ASSIGNMENT 3

Objective

Simulate cloud scenarios and implement a custom scheduling algorithm in CloudSim.

Tools Required

1. **Eclipse IDE**
2. **CloudSim 3.0.3**
3. **Java Development Kit (JDK)** – Version 8 or later

Procedure & Implementation

Step 1: Setup CloudSim

1. Install **JDK** and configure JAVA_HOME.
2. Install **Eclipse IDE**.
3. Download and extract **CloudSim-3.0.3**.
4. Open Eclipse → Create a new **Java Project** → Add CloudSim `.jar` files from `cloudsim-3.0.3/jars/` into **Build Path**.

Step 2: Simulation Code

```
import org.cloudbus.cloudsim.*;
import org.cloudbus.cloudsim.core.CloudSim;

import java.util.*;

public class BasicExample {
    public static void main(String[] args) {
        int numUsers = 1;   // number of cloud users
        Calendar calendar = Calendar.getInstance();
        boolean traceFlag = false;
```

```
        CloudSim.init(numUsers, calendar, traceFlag);

        Datacenter datacenter0 = createDatacenter("Datacenter_0");

        DatacenterBroker broker = createBroker();
        int brokerId = broker.getId();

        Cloudlet cloudlet = createCloudlet(brokerId);
        Vm vm = createVM(brokerId);

        broker.submitVmList(List.of(vm));
        broker.submitCloudletList(List.of(cloudlet));

        CloudSim.startSimulation();
        CloudSim.stopSimulation();

        List<Cloudlet> results = broker.getCloudletReceivedList();
        for (Cloudlet cl : results) {
            System.out.println("Cloudlet " + cl.getCloudletId() + " finished with status " +
cl.getStatus());
        }
    }

    // Helper methods for Datacenter, Broker, VM, and Cloudlet creation
    private static Datacenter createDatacenter(String name) {
        List<Host> hostList = new ArrayList<>();
        List<Pe> peList = new ArrayList<>();
        peList.add(new Pe(0, new PeProvisionerSimple(1000))); // one CPU

        hostList.add(new Host(0, new RamProvisionerSimple(2048),
                new BwProvisionerSimple(10000), 1000000, peList,
                new VmSchedulerTimeShared(peList)));

        DatacenterCharacteristics characteristics = new DatacenterCharacteristics(
                "x86", "Linux", "Xen", hostList, 10.0, 3.0,
                0.05, 0.001, 0.0);

        try {
            return new Datacenter(name, characteristics, new
VmAllocationPolicySimple(hostList), new LinkedList<>(), 0);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }

    private static DatacenterBroker createBroker() {
        try {
```

```
            return new DatacenterBroker("Broker");
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }

    private static Vm createVM(int brokerId) {
        return new Vm(0, brokerId, 1000, 1, 1024, 1000, 10000,
                "Xen", new CloudletSchedulerTimeShared());
    }

    private static Cloudlet createCloudlet(int brokerId) {
        UtilizationModel utilization = new UtilizationModelFull();
        return new Cloudlet(0, 40000, 1, 300, 300, utilization, utilization, utilization);
    }
}
```

Step 3: Scheduling Algorithm

Example: **Deadline-Based Scheduling**

- Modify **VmScheduler** to prioritize tasks with earlier deadlines.

```
public class DeadlineVmScheduler extends VmSchedulerTimeShared {

    public DeadlineVmScheduler(List<Pe> peList) {

        super(peList);

    }

    @Override

    public double updateVmProcessing(double currentTime, List<Double> mipsShare)
{

        // Sort cloudlets based on deadline (custom logic)

        // Prioritize cloudlets with nearest deadline

        return super.updateVmProcessing(currentTime, mipsShare);

    }

}
```

Stimulation Outputs

========= Simulation Results =========

Cloudlet 1 finished with status SUCCESS on VM 0 | Deadline: 10.0 | Finish Time: 5.0

Cloudlet 0 finished with status SUCCESS on VM 0 | Deadline: 20.0 | Finish Time: 12.0

Cloudlet 2 finished with status SUCCESS on VM 0 | Deadline: 30.0 | Finish Time: 20.0