# *TRABAJO PRACTICO N° 6*

# *LISTAS E ITERADORES*

***Nota1: para cada clase hacer un programa que pruebe cada uno de los métodos que provee.***

***Nota2: documentar cada clase y utilizar Javadoc para generar la misma.***

1. Utilizar una lista para cargar una nómina de empleados (ejercicio 2.10). Mostrar un ejemplo de uso de cada uno de los métodos de la interface ***List.***

2. Implementar el TAD **Stack** usando un **ArrayList** para almacenar sus elementos.

3. Implementar el TAD **Deque** usando un **ArrayList** para almacenar sus elementos.

4. La clase **java.util.ArrayList** incluye el método *trimToSize()* que reemplaza el array utilizado por uno con la capacidad igual a la cantidad de elementos de la lista. Implementar este método para la versión dinámica de **ArrayList.**

5. Modificar el método *remove(i)* para liberar la memoria del array si la cantidad de espacio libre es mayor o igual a *CAPACITY*.

6. Incluir a la interface **List** e implementar en la clase **ArrayList** los siguientes métodos:

```
/* Remueve todos los elementos de la lista */
void clear()

/* Retorna true si la lista contiene el elemento especificado */
boolean contains(E e);

/* Retorna el índice de la primer ocurrencia del elemento especificado
en la lista, o -1 si la lista no contiene el elemento */
int indexOf(E e);

/* Retorna el índice de la última ocurrencia del elemento especificado
en la lista, o -1 si la lista no contiene el elemento */
int lastIndexOf(E e);

/* Remueve la primer ocurrencia del elemento especificado desde la
lista, si está presente. */
boolean remove(E e);
```

7. Crear la clase **LinkedList** que implementa el TAD **List** (ejercicio 6.6) usando una **DoubleLinkedList** para almacenar elementos.

8. Utilizar una lista posicional para cargar empleados y facturas (ejercicio 2.11). Calcular los importes a pagar. Mostrar un ejemplo de uso de cada uno de los métodos de la interface ***PositionalList.***

# *TRABAJO PRACTICO N° 6*

# *LISTAS E ITERADORES*

9. Describir una implementación de los métodos *addLast* y *addBefore* de una lista posicional utilizando solamente métodos del siguiente conjunto *{isEmpty, first, last, before, after, addAfter, addFirst}.*

10. Extender el TAD **PositionalList** agregando el método *indexOf(p)* que retorna el índice corriente del elemento almacenado en la posición p. Mostrar cómo implementar este método usando solamente otros métodos de la interface ***PositionalList.***

11. Extender el TAD **PositionalList** agregando el método *findPosition(e)* que retorna la primera posición que contiene un elemento igual a e (o *null* si no se encuentra). Mostrar cómo implementar este método usando solamente otros métodos de la interface ***PositionalList.***

12. Hacer una aplicación que cargua en una lista figuras geométricas (ejercicio 2.9). Utilizar un iterador para recorrer la misma y eliminar aquellas que su área sea menor a un valor determinado. Modificar la aplicación para que utilice una lista posicional.

13. Implementar la interface **Iterator** a la clase **LinkedList** (ejercicio 6.7). Probar el mismo con una lista de figuras geométricas (ejercicio 2.9).