CCL Assignment No. 3

Name- Rohit Anirudha Pendse

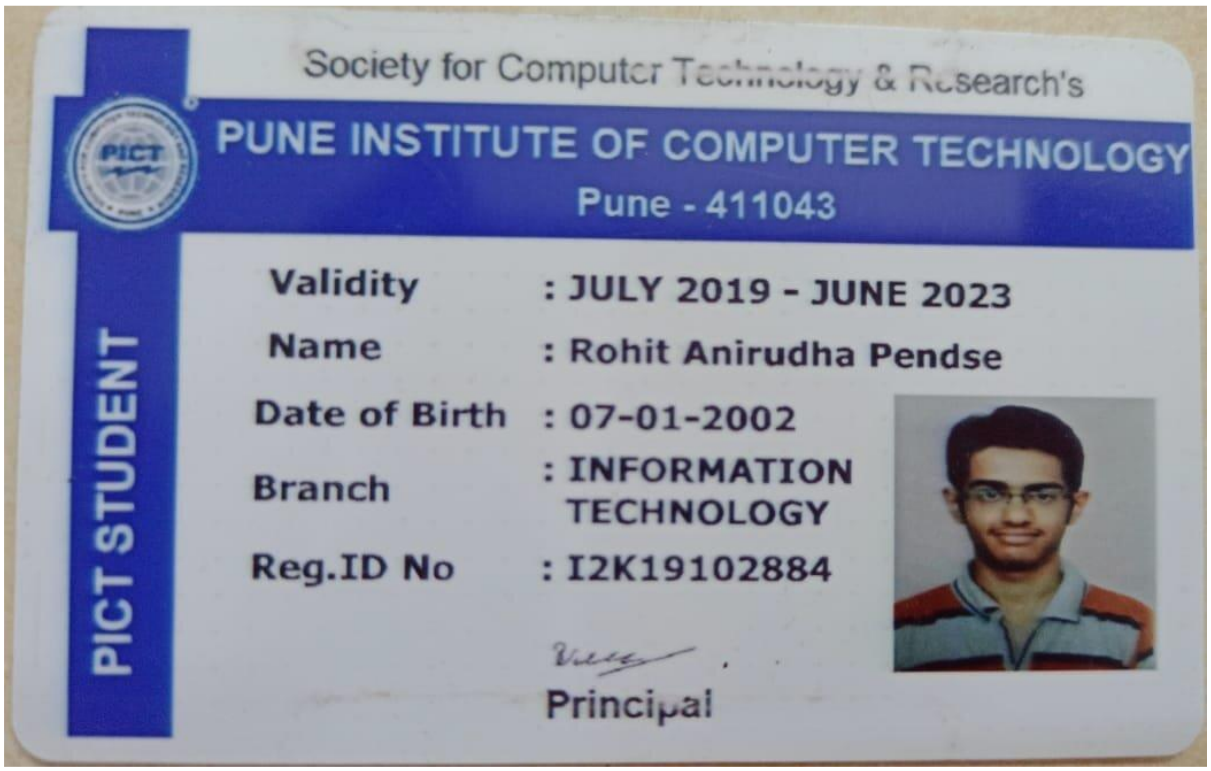
Roll No.- 33358

Batch- N11

Contact No.- 8766925932

Email Id- rapendse2002@gmail.com

ID card-



Name : Rohit Pendse Date :
Roll no: 33358
Class : TE I I
Subject : Cloud Computing Laboratory (CCCL)

Assignment - 3

Aim: Cloud Sim

Problem statement:
Simulate a cloud scenario using cloudsim and run a scheduling algorithm that is not present in cloudsim.

Theory:

Q.1. What is cloudsim?

→ - Cloudsim is an open-source framework, which is used to simulate cloud computing infrastructure and services.
- It is developed by 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 hypothesis prior to software development in order to reproduce tests and results.

Q.2. What cloudsim is used for?

→ - Cloudsim is used for modelling and simulating a cloud computing environment as a means of evaluating hypothesis prior to software development in order to reproduce tests & results.

- Cloudsim is an open-source framework of cost free. So it favours researchers working in field
- It is easy to download & setup.
- It is also moved generalized & extensible to support modelling and experimentation

Q.3. What is cloudsim & how it works?

→ Cloudsim is simulation toolkit that supports modelling & simulation of core functionality of cloud like job queue, processing of events, creation of cloud entities etc.

- How it works:
Cloudsim allows to model & simulate cloud system components, hence to support its function different sets of classes have been developed by its developers like

1. Simulating regions & data centers the class named 'Datacenter.java' is available in org.cloudbus.cloudsim package.
2. To simulate workloads for cloud the class named as "Cloudlet.java" is available in org.cloudbus.cloudsim package.
3. To simulate load balancing the policy related implementation classes named "DatacenterBroker.java" etc are available under org.cloudbus.cloudsim package.
4. Now because of different simulated hardware models are required to communicate with each other to share the cloudsim work.

Updates for cloudsim has implemented a discrete event simulation engine that keeps track of all tasks/assignments among the different simulated cloud components

Q.4. What are components of ~~of~~ cloudsim?

→ Basic components of cloudsim are:

1. Datacenter:
- Datacenter is used to model the core services as the system level of cloud infrastructure.
- It consists of a set of hosts which manage a set of virtual machines whose tasks are to handle low level processing

2. Host:
- The host is a component is used to assign processing ~~processing~~ capabilities, memory & scheduling policy to allocate different processing cores to multiple virtual machines that is in the list of VM's managed by host.

3. Virtual machines.
- This component manages allocation of different virtual machines ~~different~~ hosts, so that processing cores can be scheduled to virtual machines.

4. Datacenter broker:
- Responsibility of a broker is to mediate between user & service providers, depending on requirement of qos that user specifies

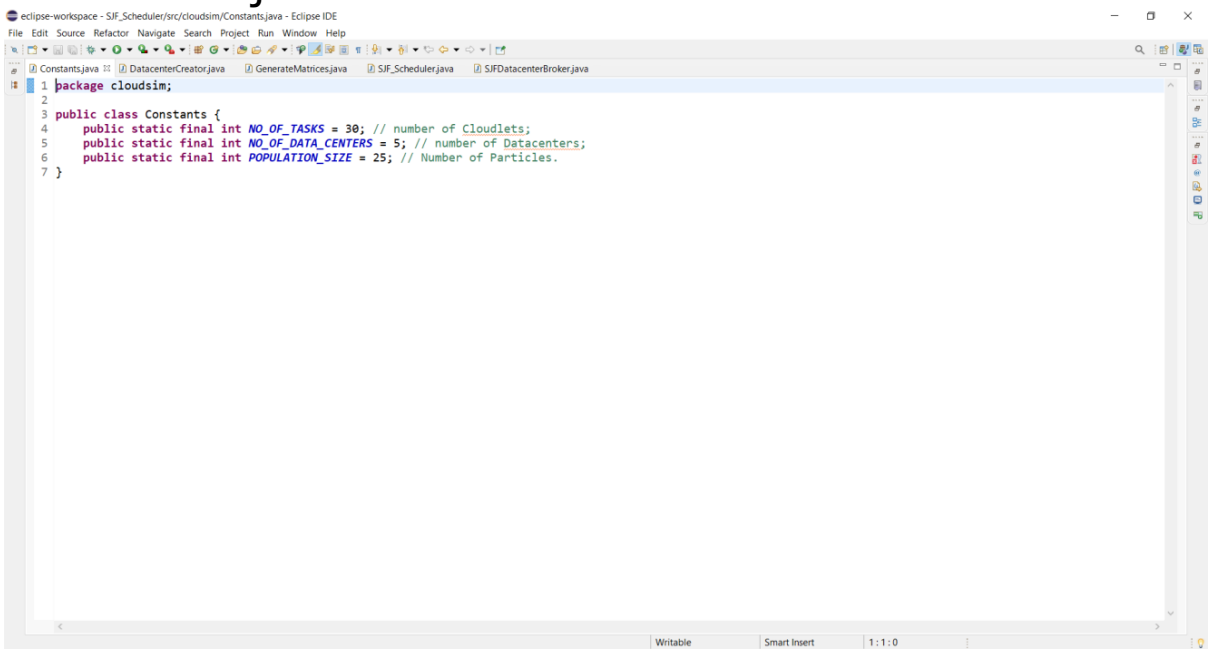
5. Cloudlet:
- This component represents application service whose complexity is modulated in cloudsim in terms of computational requirements.

Q.5. Explain scheduling algorithm used in assignment.

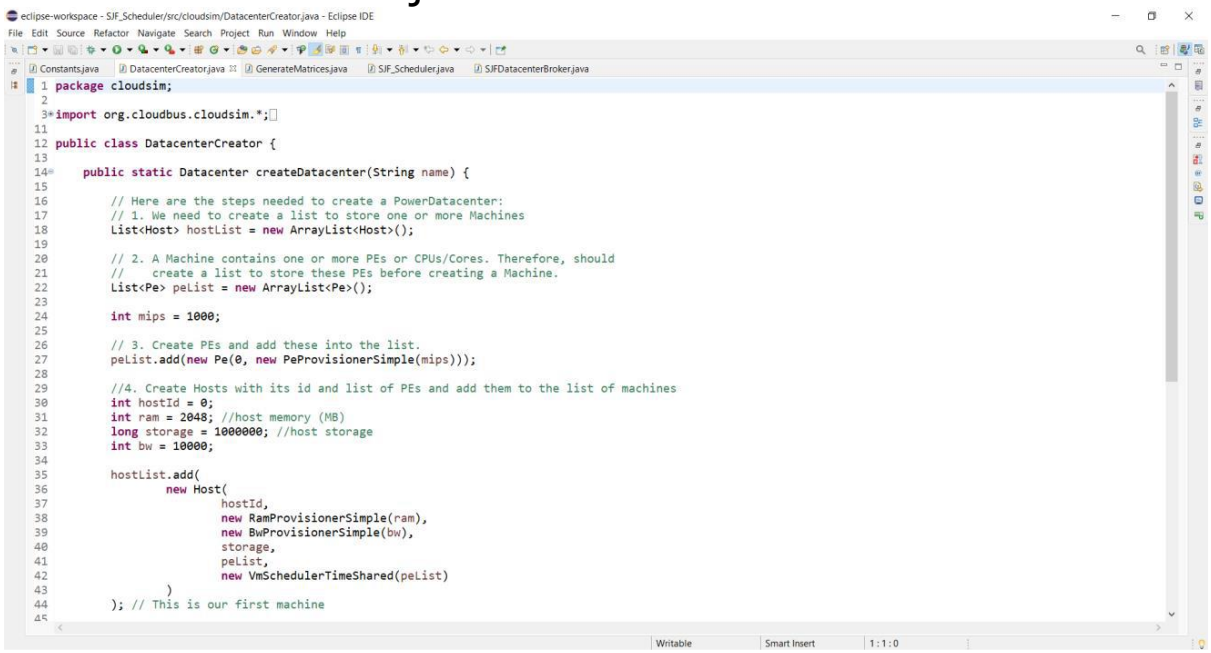
→
- SJF stands for Shortest Job first.
- SJF has advantage of having min average waiting time among all scheduling algorithms.
- It is a greedy algorithm.
- It may cause starvation if shorter processes keep coming. This can be solved using concept of aging.
- It is practically not feasible as burst time may not be known & therefore it may not sort them.
- While it is not possible to predict execution times, several methods can be used to estimate execution time for a job, such as weighted average of previous execution times.
- SJF can be used in specialized envs where accurate estimates of running time are available.

Code Screenshots:

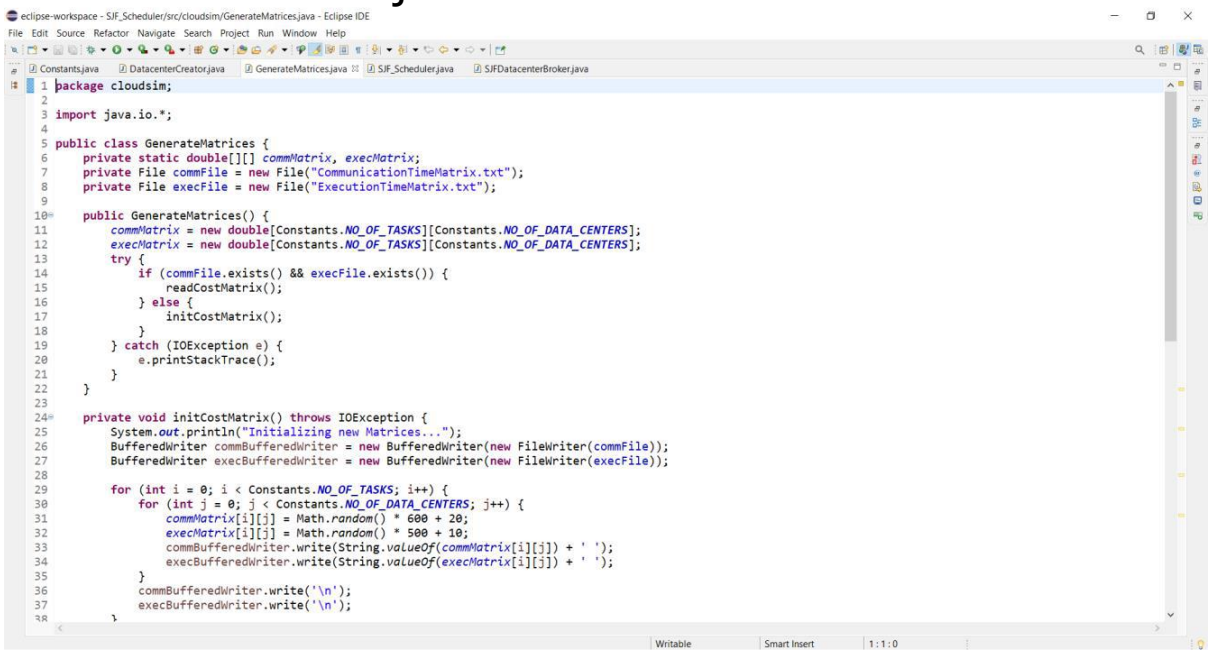
1. Constants.java



2. DatacenterCreator.java

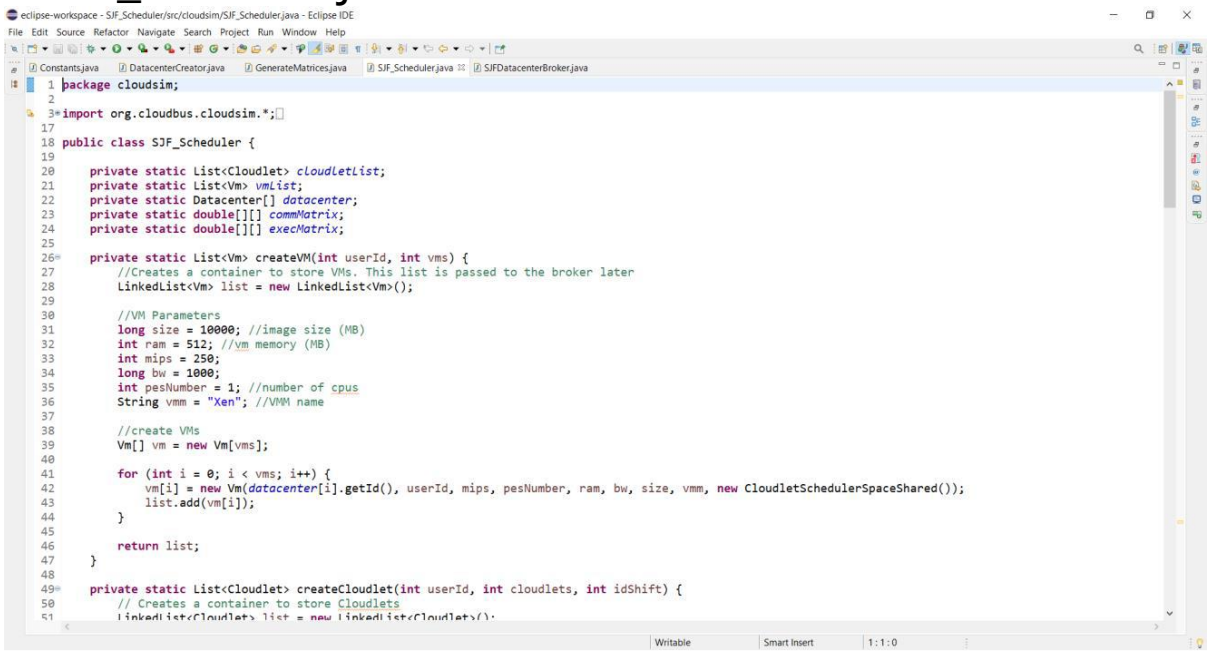


3. GenerateMatrices.java

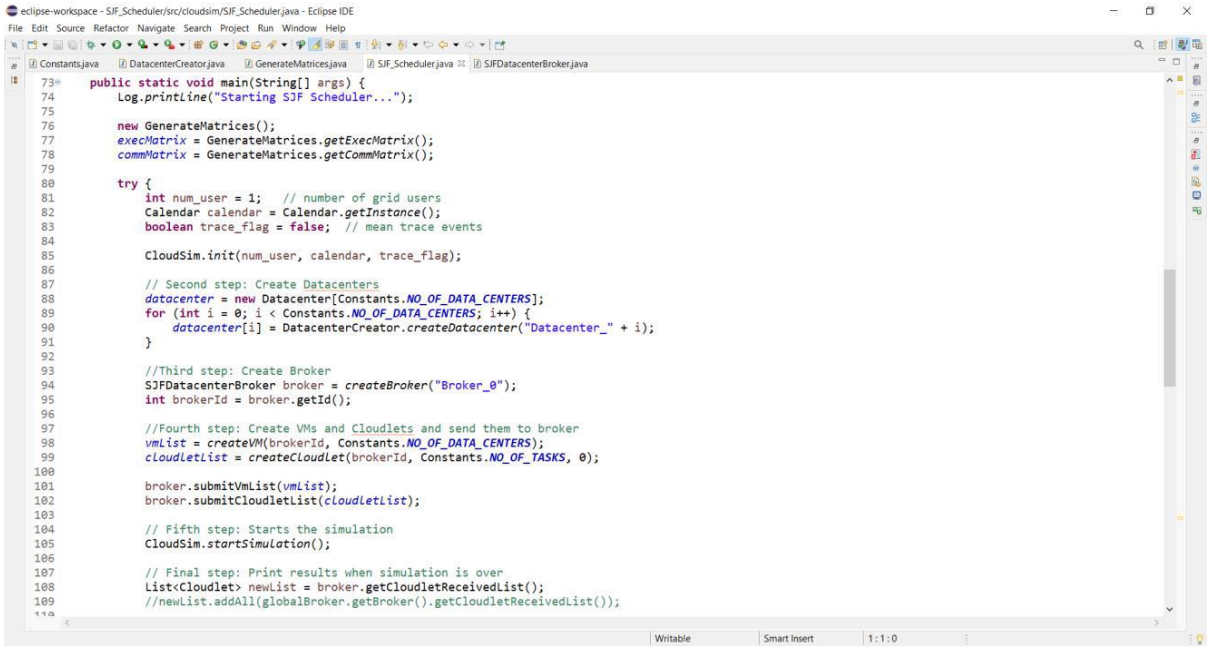



```java
package cloudsim;

import java.io.*;

public class GenerateMatrices {
    private static double[][] commMatrix, execMatrix;
    private File commFile = new File("CommunicationTimeMatrix.txt");
    private File execFile = new File("ExecutionTimeMatrix.txt");

    public GenerateMatrices() {
        commMatrix = new double[Constants.NO_OF_TASKS][Constants.NO_OF_DATA_CENTERS];
        execMatrix = new double[Constants.NO_OF_TASKS][Constants.NO_OF_DATA_CENTERS];
        try {
            if (commFile.exists() && execFile.exists()) {
                readCostMatrix();
            } else {
                initCostMatrix();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private void initCostMatrix() throws IOException {
        System.out.println("Initializing new Matrices...");
        BufferedWriter commBufferedWriter = new BufferedWriter(new FileWriter(commFile));
        BufferedWriter execBufferedWriter = new BufferedWriter(new FileWriter(execFile));

        for (int i = 0; i < Constants.NO_OF_TASKS; i++) {
            for (int j = 0; j < Constants.NO_OF_DATA_CENTERS; j++) {
                commMatrix[i][j] = Math.random() * 600 + 20;
                execMatrix[i][j] = Math.random() * 500 + 10;
                commBufferedWriter.write(String.valueOf(commMatrix[i][j]) + ' ');
                execBufferedWriter.write(String.valueOf(execMatrix[i][j]) + ' ');
            }
            commBufferedWriter.write('\n');
            execBufferedWriter.write('\n');
```

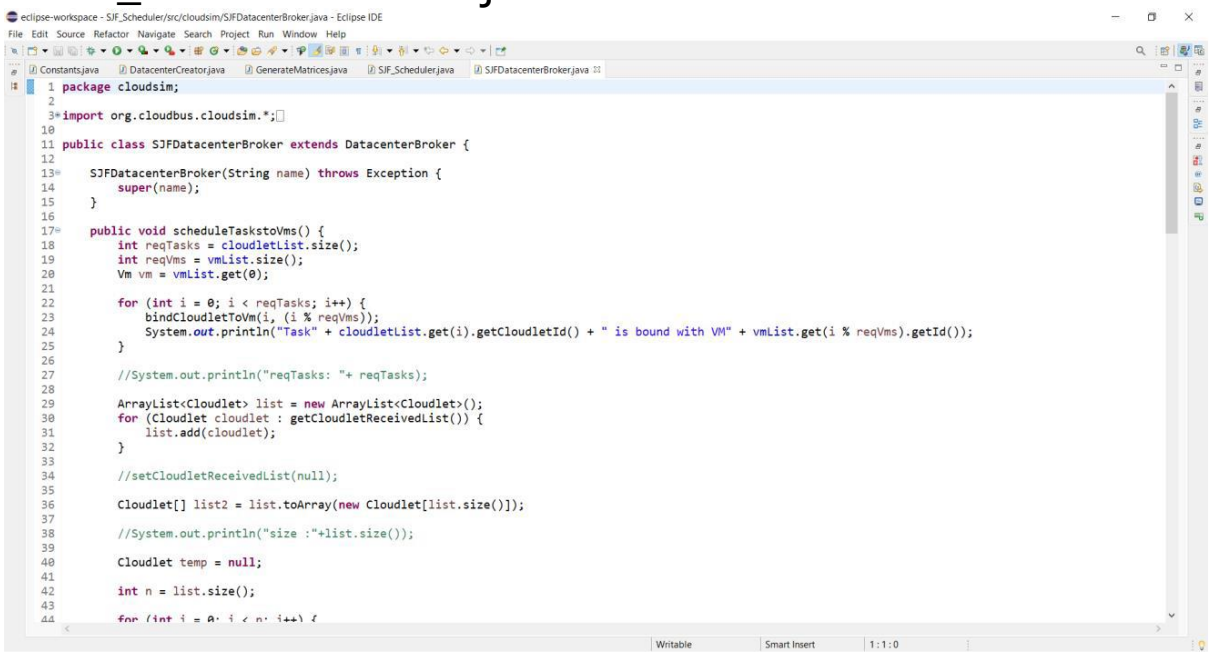
4. SJF_Scheduler.java



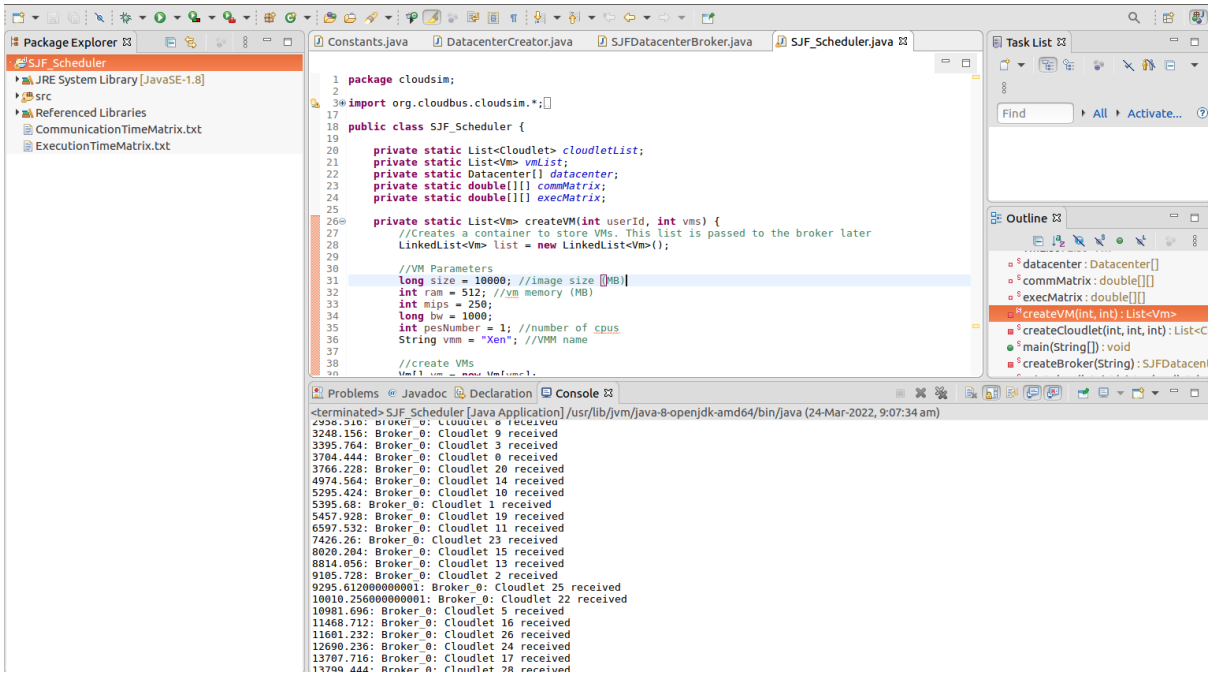

```java
package cloudsim;

import org.cloudbus.cloudsim.*;

public class SJF_Scheduler {

    private static List<Cloudlet> cloudletList;
    private static List<Vm> vmList;
    private static Datacenter[] datacenter;
    private static double[][] commMatrix;
    private static double[][] execMatrix;

    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 = 250;
        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(datacenter[i].getId(), userId, mips, pesNumber, ram, bw, size, vmm, new CloudletSchedulerSpaceShared());
            list.add(vm[i]);
        }

        return list;
    }

    private static List<Cloudlet> createCloudlet(int userId, int cloudlets, int idShift) {
        // Creates a container to store Cloudlets
        LinkedList<Cloudlet> list = new LinkedList<Cloudlet>();
```

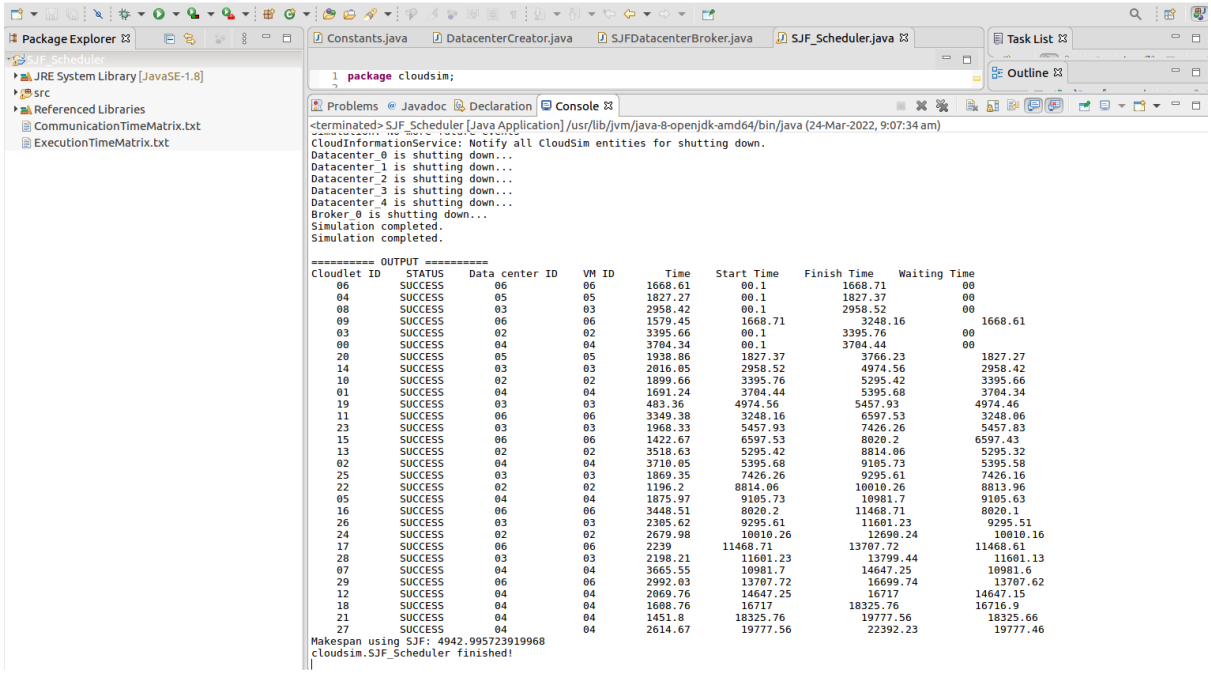


5. SJF_DatacenterBroker.java



Output Screenshots:





Conclusion:
Successfully simulated a cloudsim scenario using cloudsim & ran a scheduling algorithm of Shortest Job First (SJF) in cloudsim.

Signature
8/4