Recuperación del Segundo Parcial de IIP - ETSInf
Fecha: 28 de Enero de 2016. Duración: 2:30 horas.

1. 6.5 puntos Se dispone de la clase `Articulo`, que representa un artículo de una revista en base a tres datos: título (`String`), autores (`String[]`) y número de páginas (`int`). La siguiente es su documentación:

Constructor Summary

Constructors

Constructor and Description
`Articulo()` Crea un articulo estandar, esto es, de titulo `"Sin titulo"`, con un unico autor `"Anonimo"` y un numero de paginas igual a `1`
`Articulo(java.lang.String t, java.lang.String[] a, int n)` Crea un articulo de titulo `t`, con autores `a` y un numero de paginas igual a `n`

Method Summary

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

Modifier and Type	Method and Description
`java.lang.String[]`	`getAutores()` Devuelve los autores (array de `String`) de un articulo
`int`	`getNumPags()` Devuelve el numero de paginas (`int`) de un articulo
`java.lang.String`	`getTitulo()` Devuelve el titulo (`String`) de un articulo

`java.lang.String`	`toString()`
	Devuelve un `String` con el titulo (entrecomillado) y autores (uno, dos o mas) de un articulo, tal como muestran los siguientes ejemplos:
	`"Aplicacion de algoritmos en ADE" Miguel Rebollo`
	`"Errores tipicos en IIP" Nati Prieto y Marisa Llorens`
	`"Concursos de Programacion en la UPV" Jon Ander Gomez et al.`

Para confeccionar una revista a partir de un conjunto de artículos, **se pide** implementar la clase `Revista` mediante los atributos y métodos que se indican a continuación; al hacerlo, se deben usar las constantes definidas en la clase siempre que se requiera.

a) (0.75 puntos) Atributos:

- `MAX_ARTICULOS`, constante de clase (`static`) que representa el número máximo de artículos que puede contener una revista: `20`.
- `MAX_PAGINAS`, constante de clase (`static`) que representa el número máximo de páginas que puede tener una revista: `200`.
- `nombre`, `String` que representa el nombre de una revista.
- `num`, `int` positivo que representa el número (de ejemplar) de una revista.
- `nArticulos`, `int` en el intervalo `[0, MAX_ARTICULOS]` que representa el número actual de artículos que contiene una revista.
- `nPags`, `int` en el intervalo `[0, MAX_PAGINAS]` que representa el número actual de paginas de una revista.
- `articulos`, un array de tipo base `Articulo` y capacidad `MAX_ARTICULOS`, donde se almacenan los artículos de una revista secuencialmente, en posiciones consecutivas desde la `0` a la `nArticulos - 1`, ambas inclusive.

b) (0.5 puntos) Un constructor con parámetros, que crea el ejemplar número `numR` de una revista (de nombre) `nomR` con `0` artículos y `0` páginas (vacío). Se puede suponer `numR>0` (Precondición).

c) (1 punto) Un método con perfil:

```
public Articulo buscar(String t)
```

que devuelve el artículo con título `t` de una revista, o `null` si la revista no contiene tal articulo.

d) (0.75 puntos) Un método con perfil:

```
public boolean incluir(Articulo a)
```

que añade el artículo `a` a una revista y devuelve `true` si la revista resultante cumple las condiciones de número máximo de artículos y páginas; sino, si `a` no "cabe" en la revista, el método devuelve `false`. Se puede suponer que `a` no forma parte de la revista (Precondición).

e) (1 punto) Un método con perfil:

```
public Articulo masCorto()
```

que devuelve el artículo con menor número de páginas de una revista, el primero que aparece si hay más de uno con tal número mínimo de hojas. Se puede suponer que la revista contiene al menos un artículo (Precondición).

f) (1.5 puntos) Un método con perfil:

```
public Articulo[] conUnAutor()
```

que devuelve un array con aquellos artículos de una revista que tengan un solo autor; así, la longitud de este array será `0` si la revista no contiene ningún artículo de tal tipo.

g) (1 punto) Un método con perfil:

```
public String sumario()
```

que devuelve un `String` que representa el sumario de una revista, por lo que contiene:

- En su primera línea, el nombre y número de la revista.
- En cada una de las siguientes líneas, la descripción de un artículo de la revista (dada por el método `toString` de la clase `Articulo`) seguida de un guión y del número de página donde empieza dicho artículo, considerando que el primer artículo de la revista empieza en su página número `3`.

Se puede suponer que la revista contiene al menos un artículo (Precondición).

El siguiente ejemplo ilustra las características de este `String` sumario con detalle:

```
La Programación en la UPV - Numero 5
"Errores tipicos en IIP" Nati Prieto y Marisa Llorens - 3
"Concursos de programacion en la UPV" Jon Ander Gomez et al. - 15
"Aplicacion de algoritmos en ADE" Miguel Rebollo - 22
"Diferentes formas de aprender if e if-else" Assum Casanova y Paco Marques - 34
"Retrocomputing: descripcion de lenguajes antiguos" Jorge Gonzalez y Quique Ramos - 51
"¿Quien es el ultimo? Fundamentos de estructuras FIFO" Carlos Herrero et al. - 59
```

Solución:

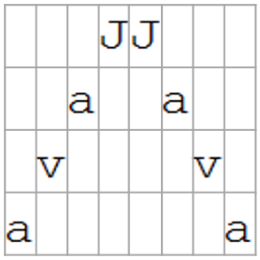
```
/** Clase Revista: representa una revista con diferentes articulos */
public class Revista {
    /** Numero ({@code int}) maximo de articulos y paginas de una revista */
    public static final int MAX_ARTICULOS = 20, MAX_PAGINAS = 200;
    private String nombre;          // Una Revista TIENE UN nombre
    private int num;                //             UN numero (de ejemplar)
    private Articulo[] articulos;   //             UNA lista de articulos
    private int nArticulos, nPags;  //             UN nro. de articulos y pags.
    /** PRECONDICION: {@code numR > 0}
     *  Crea el ejemplar numero {@code numR} de una revista de nombre
     *  {@code nomR} con {@code 0} articulos y {@code 0} paginas (vacio) */
    public Revista(String nomR, int numR) {
        nombre = nomR;
        num = numR;
        nArticulos = 0;
        nPags = 0;
        articulos = new Articulo[MAX_ARTICULOS];
    }
```

```
    /** Devuelve el articulo con titulo {@code t} de una revista, o {@code null}
     *  si la revista no contiene tal articulo */
    public Articulo buscar(String t) {
        int i = 0;
        while (i < nArticulos && !articulos[i].getTitulo().equals(t)) { i++; }
        if (i < nArticulos) { return articulos[i]; }
        return null;
    }
    /** PRECONDICION: la revista no contiene el articulo {@code a}
     *  Incluye el articulo {@code a} en una revista y devuelve {@code true}
     *  si la revista resultante cumple las condiciones de numero maximo de
     *  articulos y paginas; sino, el metodo devuelve {@code false} */
    public boolean incluir(Articulo a) {
        int nPagsA = a.getNumPags();
        if (nArticulos < MAX_ARTICULOS && (nPags + nPagsA) <= MAX_PAGINAS) {
            articulos[nArticulos++] = a;
            nPags += a.getNumPags();
            return true;
        }
        return false;
    }
    /** PRECONDICION: la revista contiene al menos un articulo
     *  Devuelve el articulo con menor numero de paginas de una revista,
     *  el 1ero que aparezca si mas de uno tiene tal numero minimo de hojas */
    public Articulo masCorto() {
        int posMin = 0;
        for (int i = 1; i < nArticulos; i++) {
            if (articulos[i].getNumPags() < articulos[posMin].getNumPags()) {
                posMin = i;
            }
        }
        return articulos[posMin];
    }
    /** Devuelve un array con aquellos articulos de una revista que tengan
     *  un solo autor, por lo que su longitud puede ser cero */
    public Articulo[] conUnAutor() {
        int num1Autor = 0;
        for (int i = 0; i < nArticulos; i++) {
            if (articulos[i].getAutores().length == 1) { num1Autor++; }
        }
        Articulo[] res = new Articulo[num1Autor];
        int j = 0;
        for (int i = 0; j < num1Autor; i++) {
            if (articulos[i].getAutores().length == 1) {
                res[j++] = articulos[i];
            }
        }
        return res;
    }
    /** PRECONDICION: la revista contiene al menos un articulo
     *  Devuelve un {@code String} que representa el sumario de una revista,
     *  por lo que contiene: en su 1era linea, el nombre y numero de la
     *  revista; en cada una de las siguientes lineas, el {@code toString}
     *  de cada articulo de la revista seguido de un guion y del numero
     *  de pagina donde empieza dicho articulo, considerando que el 1er
     *  articulo de la revista empieza en su pagina numero 3 */
    public String sumario() {
        String res = nombre + " - Numero " + num + "\n";
        int numPag = 3;
        for (int i = 0; i < nArticulos; i++) {
            res += articulos[i] + " - " + numPag + "\n";
            numPag = numPag + articulos[i].getNumPags();
        }
        return res;
    }
}
```

2. 1.5 puntos **Se pide** implementar un método de clase (`static`) que, dado un `String s` de longitud mayor que `1`, muestre por pantalla una figura que, como se aprecia en el ejemplo contiguo para el `String s = "Java"`, tenga tantas líneas como caracteres tenga `s`, con dos diagonales que se junten en la primera línea y diverjan hacia la última (como en una `'V'` invertida) y de forma que el carácter que aparezca en cada una de ellas en la `i`-ésima línea sea el `i`-ésimo carácter de `s`.

   NOTA: la figura del ejemplo aparece dentro de una cuadrícula para que se puedan distinguir mejor los espacios en blanco que la conforman.

`s = "Java"`



Solución:

```
/** PRECONDICION: s.length() > 1 */
 public static void dibujarFigura(String s) {
     for (int i = 0; i < s.length(); i++) {
         for (int j = i + 1; j < s.length(); j++) {
             System.out.print(" ");
         }
         System.out.print(s.charAt(i));
         for (int j = 0; j < i * 2; j++) {
             System.out.print(" ");
         }
         System.out.println(s.charAt(i));
     }
 }
```

3. 2 puntos **Se pide:** implementar un método de clase (`static`) que, dado un entero `n`$\geq$`0`, devuelva un array de `boolean` tal que: su longitud sea igual al número de dígitos de `n`; su `i`-ésima componente valga `true` cuando el `i`-ésimo dígito de `n` (numerando de izquierda a derecha) sea múltiplo de `3` y `false` cuando no lo sea. Por ejemplo:

Para `n = 357` devuelve

0	1	2
true	false	false

Para `n = 1609` devuelve

0	1	2	3
false	true	true	true

Solución:

```
/** PRECONDICION: n >= 0 */
public static boolean[] deIntAarrayBoolean(int n) {
    int digitosDeN = 1, copiaDeN = n;
    while (copiaDeN > 9) {
        digitosDeN++;
        copiaDeN = copiaDeN / 10;
    }

    boolean[] res = new boolean[digitosDeN];
    copiaDeN = n;
    while (digitosDeN > 0) {
        digitosDeN--;
        res[digitosDeN] = ((copiaDeN % 10) % 3) == 0;
        copiaDeN = copiaDeN / 10;
    }
    return res;
}
```