Retake of the Second Midterm IIP Exam – ETSINF
January 21st, 2020 – Duration: 2 hours and 30 minutes

Notice: The maximum mark for this exam is 10 points, but its weight in the final grade of IIP is **3,6 points**

NAME:	GROUP:

1. 6 points The conference on the climate change COP25 has recently been held in the *"Palacio de Congresos de Madrid"*. We have been hired to implement a Java application for managing this summit. Class `Event` is used to represent each one of the activities proposed by the organisers who participate in the summit. The information about an event (exposition or debate) is: starting and ending times, title and organiser's name.

   Next you have a summary of the API documentation of the `Event`[1] class, with its constants and the public methods you need for the implementation of the class `Schedule`:

Fields

Modifier and Type	Field	Description
`static int`	**`DEBATE`**	DEBATE type event with value 0
`static int`	**`EXPOSITION`**	EXPOSITION type event with value 1

Constructors

Constructor	Description
`Event``(TimeInstant start, int dur, java.lang.String org, java.lang.String tit, int type)`	Creates an event `Event` with start time `start`, duration `dur` (in minutes), organiser `org`, event title `tit` and event type `type` (DEBATE or EXPOSITION).

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method	Description
`int`	**`getDuration`**`()`	Returns the duration of the event (in minutes).
`TimeInstant`	**`getEndTime`**`()`	Returns the end time of the event.
`TimeInstant`	**`getStartTime`**`()`	Returns the start time of the event.
`java.lang.String`	**`getTitle`**`()`	Returns the title of the event.
`int`	**`getType`**`()`	Returns the type of the event.
`void`	**`updateTime`**`(TimeInstant newStart)`	The start time of the event is updated to the new time indicated, its end time is also readjusted, but its duration and other attributes remain the same.

   You have to implement the class `Schedule` for representing the schedule of events for the first day of the summit. Attributes to be defined and methods to be implemented are explained next:

   a) (0.5 points) Attributes:

   - `MAX_EVENTS`: constant, public and static attribute of type `int` which indicates the maximum number of events that can be scheduled in one day, its value must be equal to 30.
   - `numEvents`: private and instance attribute of type `int` to indicate how many events are scheduled for the first day of the summit. Its value must be in the range [0 .. `MAX_EVENTS`].
   - `program`: private and instance attribute that is an array of objects of the class `Event`. The size of this array when created in one of the constructors of the class must be `MAX_EVENTS`. The array `program` stores the scheduled events for the first day of the summit, arranged in consecutive positions of the array from 0 to `numEvents-1`.

   b) (0.5 points) A default constructor to create array `program` and reset the number of scheduled events for the first day of the summit.

   c) (1 point) A method with the following profile:

```
public int searchTitle(String title)
```

   that returns the position in the array `program` of the last scheduled event for the COP25 with the given title. If no event with the given title is stored in the array `program`, then this method must return `-1`.

[1] The `Event` class is already implemented, you do not have to write its code as part of the solution.

d) (1.5 points) A method with the following profile:

```
public boolean addEvent(String org, int type, int duration, String title)
```

that tries to add a new event to the schedule of the first day of the COP25. You can assume as a precondition that the value of `type` is 0 or 1. Additionally, you have to take into account the following requirements:

- Debates are limited to 120 minutes as maximum, and expositions to 60 minutes. That is, debates larger than 120 minutes or expositions larger than 60 minutes must be rejected.
- If an event can be accepted according to previous constraint, then it will be added to the end of the schedule, that is, in the first free position of the array `program`. Additionally, the starting time of the new event must be equal to the ending time of the last scheduled event already stored in the array.
- The first event of the day must start at 8 AM o'clock, i.e. at 8:00h.

The method returns `true` if the event has been inserted into the schedule of COP25, otherwise returns `false`.

e) (1.5 points) A method with the following profile:

```
public boolean deleteEvent(String title)
```

that removes from the schedule the event with the given title, and shifts one position to the left in the array `program` all the events that were scheduled after the one that has been removed. Starting and ending times of the re-scheduled events must be updated appropriately in order to avoid time gaps between scheduled events. This method returns `true` if one event with the given title has been removed from the schedule of the first day of the COP25, otherwise returns `false`.

f) (1 point) A method with profile:

```
public int numExpositions()
```

that returns the amount of events of type `Event.EXPOSITION` scheduled for the first day of the COP25.

Solution:

```
public class Schedule {
    public static final int MAX_EVENTS = 30;
    private int numEvents;
    private Event[] program;

    public Schedule() {
        program = new Event[MAX_EVENTS];
        numEvents = 0;
    }

    public int searchTitle(String title) {
        int i = numEvents - 1;
        while (i >= 0 && !program[i].getTitle().equals(title)) { i--; }
        return i;
    }

    /** Precondition: 0 <= type <= 1 */
    public boolean addEvent(String org, int type, int duration, String title) {
        if (numEvents == MAX_EVENTS) { return false; }
        if ((type == Event.DEBATE && duration > 120)
            || (type == Event.EXPOSITION && duration > 60)) { return false; }

        TimeInstant start;
        if (numEvents > 0) { start = program[numEvents - 1].getEndTime(); }
        else { start = new TimeInstant(8, 0); }

        program[numEvents] = new Event(start, duration, org, title, type);
        numEvents++;
        return true;
    }
```

```
    public boolean deleteEvent(String title) {
        int pos = searchTitle(title);
        if (pos == -1) { return false; }
        TimeInstant start = program[pos].getStartTime();
        for (int i = pos; i < numEvents - 1; i++) {
            program[i] = program[i + 1];
            program[i].updateTime(start);
            start = program[i].getEndTime();
        }
        numEvents--;
        program[numEvents] = null;
        return true;
    }

    public int numExpositions() {
        int num = 0;
        for (int i = 0; i < numEvents; i++) {
            if (program[i].getType() == Event.EXPOSITION) { num++; }
        }
        return num;
    }
}
```

2. [2 points] Given an integer number $n > 0$, you have to write an static method for showing on screen all their integer divisors in ascending order. The simplest way to do it is trying to divide n by all the integers in the range $[1, n]$, then, those values for which the remainder is zero are divisors of n. For each value in the range $[1, n]$ that is a divisor of n we can, in fact, obtain two divisors, so it is not necessary to do all possible checks to discover all the divisors, in fact, the process can stop earlier, when $i > n/i$. Clarification, if i is an exact divisor of n, then n/i is an integer that is another divisor of n.

   An example, if $n = 15$, then the sequence of divisions to check is:

$$\begin{array}{r|l} 15 & 1 \\ \hline 0 & 15 \end{array} \quad \begin{array}{r|l} 15 & 2 \\ \hline 1 & 7 \end{array} \quad \begin{array}{r|l} 15 & 3 \\ \hline 0 & 5 \end{array} \quad \begin{array}{r|l} 15 & 4 \\ \hline 3 & 3 \end{array}$$

   END (the process ends because $4 > 3$)

   From the first division we get 1 and 15 as divisors of 15, from the third division we get 3 and 5. Therefore, sorted in ascending order the result is [1 3 5 15], and in this order must be show in the screen.

   Hint: using an array of `boolean` with size $n + 1$ in order to be able to use n as index, then, in the case of the above example using $n = 15$, after applying the proposed algorithm with an array of `boolean` we get:

false	true	false	true	false	true	false	false	false	false	false	false	false	false	false	true
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

   positions with `true` are the divisors of n.

   You have to implement a public and static method that, given the parameter n, shows in the screen the correct result, and returns the array of type `boolean` created to indicate the divisors of n.

Solution:

```
    /** Precondition: n > 0 */
    public static boolean[] divisors(int n) {
        boolean[] a = new boolean[n + 1];

        int d = 1, c = n;
        while (d <= c) {
            if (n % d == 0) {
                a[d] = a[c] = true;
            }
            d++;
            c = n / d;
        }

        for(int i = 1; i <= n; i++) {
            if (a[i]) { System.out.print(i + " "); }
        }
        return a;
    }
```

3. [2 points] **You have to** implement a public and static method to determine if all the characters in a given string `msg` belong to a certain alphabet `alph` represented as an array of type `char`. Both `msg` and `alph` are parameters of the method. The method returns `true` in all the symbols in `msg` belong the the given alphabet, otherwise returns `false`.

   An example, given the alphabet `alph = {'a', 'c', 'g', 't'}`, if `msg` is `"gattaca"` the method must return `true`, but if `msg` is `"gattuca"`, then, must return `false`.

   Notice: recall that the instance method of the class `String`, `charAt(int n)`, returns the symbol of type `char` in the position n of the string stored in an object of the class `String`.

Solution:

```
public static boolean matches(String msg, char[] alph) {
    boolean res = true;
    int i = 0;
    while (i < msg.length() && res) {
        char c = msg.charAt(i);
        int j = 0;
        while (j < alph.length && alph[j] != c) { j++; }
        if (j >= alph.length) { res = false; }
        i++;
    }
    return res;
}
```