TRABAJO PRACTICO N° 3

ESTRUCTURAS DE DATOS FUNDAMENTALES

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. Implementar la clase Lista utilizando un arreglo. Al inicializarse la misma se debe indicar la cantidad de objetos que puede almacenar. La clase lista debe proveer los siguientes métodos:

```
/* Agrega un elemento al final de la lista */
public void add(E e) throws IndexOutOfBoundsException

/* Agrega un elemento a la lista en la posición p */
public void add(int p, E e) throws IndexOutOfBoundsException

/* Retorna el elemento que se encuentra en p */
public E get(int p) throws IndexOutOfBoundsException

/* Remueve el elemento E de la lista. Retorna null sino se encuentra*/
public E remove(E e)

/* Remueve el elemento que se encuentra en la posición p */
public E remove(int p) throws IndexOutOfBoundsException
```

2. Cargar un arreglo con elementos y probar los siguientes métodos de la clase `java.util.Arrays`

- `equals(A, B)`
- `fill(A,x)`
- `copyOf(A, n)`
- `copyOfRange(A, s, t)`
- `toString(A)`
- `sort(A)`
- `binarySearch(A, x)`

3. Mejorar la implementación de la clase **CaesarCipher** vista en clase para encriptar letras minúsculas. Extender la implementación para encriptar números. (Opcional) Utilizar el alfabeto español que incluye la ñ y contiene 27 letras en total para realizar el encriptado.

4. Desarrollar un juego similar al Tatetí, cuando comienza el juego se solicita el tamaño del tablero y luego se pide a cada usuario que ingrese alternativamente su jugada, si la posición no es válida se la solicita nuevamente. El juego termina cuando todas las fichas de un jugador están en línea (el jugador gana) o cuando no hay más posiciones libres en el tablero.

5. Agregar a la clase **SinglyLinkedList** los siguientes métodos:

TRABAJO PRACTICO N° 3

ESTRUCTURAS DE DATOS FUNDAMENTALES

```
/* Inserta el elemento e en la posicion n de la lista */
public void addPos(E e, int n) throws IndexOutOfBoundsException

/* Elimina el elemento e de la lista
/* Retorna NULL si no lo encuentra */
public E removeElement(E e)

/* Elimina elemento que se encuentra en la posicion n de la lista */
/* Retorna NULL si no es una posicion valida */
public E removePos(int n) throws IndexOutOfBoundsException

/* Inserta todos los elementos de la Lista l al final de la lista */
public void concatenate(SinglyLinkedList l)

/* Busca el elemento e dentro de la lista */
/* Retorna el elemnto si lo encuentra o Null si no esta en la lista */
public E search(E e)

/* Verifica si dos listas son iguales */
public boolean equals(Object o)

/* Retorna una copia de una lista dada */
public SinglyLinkedList<E> clone() throws CloneNotSupportedException
```

6. Agregar a la clase **CircularlyLinkedList** los mismos métodos del ejercicio 5.

7. Agregar a la clase **DoublyLinkedList** los mismos métodos del ejercicio 5.

8. Desarrollar un editor de texto elemental utilizando una lista doblemente enlazada. El editor debe permitir agregar líneas al final, insertar y borrar cualquier línea del texto, copiar, mover o borrar un conjunto de líneas. Cada línea del texto está enumerada y el usuario dispone de un menú de opciones para trabajar con el texto.