Recuperación Parcial 2 - Teoría - PRG - ETSInf - Curso 2012-13.
17 de junio de 2013. Duración: 1 hora y 50 minutos.

1. 2.5 puntos Dados el nombre de un fichero de texto `nomFich` y una palabra `pal`, ambos de tipo `String`, **se pide** implementar un método estático que copie todas las líneas que contienen dicha palabra en un fichero de texto llamado *result.txt* en el sistema, precedidas por el número de línea que ocupan en el fichero `nomFich`. El método **debe capturar** la excepción `FileNotFoundException`, escribiendo un mensaje en la salida estándar en el caso de que dicha excepción ocurra.

   NOTA: En la resolución de este ejercicio se puede usar el método `contains` de la clase `String` tal que, siendo `s1` y `s2` de tipo `String`, `s1.contains(s2)` devuelve `true` si `s1` contiene a `s2` como subcadena y, en caso contrario, devuelve `false`.

   Ejemplo: con el siguiente fichero de entrada y la palabra `ratón`:

   ```
   El ratón de Federico
   no tiene patas ni hocico.
   Es ratón de ordenador,
   de un ordenador muy listo.
   ```

   El fichero de salida (*result.txt*) sería:

   ```
   1  El ratón de Federico
   3  Es ratón de ordenador,
   ```

Solución:

```
import java.io.*;
import java.util.*;
public class Ejercicio1 {
    public static void copiar(String nomFich, String pal) {
        try {
            Scanner s = new Scanner(new File(nomFich));
            PrintWriter pw = new PrintWriter(new File("result.txt"));
            int cont = 0;
            while(s.hasNext()) {
                String linea = s.nextLine();
                cont++;
                if (linea.contains(pal)) pw.println(cont + "  " + linea);
            }
            s.close();
            pw.close();
        } catch(FileNotFoundException e) {
            System.out.println("Fichero no encontrado.");
        }
    }
}
```

2. 2.5 puntos Sea la clase `ListaPIIntEnlaOrd` una clase muy parecida a la clase `ListaPIIntEnla` presentada en el tema 5. En las listas de la clase `ListaPIIntEnlaOrd` los valores enteros están en orden estrictamente creciente. El constructor y todos los métodos de esta clase (excepto `insertar`) tienen la misma funcionalidad que en la clase `ListaPIIntEnla`.

Se pide: implementar, **sin hacer uso de los métodos de la clase (usando sólo referencias)**, el método:

```
public void insertar(int x)
```

para que inserte `x` ordenadamente en la lista. El punto de interés se situará a la derecha del elemento que se inserta.

Solución:

```
public void insertar(int x) {
    this.PI = this.primero; this.antPI = null;
    while(this.PI!=null && this.PI.dato<x) {
        this.antPI = this.PI;
        this.PI = this.PI.siguiente;
    }
    if (this.PI==null || this.PI.dato>x) {
        NodoInt nuevo = new NodoInt(x,this.PI);
        if (this.PI==this.primero) this.primero = nuevo;
        else this.antPI.siguiente = nuevo;
        this.antPI = nuevo;
        this.talla++;
    }
}
```

3. 2.5 puntos Dada una lista con punto de interés `ListaPIIntEnla l` y un entero `x`, **se pide** implementar un método estático `borrar` que devuelva una nueva `ListaPIIntEnla` con los elementos de la lista `l` iguales a `x`. Además, deberá eliminar dichos valores de la lista `l`.

Por ejemplo, si la lista `l` contiene los valores 2, 3, 4, 3, 7, 5 y 3, tras la ejecución de la llamada `ListaPIIntEnla lis = borrar(l,3)`, la lista `lis` contiene los valores 3, 3 y 3 y la lista `l` contiene los valores 2, 4, 7 y 5.

Se usarán únicamente los métodos públicos de la clase `ListaPIIntEnla`.

Solución:

```
public static ListaPIIntEnla borrar(ListaPIIntEnla l, int x) {
    ListaPIIntEnla lista = new ListaPIIntEnla();
    l.inicio();
    while(!l.esFin()) {
        if (l.recuperar()==x) {
            lista.insertar(x) ;
            l.eliminar();
        }
        else l.siguiente();
    }
    return lista;
}
```

4. 2.5 puntos Dada una `PilaIntEnla p` y un entero `x`, **se pide** implementar un método estático **recursivo** con el siguiente perfil:

```
public static void eliminarMenoresQue(PilaIntEnla p, int x)
```

que elimine de `p` los elementos menores que `x`. Por ejemplo, si la pila `p` contiene los valores: 3, 6, 7, 2, 5 y 4, tras la ejecución de la llamada `eliminarMenoresQue(p,5)`, la pila contiene los valores 6, 7 y 5, tal y como se muestra a continuación:

```
p antes de la ejecución                        p después de la ejecución
        3
        6
        7
        2         ==== eliminarMenoresQue(p,5) ====>       6
        5                                                  7
        4                                                  5
       ---                                                ---
```

Se usarán únicamente los métodos públicos de la clase `PilaIntEnla`.

Solución:

```
private static void eliminarMenoresQue(PilaIntEnla p, int x) {
    if (!p.esVacia()) {
      int aux = p.desapilar();
      eliminarMenoresQue(p,x);
      if (aux>=x) p.apilar(aux);
   }
}
```