PRG - ETSInf. TEORÍA. Curso 2016-17. Recuperación Parcial 2.
26 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 sus líneas contienen, separados por espacios en blanco, tokens que son representaciones válidas de números enteros y tokens que no lo son. **Se pide:** implementar un método estático que, dado un `String` con la ruta y el nombre del fichero, devuelva la suma de todos los enteros que contenga el fichero. El método tiene que:

   - Capturar la excepción `FileNotFoundException` que puede ocurrir al intentar abrir el fichero, mostrando un mensaje por pantalla.
   - Leer (mientras queden) las líneas del mismo, usando el método `nextLine()` de la clase `Scanner`.
   - Dividir cada línea en los tokens correspondientes, usando el método `split(expReg)` de la clase `String`. Mediante dicho método, se obtiene un array de `String`, tal que cada componente del mismo es uno de los tokens que aparecen en la `String` sobre la que se aplica, utilizando como separadores válidos los descritos en `expReg`. En este ejercicio se usará como separador el espacio en blanco, esto es, `split(" ")`.
   - Convertir cada token o `String` del array anterior al valor entero correspondiente, usando el método `Integer.parseInt(String)`.
   - Si el token es una representación válida de un número entero, dicho número se sumará al resultado a devolver.
   - Si el token no es una representación válida de un número entero, entonces se debe capturar la excepción `NumberFormatException` que se genera en tal caso, mostrando por pantalla el valor del token que la ha generado. Esta circunstancia no debe impedir que continúe la conversión de tokens a enteros ni la lectura del fichero.

Solución:

```
public static int leeYSuma(String nomFichero) {
    int suma = 0;
    Scanner s = null;
    try {
        s = new Scanner(new File(nomFichero));
        while (s.hasNextLine()) {
            String[] linea = s.nextLine().trim().split(" ");
            for (int i = 0; i < linea.length; i++) {
                try {
                    suma += Integer.parseInt(linea[i]);
                } catch (NumberFormatException e) {
                    System.out.println(linea[i]);
                }
            }
        }
    } catch (FileNotFoundException e) {
        System.out.println("No se encontró el fichero");
    } finally {
        if (s != null) { s.close(); }
    }
    return suma;
}
```

2. 2.5 puntos Se dispone de la siguiente clase que implementa una secuencia enlazada con nodos de tipo `NodoInt`:

```
public class SecEnla {
    private NodoInt sec;  // referencia al primer elemento de la secuencia
    . . .
}
```

Se pide: implementar un método de instancia en la clase `SecEnla` que, dado un valor entero `x`, encuentre la primera ocurrencia de dicho valor en la secuencia y lo adelante una posición. Si `x` se encuentra en el primer nodo de la secuencia, es decir, no se puede adelantar una posición, pasará al último y viceversa. En caso de que `x` no se encuentre, no hará nada. Se supone, por precondición, que la secuencia tiene, al menos, 2 nodos. Por ejemplo, si la secuencia contiene `[3, 7, 2, 8, 5]` y ejecutamos la llamada al método `adelantar(8)`, la secuencia quedará `[3, 7, 8, 2, 5]`. Si, a continuación, ejecutamos `adelantar(3)`, la secuencia quedará `[5, 7, 8, 2, 3]`.

Solución:

```
/** Precondición: la secuencia tiene, al menos, 2 nodos */
public void adelantar(int x) {
    NodoInt ant = null, aux = sec;
    while (aux != null && aux.dato != x) {
        ant = aux;
        aux = aux.siguiente;
    }
    if (aux != null) {
        if (ant == null) {
            ant = aux;
            while (ant.siguiente != null) {
                ant = ant.siguiente;
            }
        }
        aux.dato = ant.dato;
        ant.dato = x;
    }
}
```

3. 2.5 puntos **Se pide:** implementar un método de instancia en la clase `ListaPIIntEnla` con perfil

```
public void insertarEn(int x, boolean inicio)
```

que, dado un entero `x`, lo inserte al inicio de la lista si el parámetro `inicio` es `true` y, en caso contrario, lo inserte al final. El punto de interés deberá quedar sobre el elemento insertado.

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

Solución:

```
public void insertarEn(int x, boolean inicio) {
    NodoInt nuevo = new NodoInt(x);
    if (primero == null) { primero = nuevo; }
    else if (inicio) {
        nuevo.siguiente = primero;
        primero = nuevo;
        antPI = null;
    }
    else {
        NodoInt aux = primero;
        while (aux.siguiente != null) { aux = aux.siguiente; }
        aux.siguiente = nuevo;
        antPI = aux;
    }
    pI = nuevo;
    talla++;
}
```

4. 2.5 puntos **Se pide:** implementar un método estático fuera de la clase `ColaIntEnla` tal que, dada una `ColaIntEnla` con valores en [0..9], devuelva el número entero formado por los dígitos de la misma. La cola debe quedar en su estado original. Por ejemplo, si la cola es ← `5 1 4 7` ←, el entero resultante será `5147`. Fíjate que dicho entero puede obtenerse como sigue: `(((((5 * 10) + 1) * 10) + 4) * 10) + 7`.

Solución:

```
/** Versión 1: sin estructuras auxiliares */
public static int fromColaToInt(ColaIntEnla q) {
    int res = 0;
    for (int i = 0; i < q.talla(); i++) {
        int x = q.desencolar();
        res = res * 10 + x;
        q.encolar(x);
    }
    return res;
}

/** Versión 2: usando una cola auxiliar */
public static int fromColaToInt(ColaIntEnla q) {
    int res = 0;
    ColaIntEnla qAux = new ColaIntEnla();
    while (!q.esVacia()) {
        int x = q.desencolar();
        res = res * 10 + x;
        qAux.encolar(x);
    }
    while (!qAux.esVacia()) {
        int x = qAux.desencolar();
        q.encolar(x);
    }
    return res;
}
```