DAY 19:

ASSIGNMENT 3:

Task 3: Job Sequencing Problem

Define a class Job with properties int Id, int Deadline, and int Profit. Then implement a function List<Job> JobSequencing(List<Job> jobs) that takes a list of jobs and returns the maximum profit sequence of jobs that can be done before the deadlines. Use the greedy method to solve this problem.

ANSWER:

```
import java.util.ArrayList;

import java.util.Collections;

import java.util.Comparator;

import java.util.List;


class Job {

   int id;

   int deadline;

   int profit;


   public Job(int id, int deadline, int profit) {

      this.id = id;

      this.deadline = deadline;

      this.profit = profit;

   }

}


class JobSequencing {

   public static List<Job> jobSequencing(List<Job> jobs) {

      // Step 1: Sort jobs in descending order of profit

      Collections.sort(jobs, new Comparator<Job>() {
```

```
    @Override
    public int compare(Job job1, Job job2) {
        return job2.profit - job1.profit;
    }
});

// Find the maximum deadline to determine the size of the time slots array
int maxDeadline = 0;
for (Job job : jobs) {
    if (job.deadline > maxDeadline) {
        maxDeadline = job.deadline;
    }
}

// Create an array to keep track of free time slots
int[] timeSlots = new int[maxDeadline + 1];
for (int i = 0; i <= maxDeadline; i++) {
    timeSlots[i] = -1; // -1 means the slot is free
}

// Resulting list of jobs in the maximum profit sequence
List<Job> result = new ArrayList<>();

// Step 2: Iterate over all jobs to schedule them
for (Job job : jobs) {
    // Find a free slot for this job (from its deadline to the beginning)
    for (int j = job.deadline; j > 0; j--) {
        if (timeSlots[j] == -1) {
            timeSlots[j] = job.id;
            result.add(job);
            break;
```

```
                }
            }
        }

        return result;
    }

    public static void main(String[] args) {
        List<Job> jobs = new ArrayList<>();
        jobs.add(new Job(1, 2, 100));
        jobs.add(new Job(2, 1, 19));
        jobs.add(new Job(3, 2, 27));
        jobs.add(new Job(4, 1, 25));
        jobs.add(new Job(5, 3, 15));

        List<Job> result = jobSequencing(jobs);

        System.out.println("Job sequence for maximum profit:");
        for (Job job : result) {
            System.out.println("Job ID: " + job.id + ", Deadline: " + job.deadline + ", Profit: " + job.profit);
        }
    }
}
```

## Explanation:

1. Job Class: Represents a job with id, deadline, and profit.

2. JobSequencing Class:

- jobSequencing Method:
  - Sorts jobs in descending order of profit.
  - Determines the maximum deadline to create the time slots array.

  - Iterates through the jobs and tries to schedule each job in the latest possible time slot before its deadline.

  - Adds the job to the result list if a time slot is found.

- main Method: Demonstrates how to use the jobSequencing method with a different set of jobs, calling the method, and printing the resulting job sequence.

## How the Greedy Method Works:

- Sorting: By sorting jobs by profit, we ensure that we always try to schedule the most profitable jobs first.

- Time Slot Array: We use an array to keep track of which slots (days) are free.

- Scheduling: For each job, we find the latest available slot on or before its deadline and schedule it there if possible.

This example demonstrates how to use the greedy method to solve the Job Sequencing Problem with a different set of jobs to maximize profit within given deadlines.