PRG - ETSInf. TEORÍA. Curso 2018-19. Recuperación Parcial 2.
11 de junio de 2019. 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. 3.0 puntos Se tiene un fichero de texto con información sobre los tenistas profesionales que proporciona la ATP en la que, en cada línea, figura el apellido del tenista, su edad, los puntos conseguidos y el número de torneos en los que ha participado durante el año en curso. El fichero puede tener errores porque:
   - La información no esté completa (falte algún dato) o haya algún dato adicional.
   - La edad, o los puntos, o el número de torneos no sean números enteros, o sean números negativos.

   Se pide: diseñar un método que reciba como parámetros un String `fileIn`, con el nombre del fichero de texto a analizar, y otro String `fileOut`, con el nombre del fichero de texto resultado del proceso. El método debe leer los datos del fichero `fileIn` y para cada línea que contenga un error, copiar un mensaje en el fichero de salida indicando el error detectado. El método debe limitarse a propagar la excepción (comprobada) `FileNotFoundException` si ha habido algún problema en la apertura de los ficheros.

   Por ejemplo, si el fichero de entrada fuera:

```
Djokovic    31  12115   17
Nadal       -32 7945    16
Federer     37  5770    17
Thiem       25  4845    24  4
Zverev      22  4745
Nishikori   29  3860    23.5
Tsitsipas   20  3790    28
Anderson    32  3755.3  17
```

   El fichero de salida debería ser:

```
Error línea 2: Valor negativo.
Error línea 4: Incorrecto el número de datos.
Error línea 5: Incorrecto el número de datos.
Error línea 6: Formato de entero no válido.
Error línea 8: Formato de entero no válido.
```

   Si se supone que los datos en cada línea están separados por tabuladores, se puede utilizar la siguiente instrucción para separar los diferentes datos contenidos en una línea:
   `String[] words = linea.split("([ \t])+");`
   Por ejemplo, si la variable `linea` (de tipo `String`) contiene `"Djokovic 31 12115 17"`, el array de `String` resultante será `["Djokovic", "31", "12115", "17"]`.
   Se puede usar el método `parseInt` de la clase `Integer` para transformar un String en el entero que represente. Nótese que este método lanza la excepción `NumberFormatException` si el formato no es apropiado.

Solución:

```
public static void filtraErrores(String fileIn, String fileOut) throws FileNotFoundException  {
    File fE = new File(fileIn); File fS = new File(fileOut);
    Scanner ent = new Scanner(fE); PrintWriter sal = new PrintWriter(fS);
    int cont = 0;
    while (ent.hasNext()) {
        try {
            String linea = ent.nextLine();  cont++;
            String[] words = linea.split("([ \t])+");
            if (words.length != 4) {
                sal.println("Error línea " + cont + ": " + "Incorrecto el número de datos.");
            } else {
```

```
                int edad = Integer.parseInt(words[1]);
                int puntos = Integer.parseInt(words[2]);
                int noTorneos = Integer.parseInt(words[3]);
                if (edad < 0 || puntos < 0 || noTorneos < 0)        {
                    sal.println("Error línea " + cont + ": " + "Valor negativo.");
                }
            }
        } catch (NumberFormatException e) {
            sal.println("Error línea " + cont + ": " + "Formato de entero no válido.");
        }
    }
    in.close(); sal.close();
}
```

2. 3.5 puntos **Se pide:** implementar un método de instancia, en la clase `ListPIIntLinked`, con el perfil y precondición siguientes:

```
/** precondición: la lista this contiene datos almacenados en orden creciente */
public void removeGreaters(int e)
```

Dado un entero `e`, el método modifica la lista borrando de la misma todos los elementos mayores que `e`. Al final de la ejecución, el `PI` se sitúa al inicio de la lista.

Por ejemplo, si la lista `l` inicialmente es `[10] 12 14 15` , y el entero `e` es `12`, entonces la lista después de ejecutar el método será `[10] 12` . Considerando la misma lista `l`, y el entero `e` = `9`, entonces la lista después de ejecutar el método queda vacía.

REQUISITO: Utilizar solamente los atributos de la clase y referencias auxiliares a `NodeInt`, y no los métodos de la clase.

Solución:

```
public void removeGreaters(int e) {
    if (first != null && first.data > e) {
        first = null;
        size = 0;
    }
    else {
        int cont = 0;
        NodeInt aux = first;
        NodeInt prev = null;
        while (aux != null && aux.data <= e) {
            prev = aux;
            aux = aux.next;
            cont++;
        }
        if (aux != null) {
            prev.next = null;
            size = cont;
        }
    }
    prevPI = null;
    pI = first;
}
```

3. 3.5 puntos **Se pide:** implementar un método estático de perfil

```
public static void recular(QueueIntLinked q, int x)
```

tal que:

- Busque la primera ocurrencia del elemento `x` dentro de la cola `q` y, en caso de éxito en la búsqueda, haga que dicho elemento se traslade al final del todo y, por tanto, se quede como el último de la cola.
- En caso de fracaso en la búsqueda, la cola se queda como estaba.

REQUISITO: Se supondrá que el método se implementa en una clase diferente a `QueueIntLinked`, por tanto, solo se podrán usar los métodos públicos de la clase.

Solución:

```
public static void recular(QueueIntLinked q, int x) {
     int n = q.size();
     int i = 0;
     while (i < n && q.element() != x) {
          q.add(q.remove());
          i++;
     }
     if (i < n) {
          q.remove();
          for (int j = i + 1; j < n; j++) {
               q.add(q.remove());
          }
          q.add(x);
     }
}
```

ANEXO

Atributos de la clase `ListPIIntLinked` y métodos de las clases `QueueIntLinked`.

```
public class ListPIIntLinked {
    private NodeInt first;
    private NodeInt pI;
    private NodeInt prevPI;
    private int size;
    ...
}

public class QueueIntLinked {
    ...
    public QueueIntLinked() { ... }
    public void add(int x) { ... }
    public int remove() { ... }
    public int element() { ... }
    public boolean empty() { ... }
    public int size() { ... }
}
```