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.

```
import java.util.ArrayList;
import java.util.Collections;

public class JobSequencingProblem {
	char id;
	int deadline,profit;
	public JobSequencingProblem() {}
	public JobSequencingProblem(char id,int deadline,int profit) {
		super();
		this.id=id;
		this.deadline=deadline;
		this.profit=profit;
	}

	void printJobSequencing(ArrayList<JobSequencingProblem> arr,int t) {
		int n=arr.size();
		Collections.sort(arr,(a,b)->b.profit-a.profit);
		boolean result[]=new boolean[t];
		char job[]=new char[t];
		for(int i=0;i<n;i++){
			for(int j=Math.min(t-1, arr.get(i).deadline-1);j>=0;j--) {
				if(result[j]==false) {
					result[j]=true;
					job[j]=arr.get(i).id;
					break;
				}//if
```

```
                }//for
        }//for
        for(char jb:job) {
                System.out.println(jb+" ");
        }
        System.out.println();

}//fun

public static void main(String[] args) {

        ArrayList<JobSequencingProblem> arr=new ArrayList<JobSequencingProblem>();
        JobSequencingProblem obj= new JobSequencingProblem();
        arr.add(new JobSequencingProblem('a',2,100));
        arr.add(new JobSequencingProblem('b',1,19));
        arr.add(new JobSequencingProblem('c',2,27));
        arr.add(new JobSequencingProblem('d',1,25));
        arr.add(new JobSequencingProblem('e',3,15));

        obj.printJobSequencing(arr,3);

}

}
```