Estructuras de datos y algoritmos
Laboratorio 2: Listas: ArrayList

Objetivos:
- Utilizar la clase `ArrayList` que implementa el interfaz `List` y conocer su API
- Utilizar los iteradores para recorrer la lista y realizar operaciones sobre ella

Proyecto
- *ArrayList*
- *GestorNaciones*
- *Biblioteca*

Ejercicio I

Implementar un programa que crea una lista de números enteros y ejecuta sobre ella alguna de las siguientes operaciones:

1. Implementar un programa que dado una lista de números enteros y un número entero *x*, muestre en pantalla cuantos números de la lista son mayores que *x*.
2. Implementar un programa que dado una lista de números enteros y un número entero *x*, elimine el número *x* de la lista. En caso de que haya repetidos, solamente la primera ocurrencia.
3. Implementar un programa que dado una lista de números enteros y un número entero *x*, elimine el número *x* de la lista. En caso de que haya repetidos, eliminar todas las ocurrencias.
4. Implementar un programa que dado una lista de números enteros y un entero *x*, elimine el número *x* de la lista. En caso de que haya repetidos, solamente la última ocurrencia.

Analiza cual es el tiempo de ejecución, en notación asintótica, de cada una de las operaciones anteriores.

Ejercicio II

Queremos definir una clase para gestionar las distintas naciones (y sus capitales) del mundo. Las naciones se almacenarán ordenadas alfabéticamente. Con este propósito, nos proporcionan la definición de la clase `Nations`.

```
public class Nations {
  List<Nation> nations; //representa una lista de las naciones (Nation) del mundo

  /**
    * Contruye un gestor de naciones vacío
    */
  public Nations() {}
  /**
    * Dado el nombre de un fichero, carga los datos en una lista
    */
  public void loadData(String file) {}
  /**
    * Visualiza en pantalla todas las naciones (y sus capitales) que gestiona
    */
  public void print() {}
  /**
    * Elimina todo los datos del gestor y lo inicializa para su posterior uso
    */
  public void empty() {}
  /**
    * Dados el nombre de una nación y su capital, se añade a la lista del gestor
    * en orden alfabético de menor a mayor, según el nombre de la nación
    */
  public void insert(String nation, String city) {}
  /**
    * Dado el nombre de una nación, elimina toda la información sobre la nación
    * de la lista del gestor
    */
  public void delete(String nation) {}
}
```

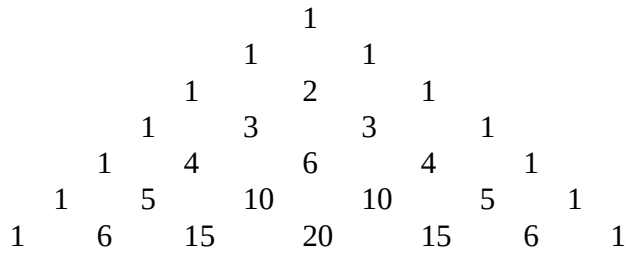
Ayuda: Para leer los datos desde un fichero, se utilizará la clase Token (ver Anexo)

Se pide:

- La implementación de todos los métodos.
- Además de implementar un programa de pruebas con las siguientes operaciones:
  - Cargar un fichero
  - Imprimir la información del gestor
  - Introducir una nueva nación y su capital, y visualizar el resultado
  - Eliminar una capital y visualizar el resultado
  - Obtener la capital de una nación.

Ejercicio III

Escribir un programa que permita visualizar el triángulo de Pascal:



En el triangulo de Pascal, cada número es la suma de los dos números situados encima de él.

Se pide:

a) Resolver el problema utilizando un *array* de una sola dimensión.
b) Resolver el problema utilizando un *ArrayList* de una sola dimensión.
c) Analiza las diferencias de los dos programa escritos anteriormente

Ejercicio IV

Se quiere gestionar una pequeña biblioteca de libros y DVDs de una empresa. La biblioteca esta formada por 5 armarios y cada armario consta de 4 baldas. Cada armario y balda están numerados del 1 al 5 y del 1 al 4, respectivamente. La anchura de las baldas es de 80cm. El número de libros y DVDs que entran en cada balda dependerá del grosor del libro o DVD y están ordenadas alfabéticamente por el titulo. Además del grosor y titulo, por cada libro también se guardarán su ISBN y autor; y por cada DVD su autor y número de discos.

Se pide:

a) Diseñar e implementar las estructuras de datos para representar una biblioteca, un armario y una balda.
b) Implementar un método que dado un Libro o DVD lo inserte en la biblioteca y muestre un mensaje con los números del armario y balda, en el cuál se ha insertado el libro o DVD dado. También hay que tener en cuenta que no se puede sobrepasar la anchura de la balda, por lo tanto, al insertar un libro o DVD puede darse que haya que extraer varios libros del extremo final de la balda y recolocarlos en la siguiente balda o armario.

Ejercicio V

Los resultados de las últimas elecciones a alcalde en Carchelejo han sido los siguientes:

Distrito	*Candidato A*	*Candidato B*	*Candidato C*	*Candidato D*
1	194	48	206	45
2	180	20	320	16
3	221	90	140	20
4	432	50	821	14
5	820	61	946	18

Escribir un programa que haga las siguientes tareas:

a) Imprimir la tabla anterior con cabeceras incluidas
b) Calcular e imprimir el número total de votos recibidos por cada candidato y el porcentaje de total de votos emitidos. Asimismo, visualizar el candidato más votado.
c) Si algún candidato recibe más del 50 por ciento de los votos, el programa imprimirá un mensaje declarando el ganador.
d) Si ningún candidato recibe más del 50 por ciento de lo votos, el programa debe imprimir el nombre de los dos candidatos más votados, que serán los que pasen a la segunda ronda de las elecciones.

Ejercicio VI

Después de estudiar los algoritmos de ordenación y búsqueda, la conclusión es que el coste de ejecutar una búsqueda o cualquier otra operación tiene menor coste si en la estructura de datos los datos están ordenados. Pero esto requiere que previamente se haya tenido que ordenar la estructura de datos: bien aplicando un algoritmo de ordenación o bien desde el inicio se han insertado de forma ordenada siguiendo un criterio. Esta última opción distribuye el coste de ordenación en cada inserción de forma que el coste de una operación (que requiera la ordenación de los datos) sobre la estructura de datos será más eficiente, ya que no hay que aplicar el algoritmo de ordenación.

Teniendo en cuenta lo anterior implementar la estructura de datos `ArrayListOrdenado<T>`, el cual es una colección de elementos de cualquier tipo `T` y estos están ordenados siguiendo un criterio (implementado por la clase que implementa la interfaz `java.util.Comparator<T>`). Implementa los siguientes métodos:

Constructores:

public `ArrayListOrdenado` (Comparator<T> c)
public `ArrayListOrdenado` (Collection <? extends T> col, Comparator c)

Metodos:

//añade el elemento en la lista de forma ordenada, según el comparador utilizado al crear la lista
public boolean add(T element)
public void setComparator(Comparator<T> c)
public int binarySearch(T element)
Más los métodos del interfaz `List<T>`

Anexo I

Información adicional:

class Token { //Definido por el usuario
//Gestiona un fichero que tiene dos palabras por línea

```
// Crea un Token y abre el fichero cuyo nombre es file
public Token(String file)

// devuelve true si hay más líneas para leer, y false en otros casos
public boolean hasMoreTokens()

// devuelve el siguiente token en el fichero
public String getToken() {
```

class String **implements** Comparable<String> { // included in java.lang library

```
/**
  * the value 0 if the argument is a string lexicographically equal to this string;
  * a value less than 0 if the argument is a string lexicographically greater than this string; and
  * a value greater than 0 if the argument is a string lexicographically less than this string.
   */
```

public int **compareTo**(String anotherString)

Example:

```
String s=”dos”;
String s2=”siete”
s.compareTo(s2);      // (result less than 0)
s2.compareTo(s);      // (result greater than 0)
s.compareTo(s)        //  (result equal 0)
```

Todas las clases envoltorio (wrapper), p.ej. Integer, también implementan el interfaz `Comparable<T>`