**Assignment Week 3**

**Code:**

import java.util.\*;

class Job implements Comparable<Job> {

String name;

int executionTime;

Set<String> requiredResources;

Set<String> dependencies;

int importance;

int startTime;

int remainingTime;

public Job(String name, int executionTime, List<String> requiredResources, List<String> dependencies, int importance) {

this.name = name;

this.executionTime = executionTime;

this.requiredResources = new HashSet<>(requiredResources);

this.dependencies = new HashSet<>(dependencies);

this.importance = importance;

this.remainingTime = executionTime;

}

@Override

public int compareTo(Job other) {

return Integer.compare(other.importance, this.importance);

}

}

class JobScheduler {

private PriorityQueue<Job> jobQueue;

private Set<String> availableResources;

private Map<String, Job> jobMap;

private int currentTime;

public JobScheduler() {

jobQueue = new PriorityQueue<>();

availableResources = new HashSet<>(Arrays.asList("CPU", "GPU"));

jobMap = new HashMap<>();

currentTime = 0;

}

public void scheduleJobs(List<Job> jobs) {

for (Job job : jobs) {

jobMap.put(job.name, job);

}

jobQueue.addAll(jobs);

while (!jobQueue.isEmpty()) {

Job job = findEligibleJob();

if (job == null) {

currentTime++;

continue;

}

executeJob(job);

}

}

private Job findEligibleJob() {

for (Job job : jobQueue) {

if (canExecuteJob(job)) {

jobQueue.remove(job);

return job;

}

}

return null;

}

private boolean canExecuteJob(Job job) {

return job.dependencies.isEmpty() && availableResources.containsAll(job.requiredResources);

}

private void executeJob(Job job) {

job.startTime = currentTime;

availableResources.removeAll(job.requiredResources);

while (job.remainingTime > 0) {

currentTime++;

job.remainingTime--;

Job interruptingJob = findInterruptingJob(job);

if (interruptingJob != null) {

jobQueue.add(job);

System.out.printf("Job %s interrupted at time %d%n", job.name, currentTime);

executeJob(interruptingJob);

return;

}

}

System.out.printf("Job %s started at time %d, finished at time %d, used resources %s%n",

job.name, job.startTime, currentTime, job.requiredResources);

availableResources.addAll(job.requiredResources);

updateDependencies(job);

}

private Job findInterruptingJob(Job currentJob) {

for (Job job : jobQueue) {

if (job.importance > currentJob.importance && canExecuteJob(job)) {

jobQueue.remove(job);

return job;

}

}

return null;

}

private void updateDependencies(Job completedJob) {

for (Job job : jobQueue) {

job.dependencies.remove(completedJob.name);

}

}

}

public class Main {

public static void main(String[] args) {

List<Job> jobs = Arrays.asList(

new Job("Job1", 6, Arrays.asList("CPU"), Arrays.asList(), 3),

new Job("Job2", 4, Arrays.asList("GPU"), Arrays.asList(), 2),

new Job("Job3", 8, Arrays.asList("CPU", "GPU"), Arrays.asList(), 4),

new Job("Job4", 3, Arrays.asList("CPU"), Arrays.asList("Job1"), 1),

new Job("Job5", 5, Arrays.asList("GPU"), Arrays.asList("Job2"), 3),

new Job("Job6", 7, Arrays.asList("CPU", "GPU"), Arrays.asList("Job4"), 2),

new Job("Job7", 2, Arrays.asList("CPU"), Arrays.asList(), 5),

new Job("Job8", 4, Arrays.asList("GPU"), Arrays.asList(), 3),

new Job("Job9", 6, Arrays.asList("CPU", "GPU"), Arrays.asList("Job7", "Job8"), 2),

new Job("Job10", 3, Arrays.asList("CPU"), Arrays.asList("Job1", "Job7"), 1),

new Job("Job11", 5, Arrays.asList("GPU"), Arrays.asList("Job2", "Job8"), 3),

new Job("Job12", 4, Arrays.asList("CPU"), Arrays.asList("Job10"), 2),

new Job("Job13", 6, Arrays.asList("GPU"), Arrays.asList("Job5"), 4),

new Job("Job14", 3, Arrays.asList("CPU", "GPU"), Arrays.asList("Job12", "Job13"), 1),

new Job("Job15", 7, Arrays.asList("CPU"), Arrays.asList("Job3", "Job6"), 3),

new Job("Job16", 5, Arrays.asList("GPU"), Arrays.asList("Job3", "Job9"), 2),

new Job("Job17", 4, Arrays.asList("CPU", "GPU"), Arrays.asList("Job11", "Job14"), 4),

new Job("Job18", 3, Arrays.asList("CPU"), Arrays.asList("Job10", "Job12"), 1),

new Job("Job19", 6, Arrays.asList("GPU"), Arrays.asList("Job13", "Job16"), 3),

new Job("Job20", 2, Arrays.asList("CPU", "GPU"), Arrays.asList("Job17", "Job18"), 2)

);

JobScheduler scheduler = new JobScheduler();

scheduler.scheduleJobs(jobs);

}

}

**Output:**

Job Job7 started at time 0, finished at time 2, used resources [CPU]

Job Job3 started at time 2, finished at time 10, used resources [CPU, GPU]

Job Job8 started at time 10, finished at time 14, used resources [GPU]

Job Job9 started at time 14, finished at time 20, used resources [CPU, GPU]

Job Job1 started at time 20, finished at time 26, used resources [CPU]

Job Job16 started at time 26, finished at time 31, used resources [GPU]

Job Job4 interrupted at time 32

Job Job2 started at time 32, finished at time 36, used resources [GPU]

Job Job5 started at time 36, finished at time 41, used resources [GPU]

Job Job13 started at time 41, finished at time 47, used resources [GPU]

Job Job19 started at time 47, finished at time 53, used resources [GPU]

Job Job11 started at time 53, finished at time 58, used resources [GPU]