CURSO 2013-14
TRABAJO PRÁCTICO
Construcción de los TADs Persona y Agenda con tecnología de listas con cabecera y centinela

1. OBJETIVO

El objetivo de la práctica es desarrollar los tipos abstractos de datos (TADs) **Persona** y **Agenda** utilizando para el segundo la tecnología de listas con cabecera y centinela[1].

2. ESPECIFICACIONES DE LOS TADs.

Los TAD's **Persona** y **Agenda** deberán cumplir las especificaciones indicadas en el hito 1 del proyecto.

3. CRITERIOS DE DISEÑO.

3.1. TAD Persona

Contiene las siguientes variables miembro:

```
private String nombre;
private String apellidos;
private String direccion;
private String poblacion;
private String provincia;
private String codigoPostal;
private String telefono;
private int anioNacim;
```

Y un único constructor:

```
public Persona () {
    nombre = "";
    apellidos = "";
    direccion = "";
    poblacion = "";
```


[1] El alumno no tiene que diseñar el TAD **Parser**. Se utilizará el facilitado en la tarea.

3.2. TAD AGENDA:

Este TAD utiliza los *TADs Persona y Parser* e implementa la interfaz *Agenda:*

```
import java.io.IOException;

public interface Agenda {
    boolean aniadirPersona (Persona p);
    boolean eliminarPersona (String nombre);
    Persona quitarPrimero ();
    boolean estaVacia ();
    int numeroPersonas ();
    boolean guardarAgenda () throws IOException;
    boolean recuperarAgenda () throws IOException;
}
```

Se trata de una lista enlazada calificada ordenada con cabecera y centinela que utiliza como clave los campos Apellido y Nombre de la persona (siendo la clave apellidos + " " + nombre). Los nodos de la lista pertenecen a la clase **NodoAgenda**, definida a continuación:

```
class NodoAgenda {
    Persona info;
    NodoAgenda sig;
    NodoAgenda (Persona p, NodoAgenda siguiente) {
        info = p;
        sig = siguiente;
    }
}
```

Mientras que su estructura interna es:

```
public class Agenda2 implements Agenda {
    private NodoAgenda cab, cent;
    private int numPersonas;
    public Agenda2 () {
        cent = new NodoAgenda (null, null);
        cab = new NodoAgenda (null, cent);
        numPersonas = 0;
    }
}
```

ANEXO. CONCEPTO DE CLASE INTERFAZ

Una clase de interfaz es un mecanismo que permite implementar una clase de formas diferentes. En el caso del proyecto de la práctica se tendría (clase *Agenda.java*):

```
/*<modificador de visibilidad>*/ interface Agenda {
   boolean aniadirPersona(Persona p);
   boolean eliminarPersona(String nombre);
   Persona quitarPrimero();
   boolean estaVacia();
   int numeroPersonas();
   boolean guardarAgenda() throws IOException;
   boolean recuperarAgenda() throws IOException;
}
```

Observe que se utiliza la palabra reservada ***interface*** en lugar de ***class***.
Una clase de interfaz (***interface***) contiene solo las cabeceras de los métodos accesibles por otras clases tanto dentro como fuera del paquete donde se encuentre la clase de interfaz.

Una vez definida la interfaz, deberá implementar los métodos que se enumeran en ella. A continuación se muestra un esqueleto (clase *Agenda 2*).

```
class Agenda2 implements Agenda {
   /* objetos, variables y constantes*/

   /*constructor*/ Agenda2 (){…}

   public boolean aniadirPersona(Persona p){…}
   public boolean eliminarPersona(String nombre) {…}
   public Persona quitarPrimero() {…}
   public boolean estaVacia() {…}
   public int numeroPersonas() {…}
   public boolean guardarAgenda()throws IOException {…}
   public boolean recuperarAgenda()throws IOException {…}
}
```

Observe la palabra reservada ***implements*** en la declaración de la clase *Agenda2*. Se utiliza para indicar que dicha clase es una posible manera de implementar la interfaz *Agenda, pero podría no ser la única forma de hacerlo.*

Cuando en un programa principal (método *main*) se desea utilizar objetos (previamente creados) implementados de una forma determinada se utiliza la siguiente sintaxis: