Tercer parcial de PRG - ETSInf
Fecha: 8 de junio de 2011. Duración: 2 horas.

1. 3 puntos Teniendo en cuenta la definición de `ListaIntEnla` vista en clase,
   Se pide: escribir un método:

   ```
   public void insertarSinRepetidos(int valor, boolean delante)
   ```

   para insertar un elemento `valor` en la `ListaIntEnla` en el caso de que no existiera previamente en ella, bien al inicio de la lista, como primer elemento, si el argumento `delante` es `true`, bien por la parte de atrás, al final de la lista, en el caso de que el argumento `delante` sea `false`. La `ListaIntEnla` permanecerá sin cambios, en el caso de que el elemento `valor` ya existiera inicialmente en ella. **NOTA:** esta operación tendrá que tener un **coste lineal**.

   Solución:

   ```
   /** Inserta un nuevo elemento solo si no existe (sin duplicados).
    * Bien al inicio (delante es true), bien al final (delante es false)
    */
   public void insertarSinRepetidos(int valor, boolean delante) {
       NodoInt p = primero;
       NodoInt q = null;
       while (p!=null && p.dato != valor) {
           q = p; p = p.siguiente;
       }
       if (p==null) {
           talla++;
           if (q==null) primero = new NodoInt(valor);
           else if (delante) primero = new NodoInt(valor,primero);
           else q.siguiente = new NodoInt(valor);
       }
   }
   ```

2. 2 puntos Suponiendo ya hechas las clases:

   - `NodoStr`, con atributos `dato` de tipo `String` y `siguiente` de tipo `NodoStr`, y con las dos operaciones constructoras habituales definidas en este tipo de clases.
   - `ListaStrEnla`, mediante la que se mantiene una lista enlazada de elementos de tipo `NodoStr` y con todas las operaciones definidas en clase para las listas.

   Se pide: Escribir una operación, dentro de la clase `ListaStrEnla`, que determine si una lista de ese tipo está o no ordenada ascendentemente. Una lista de palabras tal como una `ListaStrEnla`, está ordenada ascendentemente si para cualquier pareja de elementos consecutivos de la lista, el primer elemento es anterior o igual al segundo. Además, por definición, una lista vacía o con sólo un elemento, está ordenada. **NOTA:** esta operación tendrá que tener un **coste lineal**.

Solución:

```
    /** True sii la lista esta ordenada ascendentemente */
    public boolean estaOrdenada() {
        if (talla==0 || talla==1) return true;
        else { NodoStr p = primero.siguiente;
               NodoStr q = primero;
               while (p!=null && q.dato.compareTo(p.dato)<=0) {
                   q = p; p = p.siguiente;
               }
               return p==null;
        }
    }
```

3. [3 puntos] Suponiendo ya definida la clase `NodoInt`, vista en clase, escribir una clase `ExamenIntEnla` con las siguientes operaciones:

   - `ExamenIntEnla()`, constructor de la estructura vacía.
   - `insertarAlInicio(int)`, inserta al inicio de la estructura el valor que se recibe como argumento.
   - `insertarAlFinal(int)`, inserta al final de la estructura el valor que se recibe como argumento.

   Se pide: Implementar la clase `ExamenIntEnla`, declarando para ella el(los) atributo(s) necesario(s), e implementando todas las operaciones pedidas. **NOTA:** todas las operaciones tendrán que tener un **coste constante**.

Solución:

```
package lineales;

/**
 * @author (PRG - ETSINF - DSIC)
 * @version (examen junio 2011)
 */
public class ExamenIntEnla {
    private NodoInt ini;
    private NodoInt ult;

    public ExamenIntEnla() { ini = ult = null; }

    public void insertarAlInicio(int x) {
        if (ini==null) ult = ini = new NodoInt(x);
        else ini = new NodoInt(x,ini);
    }

    public void insertarAlFinal(int x) {
        if (ini==null) ult = ini = new NodoInt(x);
        else ult = ult.siguiente = new NodoInt(x);
    }
}
```

4. [2 puntos] Partiendo de la clase `ListaIntEnla`, y sirviéndose de ella exclusivamente, se desea hacer una nueva clase `PilaIntEnla2` que empleando las operaciones de `ListaIntEnla` implemente todas las operaciones de una Pila, es decir, las siguientes:

   - `PilaIntEnla2()`, constructor.
   - `talla()`, devuelve la talla.
   - `esVacia()`, dice si está vacía.
   - `tope()`, devuelve, si existe, el tope.
   - `apilar(int)`, apila el elemento que se le da.
   - `desapilar()`, desapila y devuelve, si existe, el elemento del tope.

   Se pide: Implementar, sirviéndose de `ListaIntEnla`, la clase `PilaIntEnla2`, declarando para ella **un único atributo** de tipo `ListaIntEnla`, e implementando todas las operaciones pedidas. **NOTA:** Todas las operaciones tendrán que tener un **coste constante**.

Solución:

```
package lineales;

/**
 * PilaIntEnla2 implementa una PilaInt mediante una ListaIntEnla
 * @author (PRG - ETSINF - DSIC)
 * @version (examen junio 2011)
 */
public class PilaIntEnla2 {
    private ListaIntEnla laLista;

    public PilaIntEnla2() {
        laLista = new ListaIntEnla();
    }

    public int talla() { return laLista.talla(); }
    public boolean esVacia() { return laLista.esVacia(); }
    public int tope() { return laLista.recuperar(0); }
    public void apilar(int x) { laLista.insertar(0,x); }
    public int desapilar() { return laLista.eliminar(0); }
}
```