EIF207 Estructuras de Datos
Práctica #4 (#5)

Prof. M.Sc. Georges E. Alfaro S.

Implemente la solución de los siguientes ejercicios en **Java**, empleando el IDE **NetBeans**.

PARTE ÚNICA.

Ejercicio #1a (5 pts.)

Construya una función:

```
public void sort(DQueue<T> q1)
```

que ordene ascendentemente los elementos una lista doblemente enlazada (*double queue*) utilizando el algoritmo ***merge sort***. Escriba un método que permita dividir la lista en dos partes iguales en un tiempo $O(n/2)$, que no requiera examinar la lista completa.

Ejercicio #1b (10 pts.)

Escriba una versión de la función del ejercicio anterior que reciba un comparador como parámetro que permita cambiar la secuencia de ordenamiento. Pruebe el método con un comparador que permita ordenar la lista de manera descendente.

```
public void sort(DQueue<T> q1, Comparator<T> c)
```

Nota:

Puede definir ambas funciones como métodos de instancia de la clase `DQueue`.

Ejercicio #2 (50 pts.)

Modifique el proyecto **'ejercicio14'** que se encuentra en la carpeta de la semana 8.

- implemente la clase `Empleado` (constructor, métodos de acceso y mutadores)
- implemente la clase `ListaEmpleados` (que contiene una colección de empleados y permite los métodos básicos de actualización). Estos métodos pueden implementarse por delegación en una estructura existente dentro de la biblioteca vista en clase (`Array`, `List`, `DQueue`).
- implemente un método que recorra el árbol del documento XML para cargar una lista de empleados.
- construya un método `toXMLString()` que convierta la lista en un documento válido XML (sin utilizar ninguna biblioteca para la conversión).
- implemente el acceso al conjunto de empleados mediante la clase `EmpleadoDAO`. Esta clase implementa las cuatro (4) operaciones básicas del almacenamiento persistente (CRUD). Esta clase se puede utilizar para manejar el conjunto de empleados por medio de una interfaz consistente que puede asociarse luego a un elemento de almacenamiento externo (como un gestor de base de datos, por ejemplo)

Ejercicio #3 (35 pts.)

Modifique el **ejemplo 16b** para incluir campos en la interfaz que permitan agregar y eliminar elementos de un **árbol binario de búsqueda**. El programa ya no deberá incluir datos desde el inicio, sino que recibirá solamente los que el usuario digite por medio de la interfaz. Implemente el programa usando un árbol que contenga solamente valores enteros (`Integer`).