**import** java.util.\*;

**public** **class** Partitioning {

// JOBS: A 0 B 1 C 2 D 3 E 4 F 5 G 6 H 7 I 8

**private** **static** **double** *jobs*[][] = {{9.5, 11},{9.5, 13},{9.5, 11},{11.5, 13},{11, 14},{13.5, 15},{13.5, 15},{14.5, 17},{15.5, 17}};

**private** **static** ArrayList<Integer> *myList*= **new** ArrayList<Integer>();

**private** **static** ArrayList<Integer> *notcompletedList*= **new** ArrayList<Integer>();

**private** **static** ArrayList<Integer> *completedList*= **new** ArrayList<Integer>();

**private** **static** ArrayList<Integer> *toDoArray*= **new** ArrayList<Integer>();

**private** **static** ArrayList<Integer> *conflicting*;//= new ArrayList<Integer>();

**int** jobselected;

**double** starttime,endtime;

**private** **static** **int** *classofjobs*[][],*count*;

**public** Partitioning(**int** len)

{

**for**(**int** k=0;k<len;k++)

{

*toDoArray*.add(k);

*notcompletedList*.add(k);

}

System.***out***.println(*toDoArray*);

*classofjobs* = **new** **int**[len][2];

}

**public** **int** selectShortestEndJob()

{

**int** job= *toDoArray*.get(0);

**double** min=*jobs*[job][1]; //end time is jobs[i][1] start time is jobs[0][i]

System.***out***.println("2. "+*toDoArray*);

**for**(**int** k=0;k<*jobs*.length;k++)

{

**if**(*toDoArray*.contains(k))

{

**if**(*jobs*[k][1]<min)

{

job = k; //job with minimum END TIME

min = *jobs*[k][1];

}

}

}

System.***out***.println("least first END time is : "+ min);

System.***out***.println("Job with least first END time is : "+ job);

**return** job;

}

**public** **void** findconflictingjob(**int** jobselected,**double** start, **double** end)

{

*conflicting*= **new** ArrayList<Integer>();

start = start;

end = end;

**for**(**int** d=0;d<*jobs*.length;d++)

{

**if**(!(*jobs*[d][0] >= end) && d!=jobselected)

{

*conflicting*.add(d);

}

}

}

**public** **int** partitioningInterval(**double**[][] jobs)

{

*count* = 1;

**while**(!*notcompletedList*.isEmpty())

{

**while**(!*toDoArray*.isEmpty())

{

// select shortest end time job get i j of double array

jobselected = selectShortestEndJob();

// eliminate conflicting jobs -> put in todo

starttime = jobs[jobselected][0];

endtime = jobs[jobselected][1];

*classofjobs*[jobselected][0] = jobselected;

*classofjobs*[jobselected][1] = *count*;

System.***out***.println("Count is"+*count*);

**int** index = *toDoArray*.indexOf(jobselected);

System.***out***.println("1. ToDO Array: "+*toDoArray*);

System.***out***.println("1. Array element position: "+index);

*toDoArray*.remove(index);

System.***out***.println("1. ToDO Array: "+*toDoArray*);

// completedList.add(jobselected);

System.***out***.println("Least Job Start : "+ starttime +" and End time is : "+endtime);

findconflictingjob(jobselected,starttime,endtime);

// System.out.println(conflicting);

*toDoArray*.removeAll(*conflicting*);

System.***out***.println("TODO after removing conflicts"+*toDoArray*);

System.***out***.println(*toDoArray*);

}

*count*++;

**for**(**int** s=0; s<*classofjobs*.length;s++)

{

**if**(*classofjobs*[s][1]!=0)

{

**int** index = *notcompletedList*.indexOf(s);

**if**(index!=-1)

{

*notcompletedList*.remove(index);

}

// completedList.add(s);

}

**else**

{

*toDoArray*.add(s);

}

}

System.***out***.println();

}

**return** 0;

}

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

// // JOBS: A 0 B 1 C 2 D 3 E 4 F 5 G 6 H 7 I 8

// double jobs[][] = {{9.5, 11},{9.5, 13},{9.5, 11},{11.5, 13},{11, 14},{13.5, 15},{13.5, 15},{14.5, 17},{15.5, 17}};

Partitioning P = **new** Partitioning(*jobs*.length);

System.***out***.println(*jobs*.length);

**for**(**int** i=0;i<*jobs*.length;i++)

{

**for**(**int** k=0;k<*jobs*[i].length;k++)

System.***out***.print(*jobs*[i][k]+" ");

System.***out***.println();

}

P.partitioningInterval(*jobs*);

System.***out***.println("Jobs : Class");

**for**(**int** k=0;k<*classofjobs*.length;k++)

{

**for**(**int** s=0; s<*classofjobs*[k].length;s++)

System.***out***.print(*classofjobs*[k][s]+" ");

System.***out***.println();

}

}

}