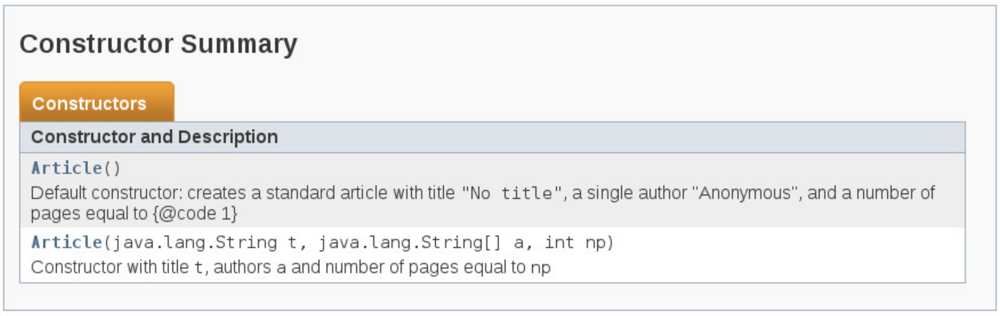
IIP Retake of Second Partial - ETSInf
January 28th of 2016. Time: 2 hours and 30 minutes.

1. $\boxed{\text{6.5 points}}$ You have available the `Article` class, that represents an article in a published number of a journal. Each article is defined based on three data items: title of the article (`String`), authors (array of `String`), and number of pages (`int`). This class documentation summary is the one at the side.

You must: implement the `Journal` class, that represents the number of a journal with its corresponding articles, by using the attributes and methods defined below.

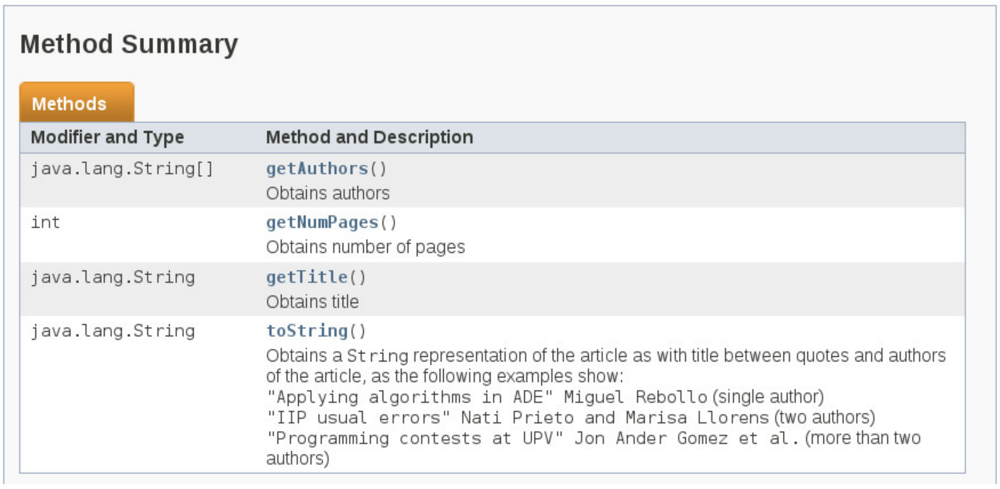
Remember you have to use the constants you are asked to define whenever required.



Constructor Summary

Constructors

Constructor and Description
`Article()` Default constructor: creates a standard article with title `"No title"`, a single author "Anonymous", and a number of pages equal to {@code 1}
`Article(java.lang.String t, java.lang.String[] a, int np)` Constructor with title `t`, authors `a` and number of pages equal to `np`



Method Summary

Methods

Modifier and Type	Method and Description
`java.lang.String[]`	`getAuthors()` Obtains authors
`int`	`getNumPages()` Obtains number of pages
`java.lang.String`	`getTitle()` Obtains title
`java.lang.String`	`toString()` Obtains a `String` representation of the article as with title between quotes and authors of the article, as the following examples show: `"Applying algorithms in ADE" Miguel Rebollo` (single author) `"IIP usual errors" Nati Prieto and Marisa Llorens` (two authors) `"Programming contests at UPV" Jon Ander Gomez et al.` (more than two authors)

a) (0.75 points) Attributes:

- `MAX_ARTICLES`, class (`static`) constant atttribute that represents the maximum number of articles that can have a journal, with value 20.
- `MAX_PAGES`, class (`static`) constant atttribute that represents the maximum number of pages of a journal, with value 200.
- `name`, `String` with the name of the journal.
- `num`, positive integer with the number of the journal.
- `articles`, array of `Article`, with length `MAX_ARTICLES`, to store the articles of the journal; they are placed in consecutive positions of the array, from 0 to `nArticles - 1` (both included).
- `nArticles`, integer in the interval `[0, MAX_ARTICLES]` which represents the number of articles of the journal.
- `nPages`, integer in the interval `[0, MAX_PAGES]` which represents the number of pages of the journal.

b) (0.5 points) Implement a constructor that receives as parameters the journal name and number (supposed to be valid) and creates a journal number with that name and number and with empty contents (no articles and number of pages equal to 0).

c) (1 point) Write a method with header:

```
public Article search(String t)
```

such that, given the title of an article, returns the corresponding `Article` object, or `null` if it is not present in the journal.

d) (0.75 points) Write a method with header:

```
public boolean add(Article a)
```

such that adds the article `a` to the journal if the resulting journal accomplishes the conditions for maximum number of articles and maximum number of pages. You can suppose that the given article is not present in the journal. Method must return `true` if addition was possible and `false` otherwise.

e) (1 point) Write a method with header:

```
public Article shortestArticle()
```

such that returns the article with lowest number of pages; in case different articles present that lowest number, the first of them must be returned; you can suppose at least an article is present.

f) (1.5 points) Write a method with header:

```
public Article [] singleAuthor()
```

such that returns the set of the `Article` objects (array of `Article`) in the journal with a single author. Array size must be 0 if no single author articles are present.

g) (1 point) Write a method with header:

```
public String tableOfContents()
```

such that returns a `String` with the following structure:

- First line: it will contain journal name and number (as shown in the example below).
- Next lines: it will contain the description of the corresponding article (given by `toString()` method of `Article`), followed by a dash (`-`) and its starting page (as shown in the example below); the first page of the first article is page number 3.

You can suppose that there is at least one article in the journal.
For example:

```
Programming at UPV - Number 5
"IIP usual errors" Nati Prieto and Marisa Llorens - 3
"Programming contests at UPV" Jon Ander Gomez et al. - 15
"A tablet experience for programming" Mabel Galiano - 22
"Different ways of learning if and if-else" Assump Casanova and Paco Marques - 34
"Retrocomputing: ancient language descriptions" Jorge Gonzalez and Quique Ramos - 51
"What's the last? FIFO structures basics" Carlos Herrero et al. - 59
"Applying algorithms in ADE" Miguel Rebollo - 78
```

Solution:

```
/**
 * Class Journal: represents a journal with different articles
 *
 * @author IIP Exam
 * @version Retake Second Partial - Year 2015-2016
 */
public class Journal {
  /** Constant that represents maximum number of articles
    * ({@code int}) of a journal
    */
  public static final int MAX_ARTICLES = 20;
  /** Constant that represent maximum number of pages
    * ({@code int}) in a number of the journal
    */
  public static final int MAX_PAGES = 200;

  private String name;              // Name of the journal
  private int num;                  // Number of the issue
  private Article [] articles;      // Set of articles
  private int nArticles;            // Number of articles
  private int nPages;               // Number of pages
```

```
/** Constructor: creates new issue of number {@code nu} with
  * journal name {@code na} and no articles (0 articles, 0 pages)
  * @param na Name of the journal ({@code String})
  * @param nu Number of current issue ({@code int}), nu >= 0
  */
public Journal(String na, int nu) {
  name = new String(na);
  num = nu;
  articles = new Article[MAX_ARTICLES];
  nArticles = 0;
  nPages = 0;
}

/** Search an article given its title
  * @param t String with the title
  * @return Article, article of that title or null is not found
  */
public Article search(String t) {
  int i = 0;
  while (i < nArticles && !articles[i].getTitle().equals(t)) i++;
  if (i < nArticles) return articles[i];
  return null;
}

/** Adds an article if possible: no more than MAX_ARTICLES articles,
  * and no more than MAX_PAGES pages in the resulting journal issue
  * PRECONDITION: article {@code a} not present in the journal
  * @param a Article to be added
  * @return boolean, {@code true} it it was added, {@code false} otherwise
  */
public boolean add(Article a) {
  if (nArticles < MAX_ARTICLES && (nPages + a.getNumPages()) <= MAX_PAGES) {
    articles[nArticles] = a;
    nArticles++;
    nPages += a.getNumPages();
    return true;
  }
  return false;
}

/** Returns shortest article (first one if more than one) in the journal
  * PRECONDITION: at least one article in the journal issue
  * @return Article, that with less pages in the issue
  */
public Article shortestArticle() {
  int s = 0, i;

  for (i = 1; i < nArticles; i++)
    if (articles[i].getNumPages() < articles[s].getNumPages()) s = i;

  return articles[s];
}

/** Returns array of Articles that have a single author with a
  * length equal to the number of articles with a single author
  * (could be 0)
  * @return Article[], array with articles with single author
  */
public Article [] singleAuthor() {
  int na = 0, i, j;
  for (i = 0; i < nArticles; i++)
    if (articles[i].getAuthors().length == 1) na++;

  Article [] singleAutArt = new Article[na];

  j = 0;
  for (i = 0; i < nArticles && j < na; i++)
    if (articles[i].getAuthors().length == 1) {
      singleAutArt[j] = articles[i];
      j++;
    }

  return singleAutArt;
}
```

```
  /** Returns a {@code String} with the table of contents of the
    * journal issue, indicating: <br>
    * - At first line, journal name and issue number
    * - In each following line, a description of the corresponding
    *   article of the journal (given by its {@code toString()} method,
    *   followed by "-" and with its initial page number; first article
    *   starts at page 3
    * PRECONDITION: at least there is an article in the journal
    * @return String, table of contents of the journal issue
    */
  public String tableOfContents() {
    String s = name + " - Number " + num + "\n";
    int page = 3;

    for (int i = 0; i < nArticles; i++) {
      s += articles[i] + " - " + page + "\n";
      page = page + articles[i].getNumPages();
    }

    return s;
  }
}
```

2. 1.5 points Write a Java class (`static`) method that, given a `String s` as parameter, shows on the screen a drawing with as many lines as letters has the `String` with two diagonals that join in the first line and diverge towards the last line, such that in each line the corresponding character of the `String` parameter is shown in each diagonal position (i.e., an inverted 'V'). You can suppose that the `String` parameter length is greater than 1.
NOTE: the example shows a grid to make clearer blank spaces between the different characters.

E.g., for s=`"Java"`:

			J	J			
		a			a		
	v					v	
a							a

Solution:

```
/** Write on standard output a drawing with n lines
 *  formed by two diagonals with the letters of the String parameter
 *  (with length greater than 1) forming an inverted 'V'
 */
 public static void writeMount(String s) {
     int i, j;
     for (i = 0; i < s.length(); i++) {
         for (j = i + 1; j < s.length(); j++) {
              System.out.print(" ");
         }
         System.out.print(s.charAt(i));
         for (j = 0; j < i * 2; j++) {
             System.out.print(" ");
         }
         System.out.println(s.charAt(i));
     }
 }
```

3. 2 points Write a Java class (`static`) method that, given an integer non-negative number, returns an array of `boolean`, of length equal to number of digits of the parameter, such that position `i` is `true` when digit at position `i` (starting from the left) is multiple of 3, and `false` otherwise.

For example:

For 357 must return

0	1	2
true	false	false

For 1609 must return

0	1	2	3
false	true	true	true

Solution:

```
/** Returns array of boolean indicating if each digit
 *  of parameter n (n>0) is multiple of 3 or not
 */
public static boolean [] intToBool(int n) {
    int s = 1, copy = n;

    while (copy > 9) {
        s++;
        copy = copy / 10;
    }

    boolean [] a = new boolean[s];

    while (s > 0) {
        s--;
        a[s] = ((n % 10) % 3) == 0;
        n = n / 10;
    }

    return a;

}
```