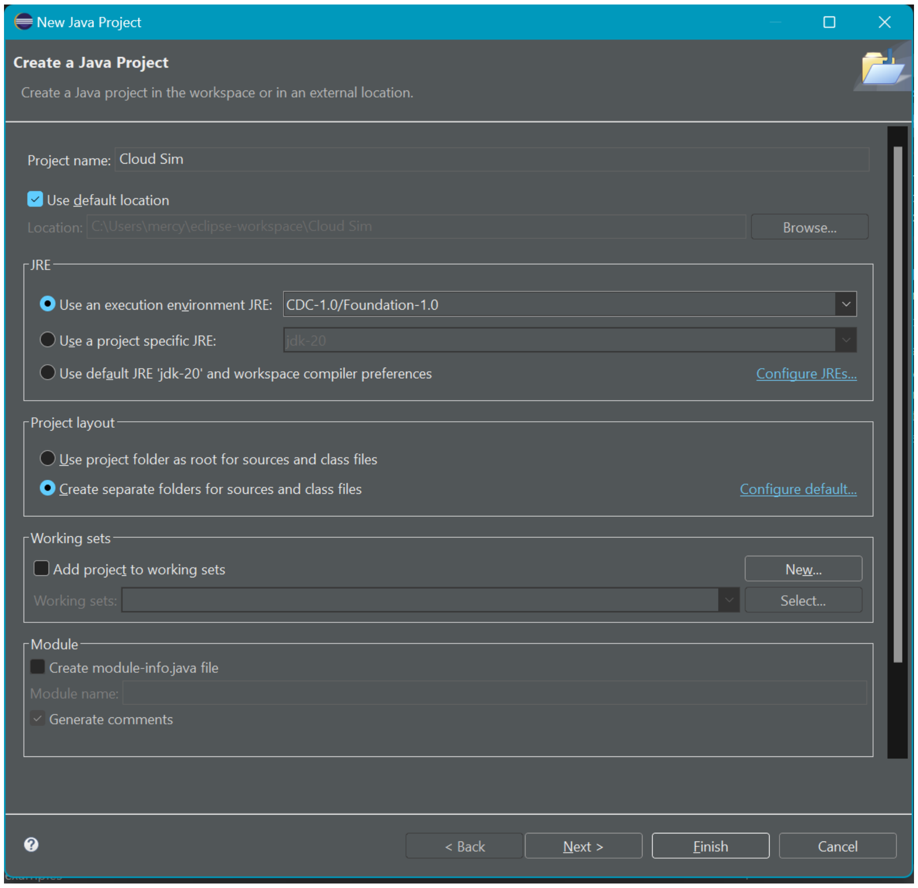
- Introduction

This cheat sheet shows you how to create the famous "Hello World" application and try it out. You will create a Java project and a Java class that will print "Hello world!" in the console when run.

If you need help at any step, click the (?) to the right. Let's get started!

Click to Restart

- Open the Java perspective
- Create a Java project
- Create your HelloWorld class
- Add a print statement
- Run your Java application


New Java Project

Create a Java Project

Create a Java project in the workspace or in an external location.

Project name: Cloud Sim

[✓] Use default location

Location: C:\Users\mercy\eclipse-workspace\Cloud Sim

JRE

[● Use an execution environment JRE: CDC-1.0/Foundation-1.0]

○ Use a project specific JRE: jdk-20

○ Use default JRE 'jdk-20' and workspace compiler preferences

Project layout

○ Use project folder as root for sources and class files

[● Create separate folders for sources and class files]

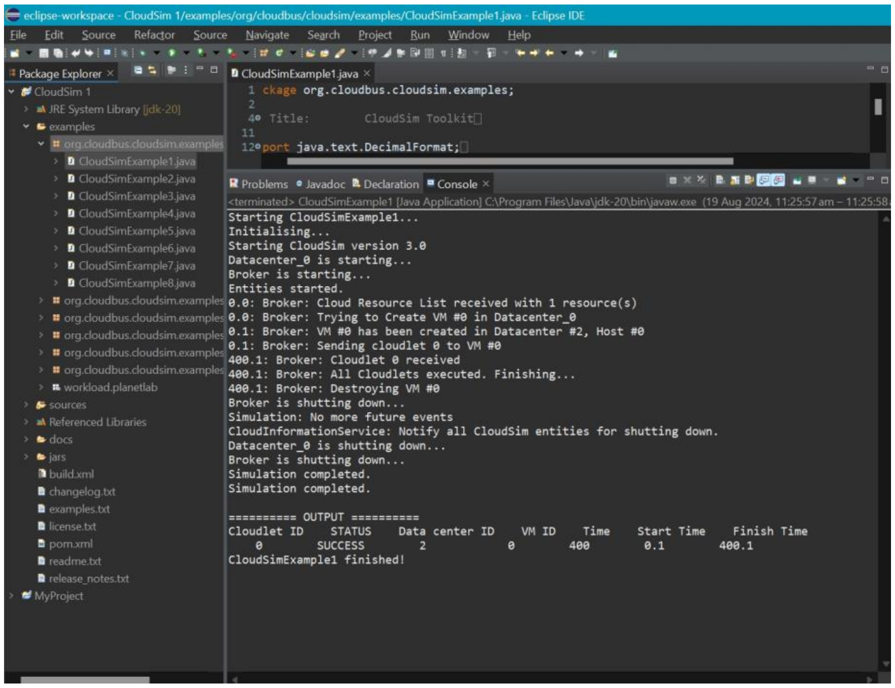
Working sets

[ ] Add project to working sets

Module

[ ] Create module-info.java file

[✓] Generate comments


eclipse-workspace - CloudSim 1/examples/org/cloudbus/cloudsim/examples/CloudSimExample1.java - Eclipse IDE

```
ckage org.cloudbus.cloudsim.examples;

 Title:        CloudSim Toolkit

port java.text.DecimalFormat;
```

```
<terminated> CloudSimExample1 [Java Application] C:\Program Files\Java\jdk-20\bin\javaw.exe (19 Aug 2024, 11:25:57 am – 11:25:58
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.