PRG - ETSInf. TEORÍA. Curso 2016-17. Parcial 2.
5 de junio de 2017. Duración: 2 horas.

Nota: El examen se evalúa sobre 10 puntos, pero su peso específico en la nota final de PRG es de **3 puntos**.

1. 2.5 puntos Se dispone de un fichero de texto, tal que cada una de sus líneas debe contener un único valor numérico `double` (teniendo como separador decimal, caso de existir, el punto decimal).

   Sin embargo, no todas las líneas tienen un valor escrito correctamente, por lo que se desea rechazar las mismas. Para ello, se tiene que generar un nuevo fichero que contendrá, exclusivamente, las líneas correctas del fichero original. Cada línea contendrá un único valor numérico `double`.

   El nuevo fichero se escribirá en el mismo directorio que el original y su nombre se obtendrá añadiendo al del original el sufijo `"_nuevo"`.

   Por ejemplo: si el nombre del fichero original era `"../../numeros.txt"` entonces el nombre del fichero nuevo deberá ser: `"../../numeros.txt_nuevo"`.

   En el caso de que el fichero original no exista, o no sea accesible, no se escribirá ningún mensaje. Así mismo, si alguna línea contiene algún valor no correcto, deberán tratarse el resto de líneas posteriores que puedan existir, aunque no se debe escribir ningún mensaje relacionado con la línea errónea.

   Se pide: Escribir un método que reciba el nombre del fichero original como una `String` y realice el procesamiento indicado.

   NOTA: Deberá tenerse en cuenta, para la gestión de los ficheros de entrada o salida, la posible excepción: `FileNotFoundException`. Adicionalmente, para la lectura del valor numérico, deberá tenerse en cuenta, bien la excepción `InputMismatchException`, bien la `NumberFormatException`, en el caso en que se haya utilizado el método `Double.parseDouble(String)`.

Solución:

```
// Solución alternativa 1.
public static void fichNue(String ftx) {
    File f = new File(ftx);
    Scanner s = null; PrintWriter p = null;
    try {
        s = new Scanner(f).useLocale(Locale.US);
        p = new PrintWriter(ftx + "_nuevo");
        while (s.hasNextLine()) {
            try {
                String lA = s.nextLine().trim();
                p.println(Double.parseDouble(lA));
            } catch (NumberFormatException n) { }
        }
    } catch (FileNotFoundException e) { }
    finally {
        if (s != null) { s.close(); }
        if (p != null) { p.close(); }
    }
}

// Solución alternativa 2.
public static void fichNue(String ftx) {
    File f = new File(ftx);
    Scanner s = null; PrintWriter p = null;
    try {
        s = new Scanner(f).useLocale(Locale.US);
        p = new PrintWriter(ftx + "_nuevo");
        while (s.hasNextLine()) {
```

```
            try {
                p.println(s.nextDouble());
            } catch (InputMismatchException e) { }
            finally { s.nextLine(); }
        }
    } catch (FileNotFoundException e) { }
    finally {
        if (s != null) { s.close(); }
        if (p != null) { p.close(); }
    }
}
```

2. 2.5 puntos **Se pide:** implementar un nuevo constructor en la clase `PilaIntEnla`, tal que dada una `PilaIntEnla` `p` cree una nueva pila que sea una copia independendiente de `p`. Así, si se ejecuta la instrucción:

   `PilaIntEnla res = new PilaIntEnla(PilaIntEnla p);`

   se podrá apilar y desapilar de la pila `res` o de la pila `p` sin influir una en la otra.

   NOTA: Sólo se permite acceder a los atributos de la clase, quedando prohibido el acceso a sus métodos.

Solución:

```
public PilaIntEnla(PilaIntEnla p) {
    if (p.cima != null) {
        cima = new NodoInt(p.cima.dato);
        NodoInt ultimo = cima;
        NodoInt aux = p.cima.siguiente;
        while (aux != null) {
            ultimo.siguiente = new NodoInt(aux.dato);
            ultimo = ultimo.siguiente;
            aux = aux.siguiente;
        }
        talla = p.talla;
     }
}
```

3. 2.5 puntos Se quiere implementar el método `equals(Object)` en la clase `ListaPIIntEnla`, para definir cuándo la `ListaPIIntEnla` sobre la que se aplique dicho método es o no igual a otra que se recibirá como argumento.

   Dadas dos `ListaPIIntEnla` cualesquiera, se consideran iguales cuando:

   1. Tienen los mismos elementos en las mismas posiciones y
   2. su *Punto de Interés* es el mismo, esto es, se encuentra en la misma posición en ambas listas.

   Se pide: completar el código siguiente para que el método `equals(Object)` se ejecute según la definición anterior:

```
/** Devuelve si el objeto en curso es igual a o */
public boolean equals(Object o) {
    boolean res = true;
    if (!(o instanceof ListaPIIntEnla)) { res = false; }
    else {
        ListaPIIntEnla otra = (ListaPIIntEnla) o;

        // COMPLETA el código para devolver si "this" y "otra" son iguales
}
```

NOTA: La posición del *Punto de Interés* deberá permanecer inalterada tras la ejecución del método.

Solución:

```
/** Devuelve si el objeto en curso es igual a o */
public boolean equals(Object o) {
    boolean res = true;
    if (!(o instanceof ListaPIIntEnla)) { res = false; }
    else {
        ListaPIIntEnla otra = (ListaPIIntEnla) o;
        if (this.talla != otra.talla) { res = false; }
        else {
            NodoInt p = this.primero, q = otra.primero;
            while (p != null && res) {
                res = (p.dato == q.dato);
                if (res && (p == this.antPI)) { res = q == otra.antPI; }
                if (res) { p = p.siguiente; q = q.siguiente; }
            }
        }
    }
    return res;
}
```

4. 2.5 puntos Por problemas de codificación, se ha recibido cierto texto en el que todas las letras `'ñ'` han sido cambiadas por el par de caracteres `'~n'` y todas las `'Ñ'` por el par `'~N'`.

   Se dispone de una clase ya creada. `ListaPICharEnla`. que implementa una lista con punto de interés cuyos elementos son caracteres, y con los siguientes métodos públicos:

```
public void inicio()            public void siguiente()
public void insertar(char e)    public char eliminar()
public char recuperar()         public int talla()
public boolean esVacia()        public boolean esFin()
```

   Se pide: implementar un método estático, externo a la clase `ListaPICharEnla`, que dado un objeto de dicho tipo que contiene, carácter a carácter, cierto texto con el problema de codificación anteriormente expuesto, modifique el contenido de dicha lista para codificarlo de forma correcta.

   Por ejemplo: dada una lista con el texto `"Juan Nú~nez de la Pe~na, ESPA~NA"`, deberá modificarse para que quede `"Juan Núñez de la Peña, ESPAÑA"`.

   NOTA: Considérese que si aparece el carácter `'~'`, éste siempre irá seguido de una `'n'` o una `'N'`.

Solución:

```
public static void modifOneta(ListaPICharEnla l) {
    l.inicio();
    while (!l.esFin()) {
        char c1 = l.recuperar();
        if (c1 == '~') {
            l.eliminar();
            char c2 = l.eliminar();
            if (c2 == 'n') { l.insertar('ñ'); }
            else { l.insertar('Ñ'); }
        } else { l.siguiente(); }
    }
}
```