IIP second mid term exam (ETSINF)
January 8th, 2020 – duration 2 hours and 30 minutes

Notice: The maximum mark of this individual exam is 10 points, but its weight in the final grade of IIP is **3.75 points**

NAME:	GROUP:

1. 5.5 points It is needed to implement the class `BulletinBoard`, a data-type class, for representing a digital news board to keep updated news. In order to do it, you can use the class `PieceOfNews`, **whose code is already available** so that we can use it to represent news published in digital media the same day news are generated. The elements of the API of the class `PieceOfNews` you can need are in the following figure:

Fields

Modifier and Type	Field	Description
`static int`	`AUDIO`	News type AUDIO with value 0.
`static int`	`TEXT`	News type TEXT with value 2.
`static int`	`VIDEO`	News type VIDEO with value 1.

Constructors

Constructor	Description
`PieceOfNews(TimeInstant i, java.lang.String l, int n, int t)`	Creates a `PieceOfNews` generated at time `i`, stored in the file referenced by the link `l`, echoed by a total of `n` media, and with news type `t` (`AUDIO`, `VIDEO` or `TEXT`).

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

Modifier and Type	Method	Description
`int`	`compareTo (PieceOfNews other)`	Returns a negative integer if the news stored in `this`, the current object, is less popular than the news stored in `other`, returns a positive integer if the news stored in `this` is more popular than the one stored in `other`, otherwise returns `0` for indicating that both news are equally popular.
`TimeInstant`	`getInstant()`	Returns the time instant when the news was generated.
`java.lang.String`	`getLink()`	Returns the link where the news is stored.
`int`	`getType()`	Returns the news type.

You have to implement the class `BulletinBoard` with the attributes and methods described next. For the sake of simplicity, you can assume this class will be in the same package class `PieceOfNews` is.

a) (0.5 points) Attributes:

- `MINUTES`: public class constant of type `int` that must be static and whose value must be the total amount of minutes in a single day ($1440 = 24 * 60$).
- `bBoard`: private instance attribute that must be an array for storing objects of the class `PieceOfNews` and whose length must be just the amount of minutes in a whole day. Then, any news stored in the array will be located at the position indexed by the minute within a day the news was generated. So the index i of any news must be in the range $[0, 1439]$, i.e. $0 \leq i <$ `MINUTES`. In other words, `bBoard[i]` will contain the reference to a news generated in the minute i within a day or `null` if no news were generated at minute i.
  Figure 1 is a representation of an object of the class `BulletinBoard`, referenced in this case by the variable `bB`, where we can see three examples of news: news `a` generated at 00:00 stored in `bBoard[0]`, news `b` generated at 01:01 stored in `bBoard[61]`, and news `c` generated at 23:59 stored in `bBoard[1439]`.
- `numPerType`: private instance attribute that must be an array of type `int` and whose length must be just 3. This is an array of counters that must be kept updated in order to know, at any moment, how many news of each of the three types are stored in the array `bBoard`. Therefore, `numPerType[0]`, `numPerType[1]` and `numPerType[2]` are the counters of news of types `AUDIO`, `VIDEO` and `TEXT`, respectively.
  According to Figure 1, there are two news of type `AUDIO` and one news of type `TEXT` stored in the array `bBoard`, that is why the values of `numPerType[0]`, `numPerType[1]` and `numPerType[2]` are 2, 0 and 1 respectively.

b) (0.75 points) Constructor for creating objects of the class `BulletinBoard` without news, i.e. with all the positions of the array `bBoard` referencing to `null`.

c) (1.5 points) Method `add()` with the following profile:

```
public boolean add(PieceOfNews p)
```

that tries to add `p` at the corresponding position of the array `bBoard`. But if the position is not empty, then, both `p` and the previously existing news must be compared by using the method `compareTo()` of the class `PieceOfNews`. The most popular news is the one that must remain in the array. If `p` is newer than the existing one, then `p` must

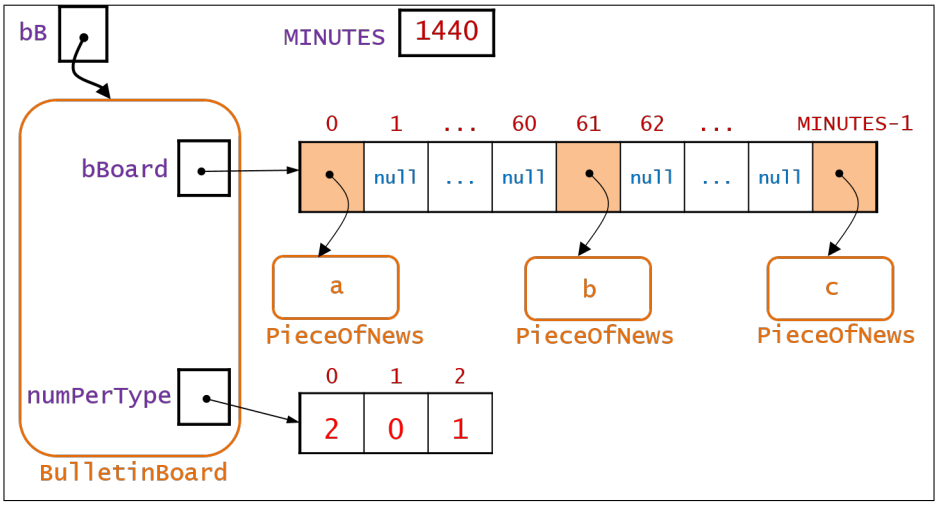

Figure 1: Representation of an example of object of the class `BulletinBoard`.

be stored in the array and the method must return `true`, otherwise the array is not modified and the method must return `false`. Additionally, this method must update the attribute `numPerType` accordingly.

Hint: class `TimeInstant` has the method `toMinutes()` that returns the number of minutes within a day from 00:00 to the corresponding instant represented by the current object. Therefore, this method can be used to get the minute within the day when the news was generated and, so, used as the index of the position of the array `bBoard` corresponding to the news.

d) (1.25 points) Method `isPosted()` with the following profile:

```
public PieceOfNews isPosted(String link)
```

that returns the news stored in `link`. This method must return `null` if no news in the array have this `link`.

e) (1.5 points) Method `filterByType()` with the following profile:

```
public PieceOfNews[] filterByType(int type)
```

that returns an array with all the news of the given `type` generated in the day. Obviously, `type` must be in the range $[0, 2]$, that is, the value of `type` can be one of `PieceOfNews.AUDIO`, `PieceOfNews.VIDEO` or `PieceOfNews.TEXT`.

Important: the size of the returned array must be equal to the amount of news of the give `type` generated in the day. The size of the array can be 0 if no news of the specified `type` were generated in the day.

Solution:

```
 */
public class BulletinBoard {
    public static final int MINUTES = 1440;
    private PieceOfNews[] bBoard;
    private int[] numPerType;

    public BulletinBoard() {
        bBoard = new PieceOfNews[MINUTES];
        numPerType = new int[3];
    }

    public boolean add(PieceOfNews p) {
        boolean added = false;
        int position = p.getInstant().toMinutes();
        if (bBoard[position] == null || p.compareTo(bBoard[position]) > 0) {
            if (bBoard[position] != null) { numPerType[bBoard[position].getType()]--; }
            bBoard[position] = p;
            numPerType[p.getType()]++;
            added = true;
        }
        return added;
    }

    public PieceOfNews isPosted(String link) {
        PieceOfNews result = null;
        boolean found = false;
        int i = 0;
        while (i < bBoard.length && !found) {
            if (bBoard[i] != null && link.equals(bBoard[i].getLink())) {
```

```
                found = true;
                result = bBoard[i];
            }
            i++;
        }
        return result;
    }

    /** Precondition: 0 <= type < 3 */
    public PieceOfNews[] filterByType(int type) {
        PieceOfNews[] result = new PieceOfNews[numPerType[type]];
        for (int i = 0, j = 0; i < bBoard.length && j < numPerType[type]; i++) {
            if (bBoard[i] != null && bBoard[i].getType() == type) {
                result[j++] = bBoard[i];
            }
        }
        return result;
    }
```

2. 2.25 points **You have to write the code** of a public and static method that, given an array of positive integers, searches and returns the first repeated value in the array. Repeated values can be located at any position, in other words, repeated values can appear separated. That two repeated values appear in consecutive positions of the array is just one of the many possibilities.

   If no repeated values are found, the method must return -1. According to the following example, in the case of array `a1` the method will return 2, as 2 is the first repeated value, but in the case of array `a2` the method will return -1 because there are no repeated values.

	0	1	2	3	4	5	6	7	8	9
a1	5	2	8	3	7	4	2	7	9	4

	0	1	2	3	4	5	6	7	8	9
a2	5	2	8	3	7	4	6	0	9	1

Solution:

```
    /**
     * Given an array of positive integers,
     * returns the value of the first repeated element
     * in any other position of the array,
     * otherwise returns <code>-1</code> for indicating
     * no repeated elements exist in the array.
     */
    /** Precondition: v is an array of positive integers. */
    public static int firstRepeatedElement(int[] v) {
        int i = 0, j;
        boolean found = false;
        while (i < v.length - 1 && !found) {
            j = i + 1;
            while (j < v.length && !found) {
                found = v[i] == v[j];
                j++;
            }
            i++;
        }
        if (found) { return v[i - 1]; }
        else { return -1; }

        /* Alternative code
        for( int i=0; i < v.length-1; i++ ) {
            for( int j=i+1; j < v.length; j++ ) {
                if ( v[i] == v[j] ) return v[i];
            }
        }
        return -1;
        */
    }
```

3. 2.25 points Given a real value x and the following recurrence:

$$C_1 = x \quad ; \quad C_n = C_{n-1} * n \quad \forall n > 1$$

You have to write the code of a public and static method that given two parameters of type double, x and ϵ, with $\epsilon \in [0, 1[$:

- returns -1 if $x \leq 0$;
- otherwise, returns how many terms must be computed until $\frac{1}{C_n - C_{n-1}} \leq \epsilon$ is evaluated to `true`.

Hint: in the code use `epsilon` as the name of the variable instead of the symbol ϵ.

Solution:

```
/** Precondition: 0 <= epsilon < 1 */
public static int numTerms(double x, double epsilon) {
    if (x <= 0) { return -1; }
    double cPrev = x;
    double cActual = cPrev * 2;
    int i = 2;
    while (1 / (cActual - cPrev) > epsilon) {
        i++;
        cPrev = cActual;
        cActual = cPrev * i;
    }
    return i;
}
```