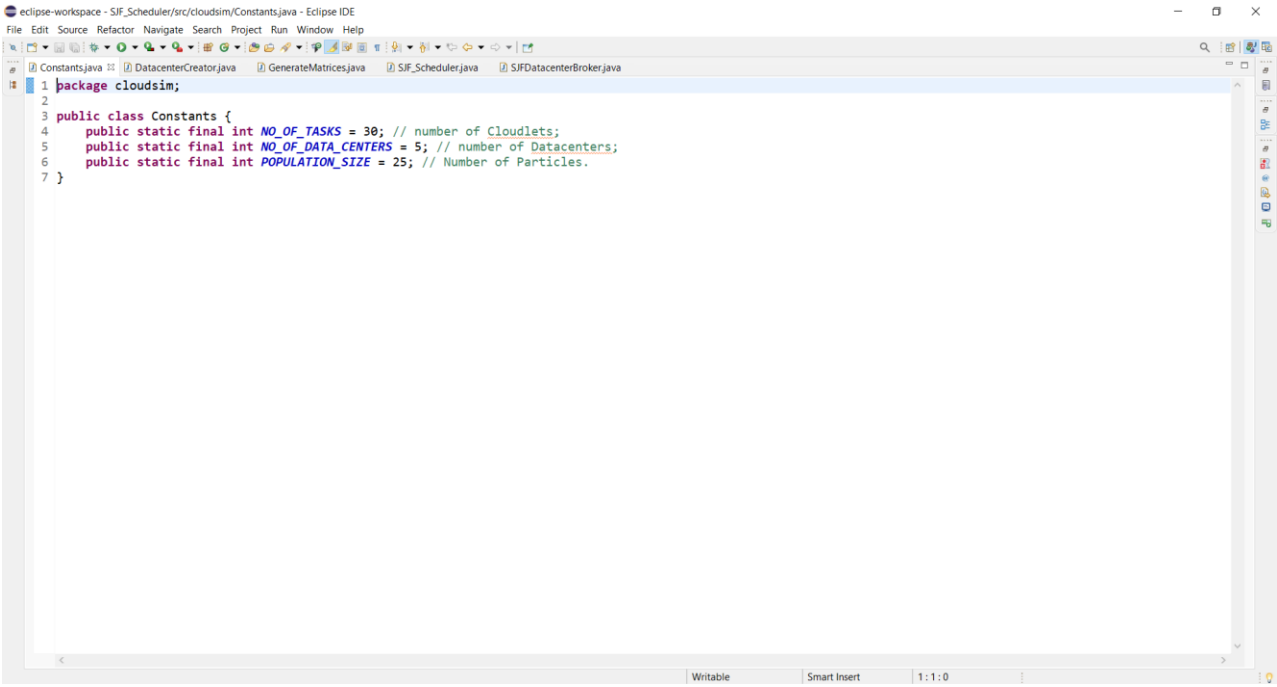
CCL Assignment: 3

Roll No.: 33373
Contact No.: 8600890813
Email ID: shlokshaha1421@gmail.com
Class: TE11
Batch: N11
Subject: Cloud Computing Lab

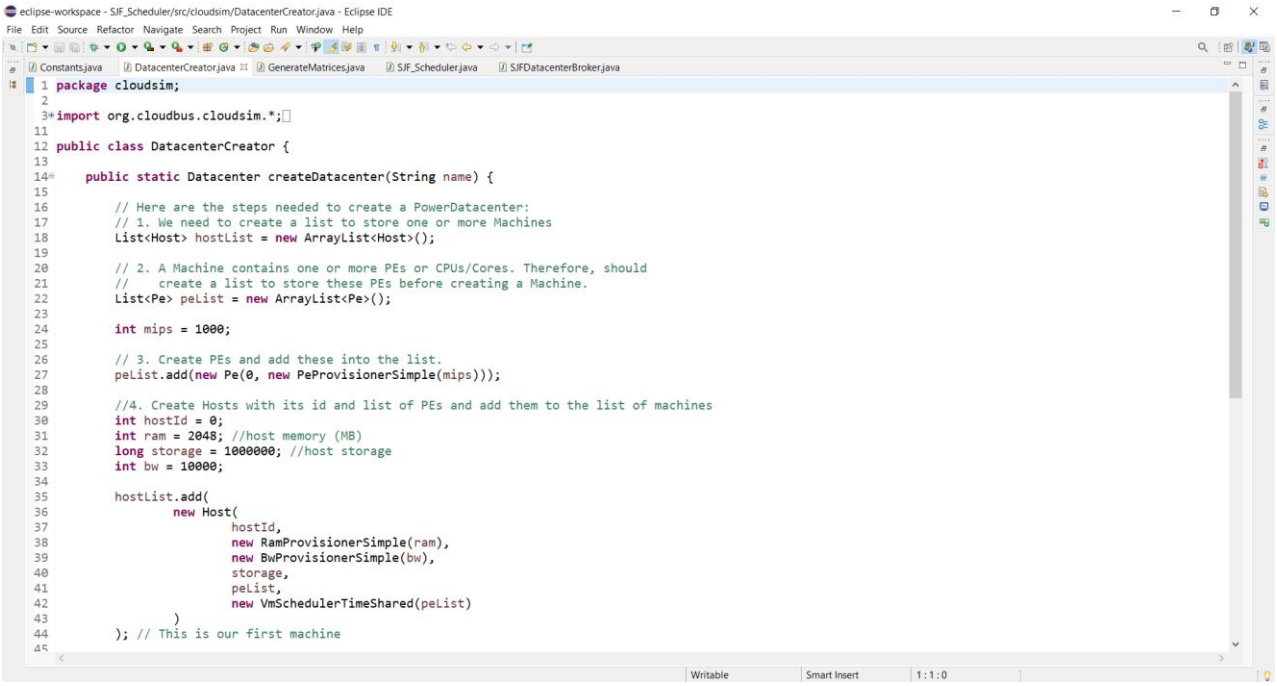
Code Screenshots:

1. Constants.java




```java
package cloudsim;

public class Constants {
    public static final int NO_OF_TASKS = 30; // number of Cloudlets;
    public static final int NO_OF_DATA_CENTERS = 5; // number of Datacenters;
    public static final int POPULATION_SIZE = 25; // Number of Particles.
}
```

2. DatacenterCreator.java




```java
package cloudsim;

import org.cloudbus.cloudsim.*;

public class DatacenterCreator {

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

        int mips = 1000;

        // 3. Create PEs and add these into the list.
        peList.add(new Pe(0, 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,
                        peList,
                        new VmSchedulerTimeShared(peList)
                )
        ); // This is our first machine
```

3. GenerateMatrices.java

eclipse-workspace - SJF_Scheduler/src/cloudsim/GenerateMatrices.java - Eclipse IDE

```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

eclipse-workspace - SJF_Scheduler/src/cloudsim/SJF_Scheduler.java - Eclipse IDE

```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>();
```

eclipse-workspace - SJF_Scheduler/src/cloudsim/SJF_Scheduler.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Constants.java | DatacenterCreator.java | GenerateMatrices.java | SJF_Scheduler.java | SJFDatacenterBroker.java

```java
    public static void main(String[] args) {
        Log.printLine("Starting SJF Scheduler...");

        new GenerateMatrices();
        execMatrix = GenerateMatrices.getExecMatrix();
        commMatrix = GenerateMatrices.getCommMatrix();

        try {
            int num_user = 1;   // number of grid users
            Calendar calendar = Calendar.getInstance();
            boolean trace_flag = false;  // mean trace events

            CloudSim.init(num_user, calendar, trace_flag);

            // Second step: Create Datacenters
            datacenter = new Datacenter[Constants.NO_OF_DATA_CENTERS];
            for (int i = 0; i < Constants.NO_OF_DATA_CENTERS; i++) {
                datacenter[i] = DatacenterCreator.createDatacenter("Datacenter_" + i);
            }

            //Third step: Create Broker
            SJFDatacenterBroker broker = createBroker("Broker_0");
            int brokerId = broker.getId();

            //Fourth step: Create VMs and Cloudlets and send them to broker
            vmList = createVM(brokerId, Constants.NO_OF_DATA_CENTERS);
            cloudletList = createCloudlet(brokerId, Constants.NO_OF_TASKS, 0);

            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();
            //newList.addAll(globalBroker.getBroker().getCloudletReceivedList());
```

Writable | Smart Insert | 1 : 1 : 0

5. SJF_DatacenterBroker.java

eclipse-workspace - SJF_Scheduler/src/cloudsim/SJFDatacenterBroker.java - Eclipse IDE

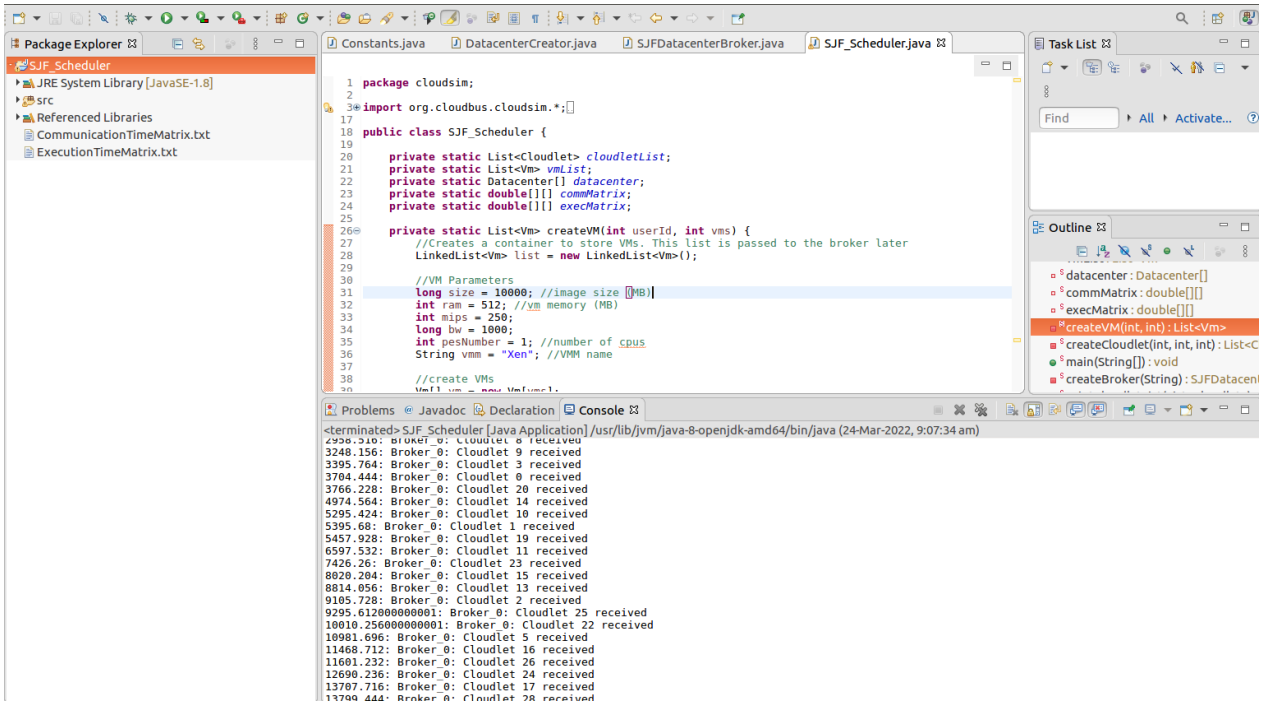
File Edit Source Refactor Navigate Search Project Run Window Help

Constants.java | DatacenterCreator.java | GenerateMatrices.java | SJF_Scheduler.java | SJFDatacenterBroker.java

```java
package cloudsim;

import org.cloudbus.cloudsim.*;

public class SJFDatacenterBroker extends DatacenterBroker {

    SJFDatacenterBroker(String name) throws Exception {
        super(name);
    }

    public void scheduleTaskstoVms() {
        int reqTasks = cloudletList.size();
        int reqVms = vmList.size();
        Vm vm = vmList.get(0);

        for (int i = 0; i < reqTasks; i++) {
            bindCloudletToVm(i, (i % reqVms));
            System.out.println("Task" + cloudletList.get(i).getCloudletId() + " is bound with VM" + vmList.get(i % reqVms).getId());
        }

        //System.out.println("reqTasks: "+ reqTasks);

        ArrayList<Cloudlet> list = new ArrayList<Cloudlet>();
        for (Cloudlet cloudlet : getCloudletReceivedList()) {
            list.add(cloudlet);
        }

        //setCloudletReceivedList(null);

        Cloudlet[] list2 = list.toArray(new Cloudlet[list.size()]);

        //System.out.println("size :"+list.size());

        Cloudlet temp = null;

        int n = list.size();

        for (int i = 0; i < n; i++) {
```

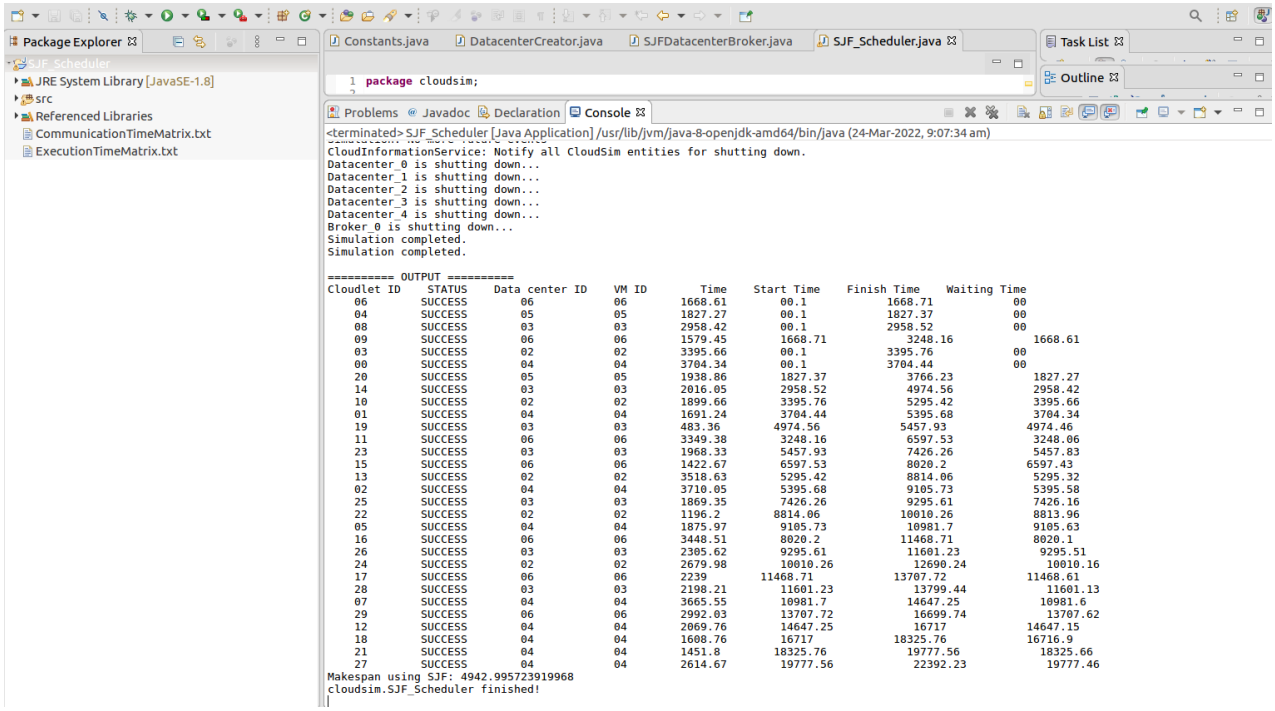
Writable | Smart Insert | 1 : 1 : 0

Output Screenshots:




```
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
```

```
3248.156: Broker_0: Cloudlet 9 received
3395.764: Broker_0: Cloudlet 3 received
3704.444: Broker_0: Cloudlet 0 received
3766.228: Broker_0: Cloudlet 20 received
4974.564: Broker_0: Cloudlet 14 received
5295.424: Broker_0: Cloudlet 10 received
5395.68: Broker_0: Cloudlet 1 received
5457.928: Broker_0: Cloudlet 19 received
6597.532: Broker_0: Cloudlet 11 received
7426.26: Broker_0: Cloudlet 23 received
8020.204: Broker_0: Cloudlet 15 received
8814.056: Broker_0: Cloudlet 13 received
9105.728: Broker_0: Cloudlet 2 received
9295.612000000001: Broker_0: Cloudlet 25 received
10010.256000000001: Broker_0: Cloudlet 22 received
10981.696: Broker_0: Cloudlet 5 received
11468.712: Broker_0: Cloudlet 16 received
11601.232: Broker_0: Cloudlet 26 received
12690.236: Broker_0: Cloudlet 24 received
13707.716: Broker_0: Cloudlet 17 received
```




```
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Datacenter_1 is shutting down...
Datacenter_2 is shutting down...
Datacenter_3 is shutting down...
Datacenter_4 is shutting down...
Broker_0 is shutting down...
Simulation completed.
Simulation completed.

========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID        Time    Start Time    Finish Time    Waiting Time
    06         SUCCESS        06            06        1668.61       00.1          1668.71          00
    04         SUCCESS        05            05        1827.27       00.1          1827.37          00
    08         SUCCESS        03            03        2958.42       00.1          2958.52          00
    09         SUCCESS        06            06        1579.45       1668.71          3248.16          1668.61
    03         SUCCESS        02            02        3395.66       00.1          3395.76          00
    00         SUCCESS        04            04        3704.34       00.1          3704.44          00
    20         SUCCESS        05            05        1938.86       1827.37          3766.23          1827.27
    14         SUCCESS        03            03        2016.05       2958.52          4974.56          2958.42
    10         SUCCESS        02            02        1899.66       3395.76          5295.42          3395.66
    01         SUCCESS        04            04        1691.24       3704.44          5395.68          3704.34
    19         SUCCESS        03            03        483.36       4974.56          5457.93          4974.46
    11         SUCCESS        06            06        3349.38       3248.16          6597.53          3248.06
    23         SUCCESS        03            03        1968.33       5457.93          7426.26          5457.83
    15         SUCCESS        06            06        1422.67       6597.53          8020.2          6597.43
    13         SUCCESS        02            02        3518.63       5295.42          8814.06          5295.32
    02         SUCCESS        04            04        3710.05       5395.68          9105.73          5395.58
    25         SUCCESS        03            03        1869.35       7426.26          9295.61          7426.16
    22         SUCCESS        02            02        1196.2       8814.06          10010.26          8813.96
    05         SUCCESS        04            04        1875.97       9105.73          10981.7          9105.63
    16         SUCCESS        06            06        3448.51       8020.2          11468.71          8020.1
    26         SUCCESS        03            03        2305.62       9295.61          11601.23          9295.51
    24         SUCCESS        02            02        2679.98       10010.26          12690.24          10010.16
    17         SUCCESS        06            06        2239       11468.71          13707.72          11468.61
    28         SUCCESS        03            03        2198.21       11601.23          13799.44          11601.13
    07         SUCCESS        04            04        3665.55       10981.7          14647.25          10981.6
    29         SUCCESS        06            06        2992.03       13707.72          16699.74          13707.62
    12         SUCCESS        04            04        2069.76       14647.25          16717          14647.15
    18         SUCCESS        04            04        1608.76       16717          18325.76          16716.9
    21         SUCCESS        04            04        1451.8       18325.76          19777.56          18325.66
    27         SUCCESS        04            04        2614.67       19777.56          22392.23          19777.46
Makespan using SJF: 4942.995723919968
cloudsim.SJF_Scheduler finished!
```