كلية العلوم و التقنيات بطنجة
Faculté des Sciences et Techniques de Tanger

IASD
INTELLIGENCE ARTIFICIELLE
ET SCIENCES DE DONNEES

# Cloud Computing

## Activities-3-Report

Supervised by Pr. Chakir El Amrani

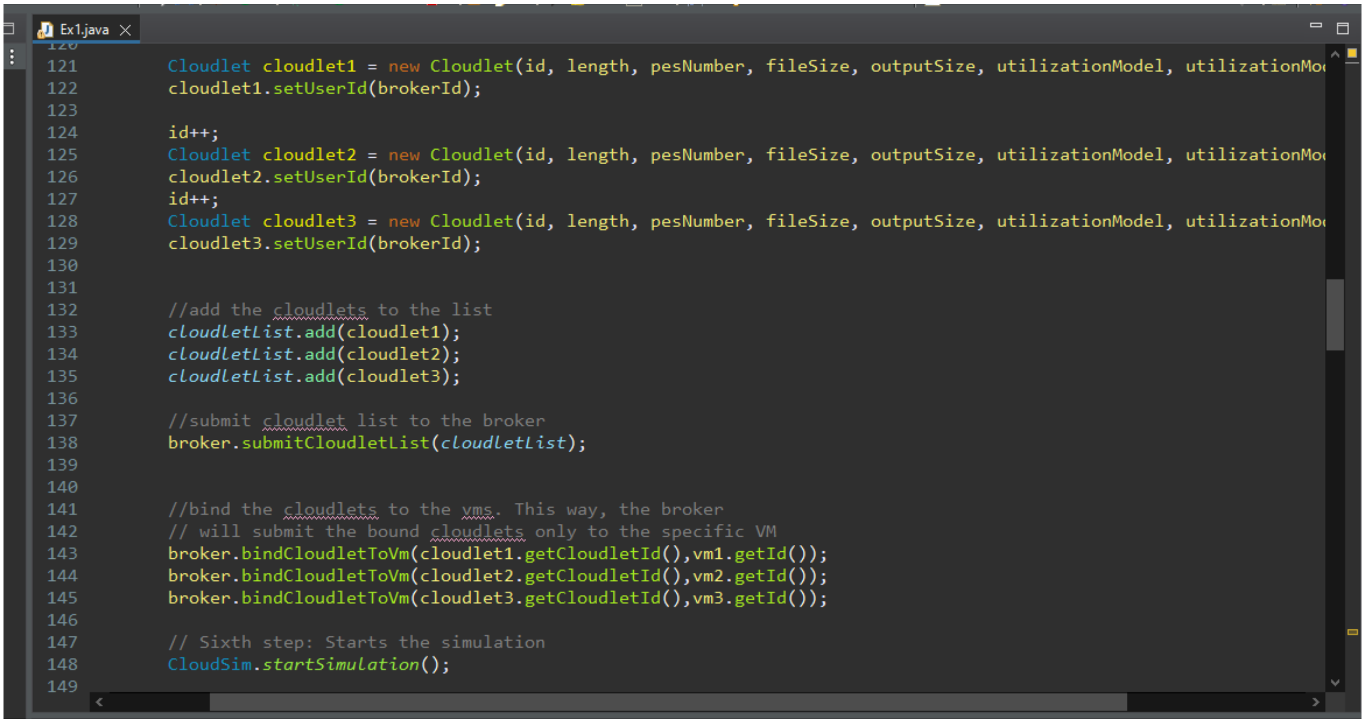
Realized by EL GHAZI Loubna

# Contents

## 1-Exercice 1

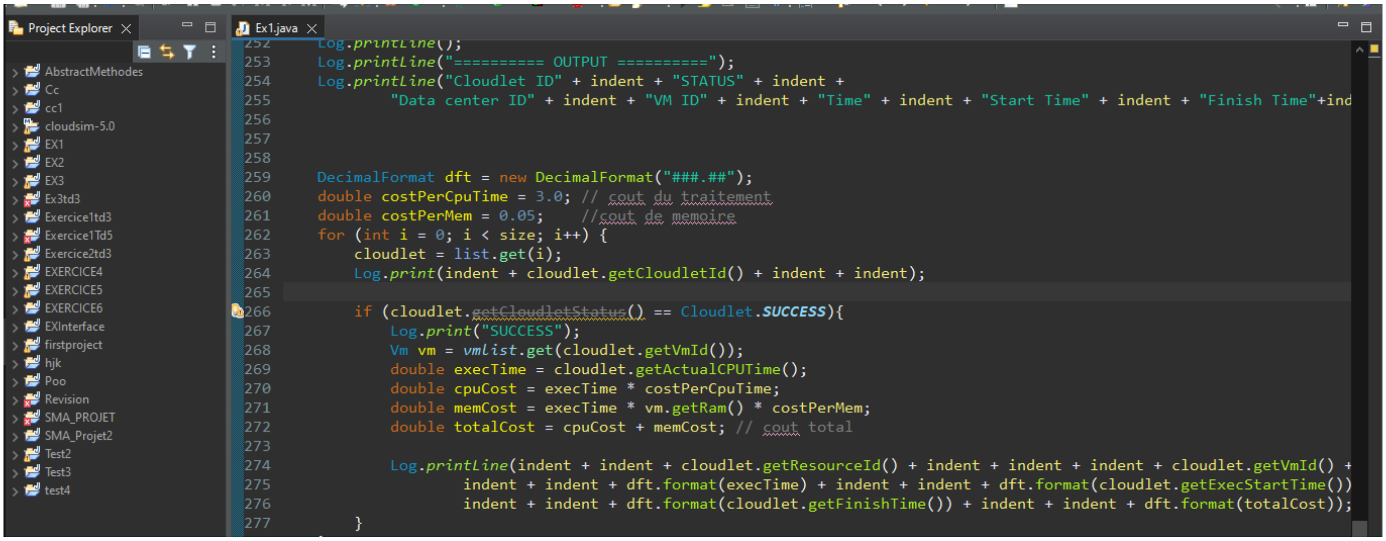
On the basis of “example2”,  we simulated 3 cloudlets and calculated  the cost as shown below :

**1-The creation of VMs and Cloudlets : The three VMs and three cloudlets are created and correctly added to their respective lists. Each cloudlet is properly assigned to a unique VM via broker.bindCloudletToVm.**



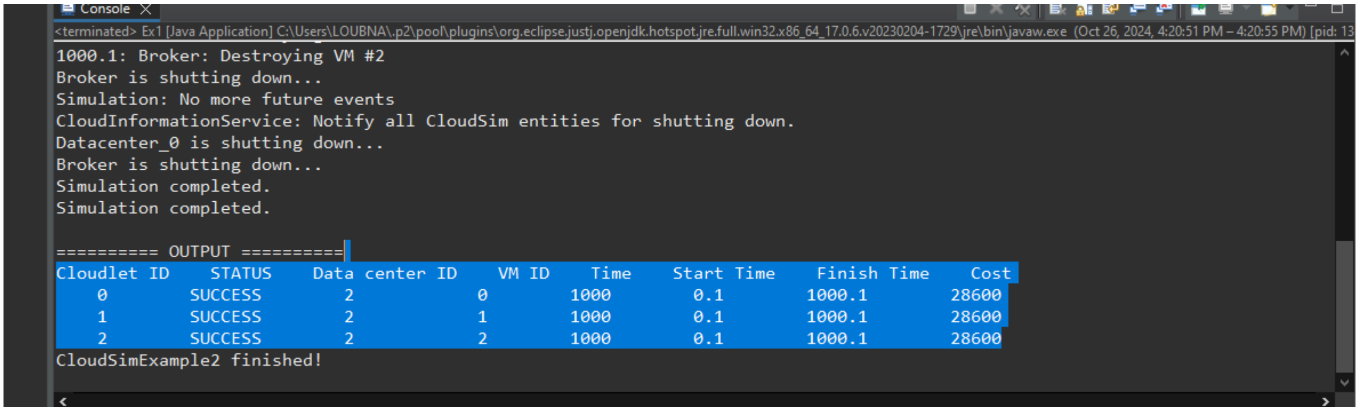
```java
        Cloudlet cloudlet1 = new Cloudlet(id, length, pesNumber, fileSize, outputSize, utilizationModel, utilizationMo
        cloudlet1.setUserId(brokerId);

        id++;
        Cloudlet cloudlet2 = new Cloudlet(id, length, pesNumber, fileSize, outputSize, utilizationModel, utilizationMo
        cloudlet2.setUserId(brokerId);
        id++;
        Cloudlet cloudlet3 = new Cloudlet(id, length, pesNumber, fileSize, outputSize, utilizationModel, utilizationMo
        cloudlet3.setUserId(brokerId);


        //add the cloudlets to the list
        cloudletList.add(cloudlet1);
        cloudletList.add(cloudlet2);
        cloudletList.add(cloudlet3);

        //submit cloudlet list to the broker
        broker.submitCloudletList(cloudletList);


        //bind the cloudlets to the vms. This way, the broker
        // will submit the bound cloudlets only to the specific VM
        broker.bindCloudletToVm(cloudlet1.getCloudletId(),vm1.getId());
        broker.bindCloudletToVm(cloudlet2.getCloudletId(),vm2.getId());
        broker.bindCloudletToVm(cloudlet3.getCloudletId(),vm3.getId());

        // Sixth step: Starts the simulation
        CloudSim.startSimulation();

```

**2-Cost Calculation: the total cost for each cloudlet is accurately calculated by combining the CPU and memory costs.**



```java
    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"+ind



    DecimalFormat dft = new DecimalFormat("###.##");
    double costPerCpuTime = 3.0; // cout du traitement
    double costPerMem = 0.05;    //cout de memoire
    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");
            Vm vm = vmlist.get(cloudlet.getVmId());
            double execTime = cloudlet.getActualCPUTime();
            double cpuCost = execTime * costPerCpuTime;
            double memCost = execTime * vm.getRam() * costPerMem;
            double totalCost = cpuCost + memCost; // cout total

            Log.printLine(indent + indent + cloudlet.getResourceId() + indent + indent + indent + cloudlet.getVmId() +
                    indent + indent + dft.format(execTime) + indent + indent + dft.format(cloudlet.getExecStartTime())
                    indent + indent + dft.format(cloudlet.getFinishTime()) + indent + indent + dft.format(totalCost));
        }
```

**3-Execution :the final output table includes a column for cost, meeting the requirements.**



```
<terminated> Ex1 [Java Application] C:\Users\LOUBNA\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.6.v20230204-1729\jre\bin\javaw.exe (Oct 26, 2024, 4:20:51 PM – 4:20:55 PM) [pid: 13
1000.1: Broker: Destroying VM #2
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     Cost
    0          SUCCESS          2               0       1000        0.1          1000.1         28600
    1          SUCCESS          2               1       1000        0.1          1000.1         28600
    2          SUCCESS          2               2       1000        0.1          1000.1         28600
CloudSimExample2 finished!
```

**Each VM is hosted independentl so the execution characteristics for each cloudlet will be nearly identical, resulting in the same total cost.**

## 2-Exercice 2

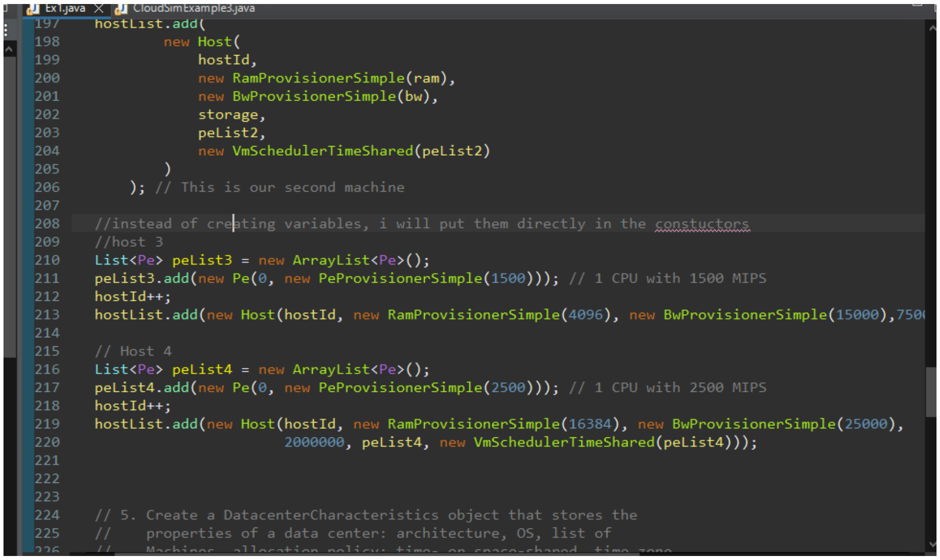
On the basis of “example3”:  we simulated 4 hosts having each different characteristics, and ran 5 various cloudlets:

The cloudlets  take different time to complete the execution depending on the requested VM performance :

four Hosts: Each with different RAM, storage, bandwidth, and MIPS.

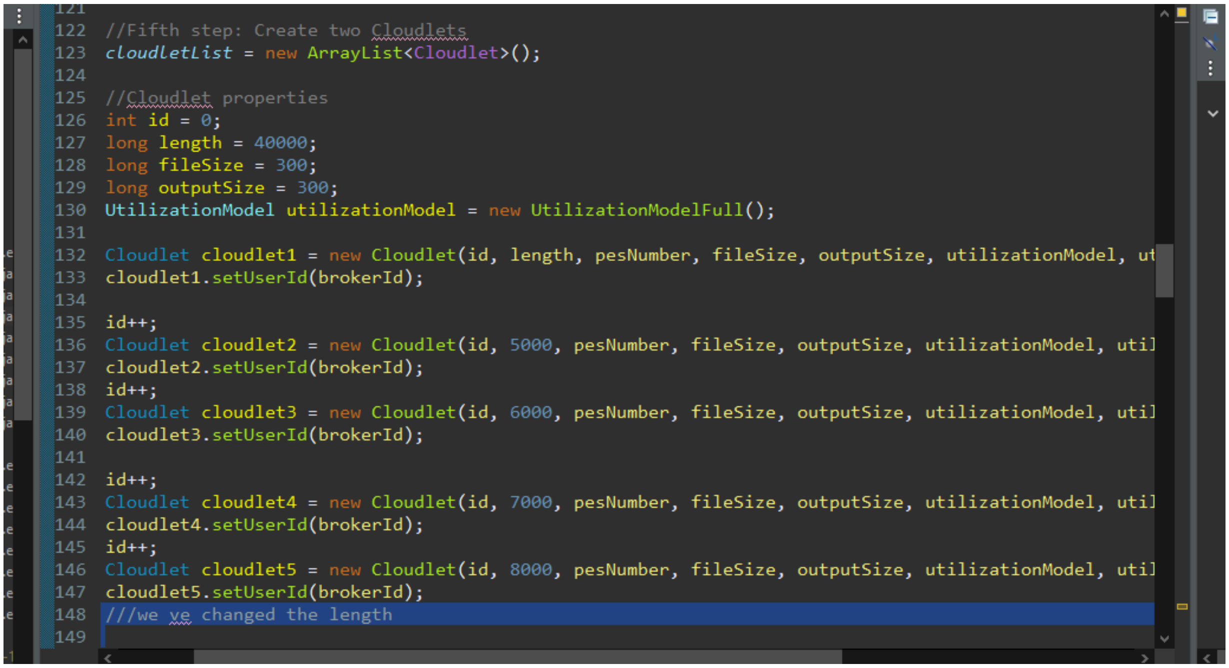
Five Cloudlets: Different length attributes simulate varying workloads.

**1-Adding the four hosts :**



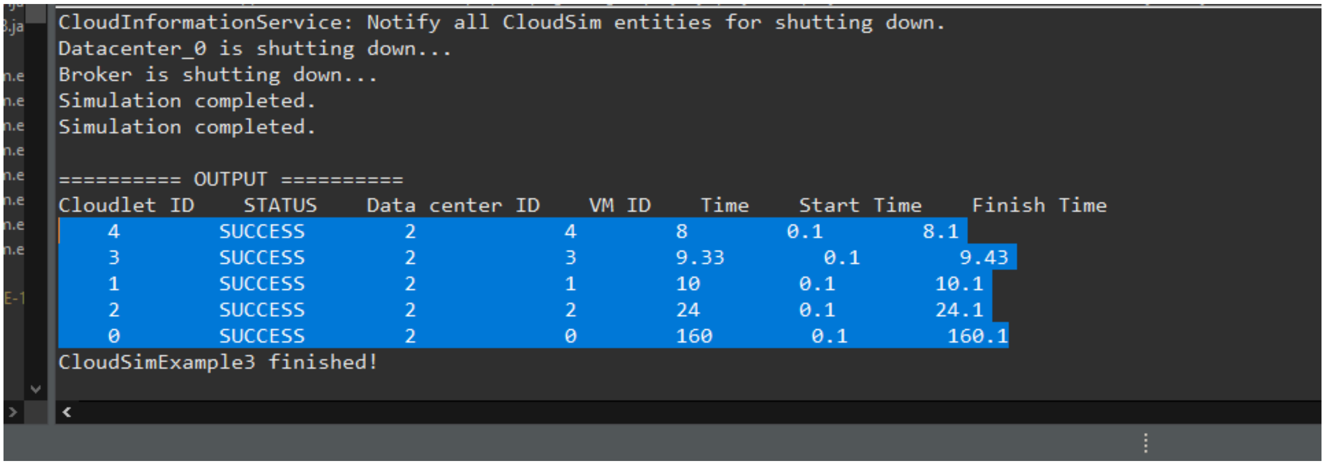
```java
    hostList.add(
            new Host(
                hostId,
                new RamProvisionerSimple(ram),
                new BwProvisionerSimple(bw),
                storage,
                peList2,
                new VmSchedulerTimeShared(peList2)
            )
        ); // This is our second machine

    //instead of creating variables, i will put them directly in the constuctors
    //host 3
    List<Pe> peList3 = new ArrayList<Pe>();
    peList3.add(new Pe(0, new PeProvisionerSimple(1500))); // 1 CPU with 1500 MIPS
    hostId++;
    hostList.add(new Host(hostId, new RamProvisionerSimple(4096), new BwProvisionerSimple(15000),750

    // Host 4
    List<Pe> peList4 = new ArrayList<Pe>();
    peList4.add(new Pe(0, new PeProvisionerSimple(2500))); // 1 CPU with 2500 MIPS
    hostId++;
    hostList.add(new Host(hostId, new RamProvisionerSimple(16384), new BwProvisionerSimple(25000),
                        2000000, peList4, new VmSchedulerTimeShared(peList4)));



    // 5. Create a DatacenterCharacteristics object that stores the
    //    properties of a data center: architecture, OS, list of
    //    Machines, allocation policy: time- or space-shared, time zone
```

## 2-Adding the cloudlets : The cloudlets were assigned to different VMs (5)



```
//Fifth step: Create two Cloudlets
cloudletList = new ArrayList<Cloudlet>();

//Cloudlet properties
int id = 0;
long length = 40000;
long fileSize = 300;
long outputSize = 300;
UtilizationModel utilizationModel = new UtilizationModelFull();

Cloudlet cloudlet1 = new Cloudlet(id, length, pesNumber, fileSize, outputSize, utilizationModel, ut
cloudlet1.setUserId(brokerId);

id++;
Cloudlet cloudlet2 = new Cloudlet(id, 5000, pesNumber, fileSize, outputSize, utilizationModel, util
cloudlet2.setUserId(brokerId);
id++;
Cloudlet cloudlet3 = new Cloudlet(id, 6000, pesNumber, fileSize, outputSize, utilizationModel, util
cloudlet3.setUserId(brokerId);

id++;
Cloudlet cloudlet4 = new Cloudlet(id, 7000, pesNumber, fileSize, outputSize, utilizationModel, util
cloudlet4.setUserId(brokerId);
id++;
Cloudlet cloudlet5 = new Cloudlet(id, 8000, pesNumber, fileSize, outputSize, utilizationModel, util
cloudlet5.setUserId(brokerId);
///we ve changed the length
```

## 3-Execution :



```
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
    4        SUCCESS        2              4        8          0.1          8.1
    3        SUCCESS        2              3        9.33          0.1          9.43
    1        SUCCESS        2              1        10          0.1          10.1
    2        SUCCESS        2              2        24          0.1          24.1
    0        SUCCESS        2              0        160          0.1          160.1
CloudSimExample3 finished!
```

## 4-Observation :

Execution Times: Cloudlet 0 has a significantly longer execution time (160 units) compared to others, which suggests it had the highest computational workload or was assigned fewer resources.

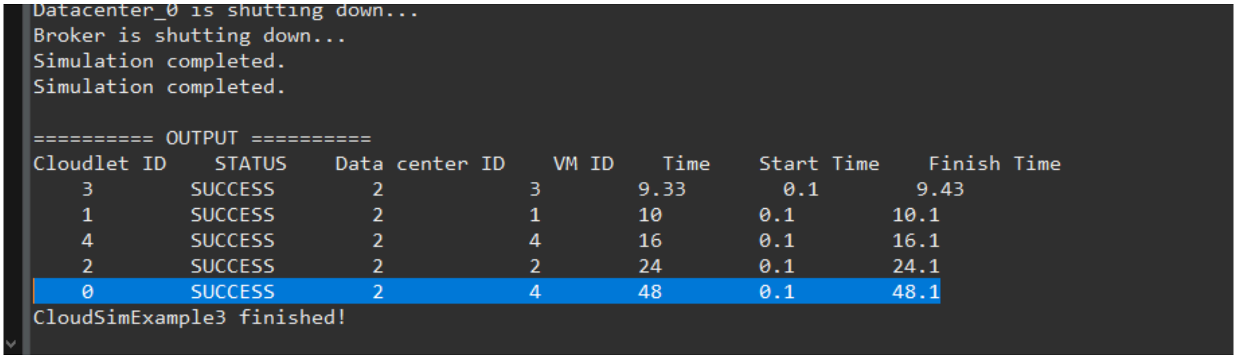
## 5-Optimizing suggestions:

**Cloudlet Length Adjustment:** We can adjust the workload for each cloudlet based on VM capabilities to reduce execution time variability

**Resource Allocation:** We can consider allocating cloudlets with higher workloads (like Cloudlet 0) to VMs with more processing power or .

**6-Test :**

**Let's allocate the cloudlet 0 to Vm 5 (Mips =250*4)**



```
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
    3         SUCCESS         2              3       9.33      0.1          9.43
    1         SUCCESS         2              1       10       0.1         10.1
    4         SUCCESS         2              4       16       0.1         16.1
    2         SUCCESS         2              2       24       0.1         24.1
    0         SUCCESS         2              4       48       0.1         48.1
CloudSimExample3 finished!
```

**Results :**

Assigning Cloudlet 0 to VM 5 significantly improved overall efficiency for the remaining cloudlets (1 to 4), as they were completed in a shorter time compared to Cloudlet 0's high execution time.

This approach enhances the overall performance and balances the execution load, especially in simulations with varying cloudlet demands.

## 3-Exercice 3

We created 3 datacenters with 2 hosts in each datacenter, and execute cloudlets for 3 users:

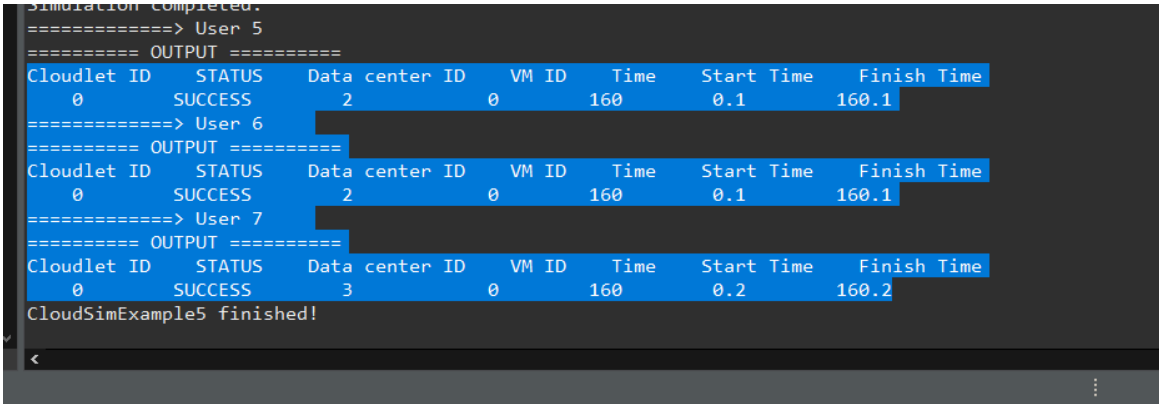
**Steps : (Based on Exercise 5)**

**1-Creating an additional datacenter (for a total of three).**

**2-Adjusting the createDatacenter method to add two hosts per datacenter.**

**3-Defining a new broker for the third user and create additional VMs and cloudlets for this user.**

**Results :**



```
Simulation completed.
=============> User 5
========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time
    0         SUCCESS         2              0       160      0.1         160.1
=============> User 6
========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time
    0         SUCCESS         2              0       160      0.1         160.1
=============> User 7
========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time
    0         SUCCESS         3              0       160      0.2         160.2
CloudSimExample5 finished!
```

**Observations :**

Each user submitted one cloudlet where:

Execution Consistency: All three cloudlets have the same execution time, which indicates identical cloudlet configurations and VM processing capabilities across data centers.

Resource Allocation: The cloudlets for Users 5 and 6 were processed in data center 2, while User 7’s cloudlet was handled in data center 3. This shows that the brokers successfully assigned cloudlets to available data centers based on load balancing or resource policies.

Slight Delay: Each cloudlet started execution at a slightly different time (0.1, 0.1, and 0.2 seconds). This minimal delay could be due to scheduling overhead or the broker's decision-making process

this resultsS suggests that the simulation successfully handled multi-user and multi-datacenter setups while managing VM resource allocation for optimal cloudlet execution.

**We can enhance the results by :**

**Dynamic VM Allocation:** Match cloudlets to VMs dynamically based on demand.

**Efficient Data Center Selection:** We can choose low-latency, nearby data centers.

**Parallel Execution**: to split large cloudlets and execute in parallel.