Ex_2. Simulate Cloud Resource Allocator using Java

Objective:

To simulate a basic cloud environment in Java, where multiple cloud hosts (servers) are available, and tasks with CPU requirements are dynamically allocated to suitable hosts based on available resources.

📋 **Requirements:**

- Java installed (JDK 8 or above)
- Any IDE or text editor (e.g., Eclipse, IntelliJ, Notepad++)

🔁 **Algorithm / Steps:**

1. **Start the program.**
2. **Define a class CloudTask** to represent a task with:
    - Task ID
    - Required CPU units
3. **Define a class CloudHost** to represent a host/server with:
    - Host ID
    - Total CPU units
    - Used CPU tracking
    - Methods to check if a task can be allocated and to allocate the task
4. **Create multiple CloudHost objects** with fixed CPU capacity.
5. **Create a list of CloudTask objects** with varying CPU demands.
6. **For each task:**
    - Traverse the host list.
    - Check if the host has enough available CPU to allocate the task.
    - If yes, allocate the task and update host usage.
    - If no hosts can accommodate, print a message that the task cannot be allocated.
7. **Display the task allocation result.**
8. **End the program.**

```
import java.util.*;

class CloudTask {
    int id;
    int requiredCpu;

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

class CloudHost {
    int id;
    int totalCpu;
    int usedCpu = 0;

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

    public boolean canAllocate(CloudTask task) {
        return (usedCpu + task.requiredCpu) <= totalCpu;
    }

    public void allocateTask(CloudTask task) {
        usedCpu += task.requiredCpu;
        System.out.println("Task " + task.id + " allocated to Host " + id + " | CPU used: " + usedCpu + "/"
+ totalCpu);
    }
```

```
}

public class CloudResourceAllocator {
    public static void main(String[] args) {
        // Create 2 cloud hosts with 8 CPUs each
        List<CloudHost> hosts = new ArrayList<>();
        hosts.add(new CloudHost(1, 8));
        hosts.add(new CloudHost(2, 8));

        // Create a few tasks
        List<CloudTask> tasks = new ArrayList<>();
        tasks.add(new CloudTask(101, 2));
        tasks.add(new CloudTask(102, 4));
        tasks.add(new CloudTask(103, 3));
        tasks.add(new CloudTask(104, 6));
        tasks.add(new CloudTask(105, 1));

        // Allocate tasks to available hosts
        System.out.println("--- Cloud Task Allocation Started ---");
        for (CloudTask task : tasks) {
            boolean allocated = false;
            for (CloudHost host : hosts) {
                if (host.canAllocate(task)) {
                    host.allocateTask(task);
                    allocated = true;
                    break;
                }
            }
            if (!allocated) {
                System.out.println("Task " + task.id + " could NOT be allocated due to insufficient
resources.");
```

```
            }
        }
        System.out.println("--- Cloud Task Allocation Finished ---");
    }
}
```