**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> *notcompletedList*= **new** ArrayList<Integer>(); //Array to keep track if task has been allotted a task or not.

**private** **static** ArrayList<Integer> *toDoArray*= **new** ArrayList<Integer>(); //Array to keep track if task can be allotted in same class.

**private** **static** ArrayList<Integer> *conflicting*; //Array to keep track of tasks that cannot be allotted in same class.

**int** jobselected; // Job selected to be present in a Class

**double** starttime,endtime; //Start time and End time of Tasks

**private** **static** **int** *classofjobs*[][],*count*; //count is the Class Number alloted to Jobs/Tasks

// Constructor - adds Jobs/Tasks to Arrays

**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];

}

// Function to Select Shortest End Time Job and Return it

**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;

}

// Function to find and Eliminate Conflicting jobs

**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) //EDIT HERE IF >= or >

{

*conflicting*.add(d);

}

}

}

// Function to perform Interval Partitioning

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

{

*count* = 1;

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

{

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

{

// Call function to Select Shortest End Time Job

jobselected = selectShortestEndJob();

// Call function to Eliminate conflicting jobs

starttime = jobs[jobselected][0];

endtime = jobs[jobselected][1];

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

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

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

// 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);

}

*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);

}

}

**else**

{

*toDoArray*.add(s);

}

}

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

}

}

**public** **void** printclasses()

{

**int** h=1;

**while**(h<=*count*)

{

System.***out***.print("Tasks alloted with class "+h+" are : ");

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

{

**if**(*classofjobs*[i][1]==h)

System.***out***.print(*classofjobs*[i][0]+" ");

}

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

h++;

}

}

// MAIN function where program starts Execution

**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);

// Print all JOBS Start & End Time

System.***out***.println("The total Number of Jobs/Tasks is "+*jobs*.length);

System.***out***.println(" Start End ");

**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();

}

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

// Call Function to perform Interval Partitioning

P.partitioningInterval(*jobs*);

// Print the JOBS & their corresponding CLASSES

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();

}

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

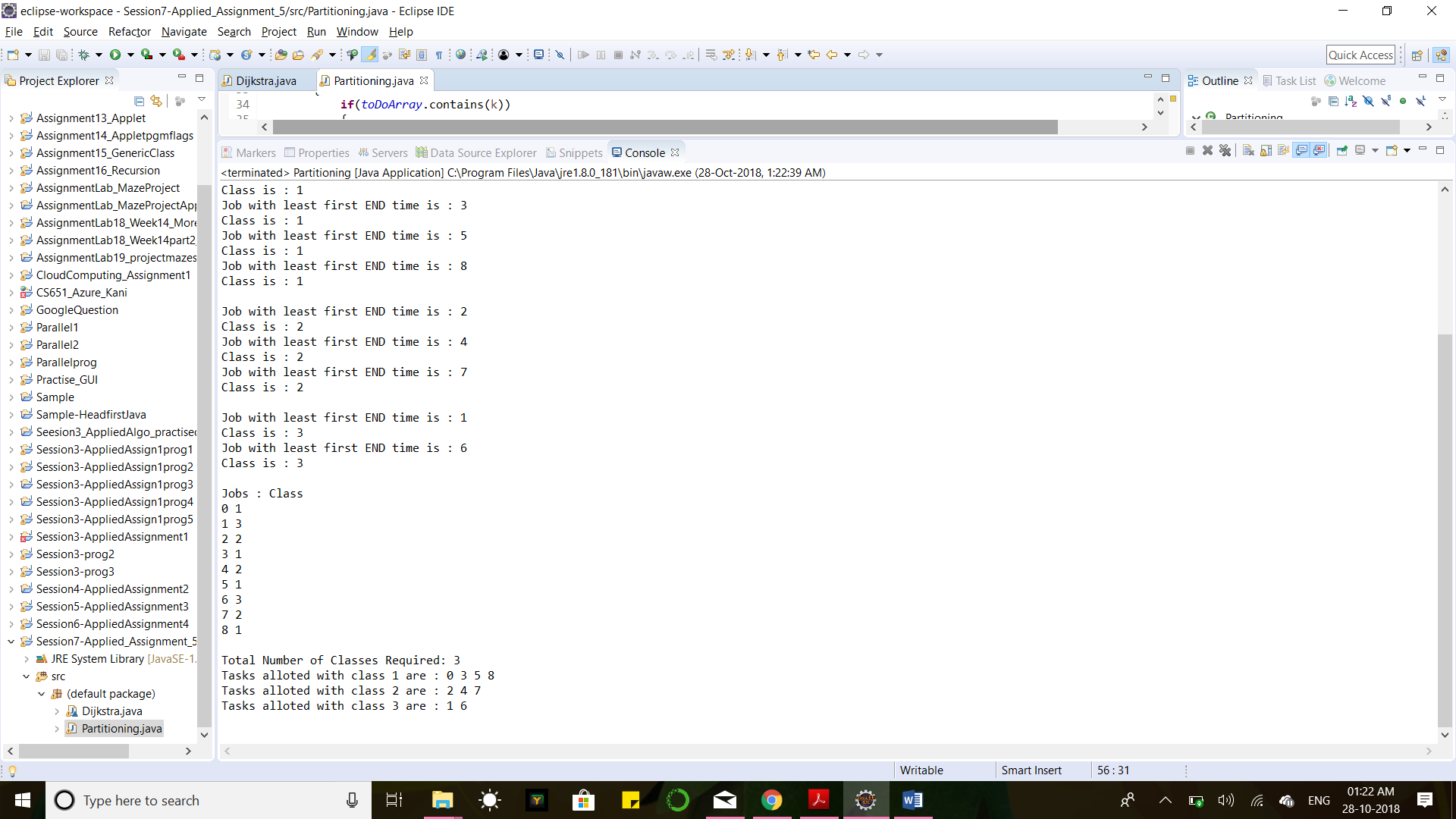
System.***out***.println("Total Number of Classes Required: "+ --*count*);

P.printclasses();

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

}

}



Output:

The total Number of Jobs/Tasks is 9

Start End

9.5 11.0

9.5 13.0

9.5 11.0

11.5 13.0

11.0 14.0

13.5 15.0

13.5 15.0

14.5 17.0

15.5 17.0

Job with least first END time is : 0

Class is : 1

Job with least first END time is : 3

Class is : 1

Job with least first END time is : 5

Class is : 1

Job with least first END time is : 8

Class is : 1

Job with least first END time is : 2

Class is : 2

Job with least first END time is : 4

Class is : 2

Job with least first END time is : 7

Class is : 2

Job with least first END time is : 1

Class is : 3

Job with least first END time is : 6

Class is : 3

Jobs : Class

0 1

1 3

2 2

3 1

4 2

5 1

6 3

7 2

8 1

Total Number of Classes Required: 3

Tasks alloted with class 1 are : 0 3 5 8

Tasks alloted with class 2 are : 2 4 7

Tasks alloted with class 3 are : 1 6

**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> *notcompletedList*= **new** ArrayList<Integer>(); //Array to keep track if task has been allotted a task or not.

**private** **static** ArrayList<Integer> *toDoArray*= **new** ArrayList<Integer>(); //Array to keep track if task can be allotted in same class.

**private** **static** ArrayList<Integer> *conflicting*; //Array to keep track of tasks that cannot be allotted in same class.

**int** jobselected; // Job selected to be present in a Class

**double** starttime,endtime; //Start time and End time of Tasks

**private** **static** **int** *classofjobs*[][],*count*; //count is the Class Number alloted to Jobs/Tasks

// Constructor - adds Jobs/Tasks to Arrays

**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];

}

// Function to Select Shortest End Time Job and Return it

**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;

}

// Function to find and Eliminate Conflicting jobs

**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) //EDIT HERE IF >= or >

{

*conflicting*.add(d);

}

}

}

// Function to perform Interval Partitioning

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

{

*count* = 1;

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

{

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

{

// Call function to Select Shortest End Time Job

jobselected = selectShortestEndJob();

// Call function to Eliminate conflicting jobs

starttime = jobs[jobselected][0];

endtime = jobs[jobselected][1];

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

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

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

// 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);

}

*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);

}

}

**else**

{

*toDoArray*.add(s);

}

}

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

}

}

**public** **void** printclasses()

{

**int** h=1;

**while**(h<=*count*)

{

System.***out***.print("Tasks alloted with class "+h+" are : ");

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

{

**if**(*classofjobs*[i][1]==h)

System.***out***.print(*classofjobs*[i][0]+" ");

}

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

h++;

}

}

// MAIN function where program starts Execution

**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);

// Print all JOBS Start & End Time

System.***out***.println("The total Number of Jobs/Tasks is "+*jobs*.length);

System.***out***.println(" Start End ");

**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();

}

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

// Call Function to perform Interval Partitioning

P.partitioningInterval(*jobs*);

// Print the JOBS & their corresponding CLASSES

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();

}

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

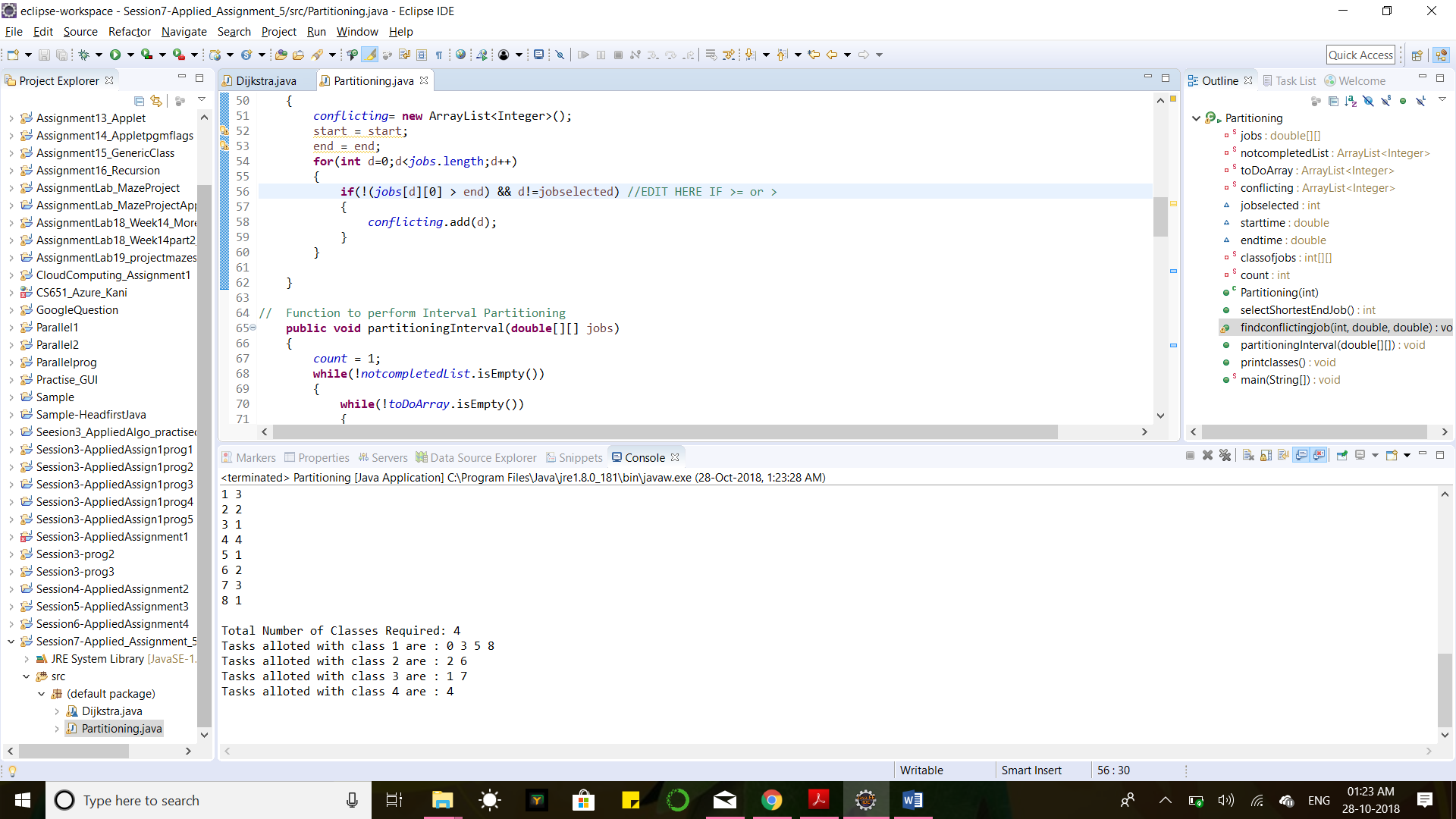
System.***out***.println("Total Number of Classes Required: "+ --*count*);

P.printclasses();

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

}

}



OUTPUT:

The total Number of Jobs/Tasks is 9

Start End

9.5 11.0

9.5 13.0

9.5 11.0

11.5 13.0

11.0 14.0

13.5 15.0

13.5 15.0

14.5 17.0

15.5 17.0

Job with least first END time is : 0

Class is : 1

Job with least first END time is : 3

Class is : 1

Job with least first END time is : 5

Class is : 1

Job with least first END time is : 8

Class is : 1

Job with least first END time is : 2

Class is : 2

Job with least first END time is : 6

Class is : 2

Job with least first END time is : 1

Class is : 3

Job with least first END time is : 7

Class is : 3

Job with least first END time is : 4

Class is : 4

Jobs : Class

0 1

1 3

2 2

3 1

4 4

5 1

6 2

7 3

8 1

Total Number of Classes Required: 4

Tasks alloted with class 1 are : 0 3 5 8

Tasks alloted with class 2 are : 2 6

Tasks alloted with class 3 are : 1 7

Tasks alloted with class 4 are : 4