Experiment 3. Simulation of Virtual Machines on Hosts in Cloud using Java

Objective:

To simulate the deployment of multiple virtual machines (VMs) on top of cloud hosts (servers) and allocate them based on available CPU capacity using Java.

Concept:

- Hosts have **physical resources** (e.g., CPU cores).
- VMs are deployed on hosts if sufficient CPU is available.
- This simulation follows a **First-Fit allocation strategy**.

Algorithm / Steps:

1. **Start the program.**

2. Create a VirtualMachine class with:
    - VM ID
    - Required CPU cores

3. Create a CloudHost class with:
    - Host ID
    - Total CPU capacity
    - Used CPU tracking
    - Methods: canHost(VM) and deployVM(VM)

4. In the main method:
    - Initialize a list of hosts (e.g., 2 hosts with 8 CPU cores each).
    - Create several VM objects with varying CPU demands.
    - Attempt to deploy each VM to the first available host with enough capacity.
    - Print the result of each allocation.

5. **End the program.**

Code:

```
import java.util.*;

class VirtualMachine {
    int id;
    int requiredCpu;

    public VirtualMachine(int id, int requiredCpu) {
        this.id = id;
        this.requiredCpu = requiredCpu;
    }
}

class CloudHost {
    int id;
    int totalCpu;
    int usedCpu = 0;
    List<Integer> deployedVMs = new ArrayList<>();

    public CloudHost(int id, int totalCpu) {
        this.id = id;
        this.totalCpu = totalCpu;
    }

    public boolean canHost(VirtualMachine vm) {
        return (usedCpu + vm.requiredCpu) <= totalCpu;
    }

    public void deployVM(VirtualMachine vm) {
        usedCpu += vm.requiredCpu;
        deployedVMs.add(vm.id);
        System.out.println("VM " + vm.id + " deployed to Host " + id +
            " | CPU used: " + usedCpu + "/" + totalCpu);
    }
}

public class VMHostSimulation {
    public static void main(String[] args) {
        // Step 1: Create Hosts
        List<CloudHost> hosts = new ArrayList<>();
        hosts.add(new CloudHost(1, 8));
        hosts.add(new CloudHost(2, 8));

        // Step 2: Create VMs
        List<VirtualMachine> vms = new ArrayList<>();
        vms.add(new VirtualMachine(101, 2));
        vms.add(new VirtualMachine(102, 3));
        vms.add(new VirtualMachine(103, 4));
        vms.add(new VirtualMachine(104, 2));
        vms.add(new VirtualMachine(105, 5));

        // Step 3: Allocate VMs to Hosts
        System.out.println("--- VM Allocation Started ---\n");
        for (VirtualMachine vm : vms) {
```

```
            boolean allocated = false;
            for (CloudHost host : hosts) {
                if (host.canHost(vm)) {
                    host.deployVM(vm);
                    allocated = true;
                    break;
                }
            }
            if (!allocated) {
                System.out.println("VM " + vm.id + " could NOT be deployed due to insufficient
CPU.");
            }
        }
        System.out.println("\n--- VM Allocation Finished ---");
    }
}
```

OUTPUT:

```
D:\Softwares\Java\Programs>java VMHostSimulation

--- VM Allocation Started ---

VM 101 deployed to Host 1 | CPU used: 2/8
VM 102 deployed to Host 1 | CPU used: 5/8
VM 103 deployed to Host 2 | CPU used: 4/8
VM 104 deployed to Host 1 | CPU used: 7/8
VM 105 could NOT be deployed due to insufficient CPU.

--- VM Allocation Finished ---
```