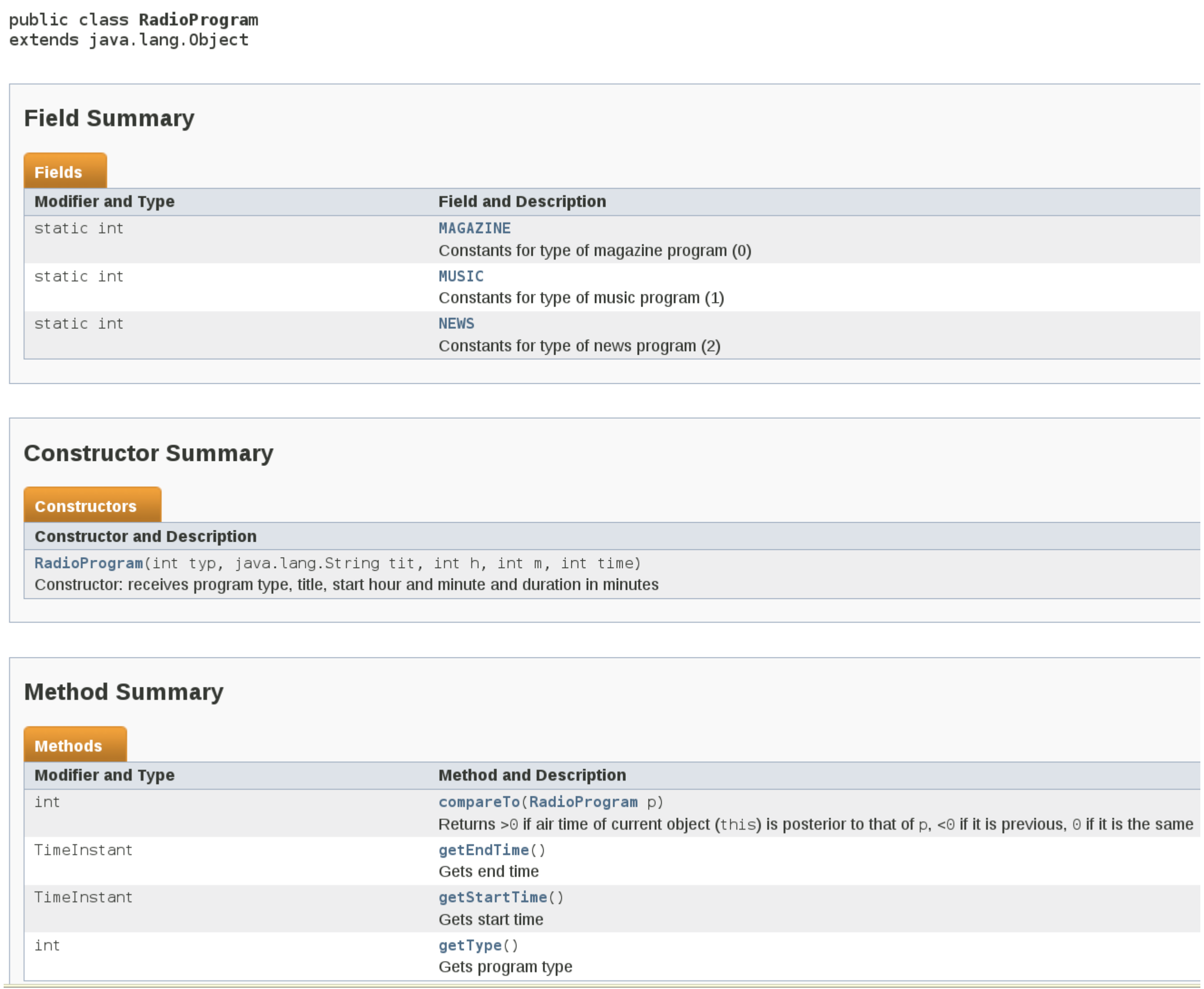
IIP Second Partial (ETSInf)
January 8th, 2018. Time: 2 hours and 30 minutes.

Nota: The exam is marked up to 10 points, but its specific weight in the final grade of IIP is **3.6 points**

NAME:	GROUP:

1. 6 points You have available the `RadioProgram` class, that represents a radio program in a radio station and which is aired between the 00:00 and the 23:59 hours in a given day. Among other data, the class includes its start and end time, title, and type. It is supposed that no radio program starts before midnight and finishes after midnight. This class is known from previous uses and below you have a summary of its documentation, along with the documentation of `TimeInstant`:



```
public class RadioProgram
extends java.lang.Object
```

Field Summary

Fields

Modifier and Type	Field and Description
`static int`	`MAGAZINE` Constants for type of magazine program (0)
`static int`	`MUSIC` Constants for type of music program (1)
`static int`	`NEWS` Constants for type of news program (2)

Constructor Summary

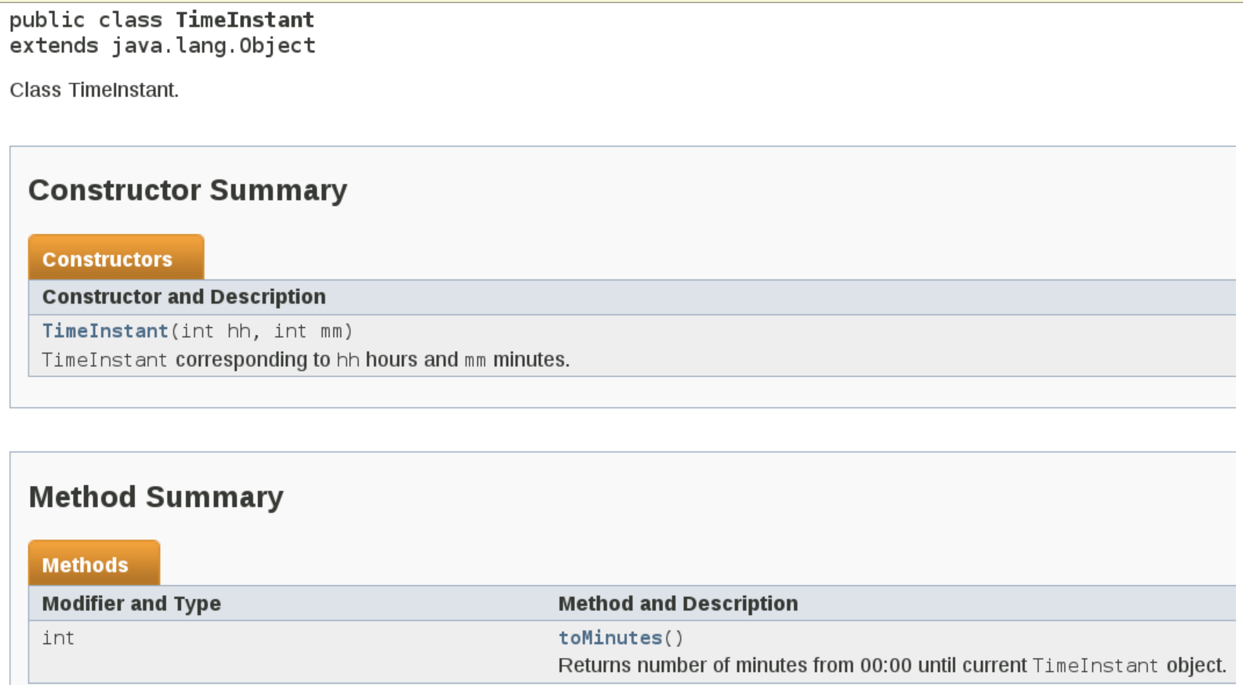
Constructors

Constructor and Description
`RadioProgram(int typ, java.lang.String tit, int h, int m, int time)` Constructor: receives program type, title, start hour and minute and duration in minutes

Method Summary

Methods

Modifier and Type	Method and Description
`int`	`compareTo(RadioProgram p)` Returns >`0` if air time of current object (`this`) is posterior to that of `p`, <`0` if it is previous, `0` if it is the same
`TimeInstant`	`getEndTime()` Gets end time
`TimeInstant`	`getStartTime()` Gets start time
`int`	`getType()` Gets program type



```
public class TimeInstant
extends java.lang.Object
```

Class TimeInstant.

Constructor Summary

Constructors

Constructor and Description
`TimeInstant(int hh, int mm)` `TimeInstant` corresponding to `hh` hours and `mm` minutes.

Method Summary

Methods

Modifier and Type	Method and Description
`int`	`toMinutes()` Returns number of minutes from 00:00 until current `TimeInstant` object.

You must: Implement the `DailySchedule` datatype class, that represents the program schedule for a radio station during a given day, by using the following attributes and methods:

a) (0.5 points) Attributes:

- `MAX_PROGS`: public class integer attribute that indicates the maximum number of programs of a daily schedule, taking into account that minimum program duration is 15 minutes.
- `programs`: private instance array attribute that stores the `RadioProgram` elements of the daily schedule; the programs would be stored in time order from position 0, and no overlaps can appear (i.e., end time of program in position i is the same or previous than start time of program in position $i+1$).
- `nPrograms`: private instance integer attribute that indicates the number of programs in the schedule, that is, the actual number of programs stored in the array `programs`.

b) (0.5 points) A default constructor (without parameters) that creates the array and initialises the current number of programs to 0.

c) (2 points) A method with header:

```
public boolean add(RadioProgram p)
```

that tries to add the program `p` to the schedule (i.e., insert it in the array `programs`), considering that `p` does not overlap without any of the programs present in the array. The method must guarantee that no excesively short programs are included, guaranteeing in that way that the actual number of programs would never exceed the array length. The method must act as follows:

- If the program is shorter than 15 minutes, it must not be inserted.
- In any other case, the program must be inserted in an ordered way, and the programs that go after it must be moved in the array to their next position.

When the program is inserted, the method must return `true`. Otherwise, it must return `false`.

d) (1.5 points) A method with header:

```
public int [] numberOfEachType()
```

that returns an `int` array that stores in each position the number of programs of type equal to that position in the current schedule.
E.g., if the types of programs stored in the array are {`NEWS, NEWS, MUSIC, MAGAZINE, NEWS, MUSIC`} it must return {`1,2,3`}.

e) (1.5 points) A method with header:

```
public int longestProgram() {
```

that returns the position in the array of the program with longest duration. You can assume that at least one radio program is in the schedule. In case that two or more programs have the longest duration, the first of them must be returned.

Solution:

```
public class DailySchedule {
  private RadioProgram [] programs;
  private int nPrograms;
  public static final int MAX_PROGS = 24 * 4;

  public DailySchedule() {
    programs = new RadioProgram[MAX_PROGS];
    nPrograms = 0;
  }

  // Precondition: p does not overlap with any program in array programs
  public boolean add(RadioProgram p) {
    int i, j;

    // Check program length
    if ((p.getEndTime().toMinutes() - p.getStartTime().toMinutes()) < 15) return false;

    // Search position to be inserted and move posterior programs
    i = nPrograms - 1;
    while (i >= 0 && p.compareTo(programs[i]) > 0) {
      programs[i + 1] = programs[i];
      i--;
    }

    // Insert program
    programs[i + 1] = p;

    nPrograms++;

    return true;
  }

  public int [] numberOfEachType() {
    int [] counts = new int[3];
    for (int i = 0; i < nPrograms; i++) counts[programs[i].getType()]++;
    return counts;
  }

  // Precondition: at least 1 program in array programs
  public int longestProgram() {
    int i, pmax, max, aux;

    pmax = 0;
    max = programs[0].getEndTime().toMinutes() - programs[0].getStartTime().toMinutes();

    for (i = 1; i < nPrograms; i++) {
      aux = programs[i].getEndTime().toMinutes() - programs[i].getStartTime().toMinutes();
      if (aux > max) {
        max = aux;
        pmax = i;
      }
    }

    return pmax;
  }

}
```

2. $\boxed{\text{2 points}}$ For $n \geq 0$, the integer functions for exponential k^n and factorial $n!$ can be calculated according to the following recurrences:

$$a_0 = 1 \qquad\qquad b_0 = 1$$
$$a_n = k \cdot a_{n-1}, \quad n > 0 \qquad\qquad b_n = b_{n-1} \cdot n, \quad n > 0$$

a_0	b_0
a_1	b_1
a_2	b_2
...	

```
1        1
3        1
9        2
27       6
81       24
243      120
729      720
2187     5040
```

Factorial function b_n grows faster than exponential function a_n; i.e., from certain n and beyond (the greater the k value, the greater this n value), it happens that $b_n > a_n$.
Write a public class method that receives as parameter an integer $k > 1$ and that shows on the screen, line by line, the different terms (as shown in the above part of the side figure) until showing the first line where factorial is greater than exponential. For example, for $k = 3$, the method must show on the screen what you have below in the side figure. You must not employ methods from Java `Math` library in your solution.

Solution:

```
    public static void compare(int k) {
        int n = 0;
        int a = 1, b = 1;
        System.out.println( a + "\t\t" + b);
        while (a >= b) {
            n++;
            b = b * n;
            a = k * a;
            System.out.println(a + "\t\t" + b);
        }
    }
```

3. 2 points Write a Java public class method that, given an array of `double` with at least one component, returns `true` when all the symmetric values sum the same, or `false` otherwise. You have to consider that, in case that the array has an odd length, the center element is symmetric to itself (i.e., it is summed twice). E.g., for {`1.5,3.0,2.0,3.5`}, {`-0.7,2.2,-0.9,-4.0,-1.1`}, {`3.1,-2.0,-1.0,-6.1`}, {`1.6,4.3`}, or {`1.3`} must return `true`, but for {`1.2,3.3,4.7`}, {`0.5,2.3,4.1,3.4,3.7`}, {`1.5,2.2,4.7,-3.3`}, or {`-1.3,2.7,-2.0,-4.7,-0.7`} must return `false`.

Solution:

```
public static boolean sumSymmetric(double [] a) {
  double sum;
  int i, j;
  i = 0;
  j = a.length - 1;
  sum = a[i] + a[j];
  while (i <= j && sum == a[i] + a[j]) {
    i++;
    j--;
  }
  return (i > j);
}
```