2 | Estructuras de control y Listas

Una vez que se conocen las construcciones de un lenguaje de programación para controlar el flujo de ejecución del código, es posible manejar cantidades de información cuyo tamaño se desconoce al momento de codificar el programa. En esta práctica se hará uso de listas para contar con un almacén de datos de tamaño variable.

Listas

Las listas son un tipo abstracto de datos que nos permite tener un número arbitrario de elementos de un tipo dado. Formalmente, la podemos definir como:

Definición 2.1 *Una* lista *es:*

- *Una lista vacía.*
- *Un elemento seguido de otra lista.*

La definición de una lista es *recursiva*: se define en términos de sí misma. En este caso decimos que se trata de una *recursión estructural*, pues las listas son objetos (estructuras) que contienen objetos del mismo tipo que ellos.

Toda recursión contiene en su definición un *caso base*, es decir, un caso muy sencillo a partir del cual se construyen los más complejos. En este caso el caso base es cuando la lista es una lista vacía, pues ésta ya no contiene a otra lista. Puedes visualizarla así:

$$l = \{\}$$

A partir de ahí se construyen las listas más largas, si los elementos son letras, otros ejemplos de listas son:

$$l_1 = \{A\{\}\}$$
$$l_2 = \{L\{I\{S\{T\{A\{\}\}\}\}\}\}$$

Observa que cada lista contiene a un elemento y otra lista dentro, al final siempre queda un lista vacía.

En Java usaremos a null para representar a lista vacía. Si nuestra lista contiene objetos de tipo cadena, podemos definir a la clase con los atributos siguientes:

```
1 public class Lista {
2   private String elemento;
3   private Lista siguiente;
4 }
```

Los métodos que operan sobre las listas tendrán que trabajar de forma distinta dependiendo de si ésta es vacía o no. En orientación a objetos nos gusta que este tipo de detalles queden ocultos al programador. Después de todo, una lista es una lista, querríamos tener un objeto que agregue elementos a lista y no nos distraiga con este tipo de detalles. Por ello utilizaremos un ManejadorDeLista. Éste se encargará de resolver todos los detalles referentes a la administración de la lista y nos dejará preocuparnos únicamente por agregar, buscar y, en general, operar con nuestros datos, independientemente de cómo los esté guardando.

Ejercicios

Para esta práctica se te entrega un archivo comprimido con dos paquetes, con varios archivos:

- icc.practica5
  - ⋆ UsoBaseDeDatosAgenda.java
  - ⋆ BaseDeDatosAgenda.java
  - ⋆ RegistroAgenda.java
  - ⋆ BuscadorPorNombre.java
- icc.util
  - ⋆ ManejadorDeLista.java
  - ⋆ ListaAgenda.java
  - ⋆ Buscador.java - Este archivo es en realidad una interfaz.

1. Abre el archivo ListaAgenda.java, el trabajo de este objeto es almacenar los datos de la lista. Hay otra clase de objetos que tiene una responsabilidad semejante: RegistroAgenda. Completa el código faltante en esta clase. Compila tu código y asegúrate de que no tenga errores.

2. Después de compilar tu código podrás solicitar a ant que genere la documentación de las clases que se te dieron utilizando el comando ant compile. Lee

la documentación de la clase `ManejadorDeLista`.

3. Abre `BaseDeDatosAgenda.java`. Completa el código que falta dentro del constructor. Debes extraer los datos de cada registro de la cadena `datos`, que recibe como parámetro. Para cada registro crea un objeto `RegistroAgenda` y guárdalo dentro de la lista utilizando al `ManejadorDeLista`.

4. Prueba tu código creando un par de `BaseDeDatosAgenda` en el método main de `UsoBaseDeDatosAgenda`.

5. Ahora comenzaremos con las búsquedas. Lee la documentación de la clase `BuscadorPorNombre` y `Buscador`. Asegúrate de que entiendes lo que hace cada método. ¿Te queda clara la relación entre la interfaz `Buscador` y la clase `BuscadorPorNombre`? Si no, pregunta a tu ayudante.

   Supon que `ManejadorDeLista` está completa, revisa la documentación de su método `encuentra(Buscador buscador)`. Trata de escribir el código del método `dameRegistroPorNombre (String nombre)` de `BaseDeDatosAgenda`. Mándalo llamar en el método main de `UsoBaseDeDatosAgenda` haciendo una consulta. Compila y trata de ejecutar tu código. No va a funcionar, porque `encuentra(Buscador buscador)` ahorita no hace nada. Pero observa que ya tienes lista la prueba para tu código, aunque no esté completo. Esto es importante porque así podrás verificar que esté bien cuando lo termines.

6. Abre el archivo `ManejadorDeLista.java`, observa que el primer elemento de la lista (si no es vacía) debe quedar en el atributo `cabeza`. Deberás completar el código del método `encuentra(Buscador buscador)`, que devuelve al registro que cumple la condición que revisa `Buscador`. Puedes leer cómo fue programado `agrega(RegistroAgenda elemento)` para darte algunas ideas. Cuando termines compila y ejecuta tu código. La búsqueda por nombre ya debe funcionar.

   Asegúrate de probar que puedes encontrar a la última persona en la agenda y que tu programa no hace nada raro si buscas a alguien que no está.

7. Ahora sólo agrega la búsqueda por teléfono. También debes probarla.