ESTRUCTURAS DE DATOS
CURSO 2022-23.
Práctica 4: Árboles.

1. OBJETIVOS:

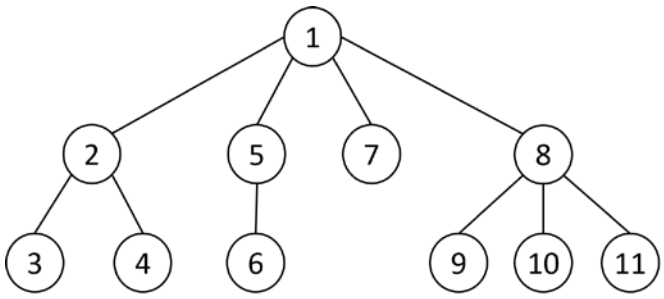
- Definir un TAD Árbol N-ario y añadirle nuevas operaciones.
- Utilizar el TAD Árbol N-ario.
- Definir un TAD Árbol Binario de Búsqueda y añadirle nuevas operaciones.
- Utilizar el TAD Árbol Binario de Búsqueda.

2. Definición del TAD Árbol N-ario.

2.1. El paquete arbolNario en el proyecto "ED 4 Practica. Arboles".

Descargue el archivo "ED 4 Practica. Arboles.zip" de *Moodle* y descomprímalo en una ubicación en la que no pierda el trabajo si se cierra el ordenador. Abra el proyecto con *IntelliJ* y, si aparece el mensaje de error, recuerde definir correctamente el JDK. En la carpeta *src* podrá ver que los archivos se organizan en dos paquetes: *arbolNario* y *arbolBusqueda*, que se corresponden con dos partes diferentes de la práctica. En esta primera parte trabajaremos con el paquete *arbolNario*, cuyos archivos son:

- **ArbolNario**, **NodoArbolNario** son las clases que implementan el TAD Árbol N-ario de enteros. Un Árbol N-ario es una generalización de un Árbol Binario, de manera que cada nodo del árbol puede tener cualquier número de nodos hijos, como el mostrado en la siguiente figura:



- **ListaNodosArbolNario**, **NodoListaNodosArbolNario** son las clases que implementan el TAD Lista Ordinal de nodos del Árbol N-ario utilizando una lista doblemente enlazada. Cada nodo del Árbol N-ario tiene como atributo una lista de este tipo para almacenar todos sus nodos hijos.
- **IteradorAdelanteListaNodosArbolNario**, **IteradorAtrasListaNodosArbolNario** para definir iteradores sobre la lista ordinal de nodos que permitan recorrerla en ambos sentidos,

mediante los métodos *hasNext* y *next* en el primer caso, y *hasPrevious* y *previous* en el segundo.

- **PilaNodosArbolNario**, **ColaNodosArbolNario** para definir los TAD Pila y Cola de nodos del Árbol N-ario utilizando listas enlazadas. Se usarán para recorrer el árbol cuando sean necesarias.
- **Pruebas** es una clase con un método *main* que contiene ya codificadas las pruebas, **sin posible modificación**, que se piden en esta práctica sobre el árbol de ejemplo mostrado en la figura anterior.

2.2. Método mostrarProfundidadRecursivo.

Se pide codificar el siguiente método de la clase *ArbolNario*:

```
public void mostrarProfundidadRecursivo()
```

Este método mostrará por pantalla todos los nodos del árbol, recorriéndolos **recursivamente en preorden**, es decir, para mostrar cada nodo, primero se muestra el entero que contiene y después se muestran todos sus nodos hijos.

Nota: para realizar este método se deberá recorrer la lista de nodos hijos de cada nodo usando un iterador, **no se puede utilizar ninguna estructura de datos adicional**.

El resultado de ejecutar el código de prueba de la clase *Pruebas* debe ser el siguiente:

```
Profundidad Recursivo: 1  2  3  4  5  6  7  8  9  10  11
Profundidad Iterativo:
Amplitud:
```

2.3. Método mostrarProfundidadIterativo.

Se pide codificar el siguiente método de la clase *ArbolNario*:

```
public void mostrarProfundidadIterativo()
```

Este método mostrará por pantalla todos los nodos del árbol de la misma forma que la anterior versión recursiva, pero recorriéndolos **iterativamente**.

Nota: para realizar este método se deberá recorrer la lista de nodos hijos de cada nodo usando un iterador. Además, **solo se puede utilizar una pila de nodos** como estructura de datos adicional.

El resultado de ejecutar el código de prueba de la clase *Pruebas* debe ser el siguiente:

```
Profundidad Recursivo: 1  2  3  4  5  6  7  8  9  10  11
Profundidad Iterativo: 1  2  3  4  5  6  7  8  9  10  11
Amplitud:
```

2.4. Método mostrarAmplitud.

Se pide codificar el siguiente método de la clase *ArbolNario*:

```
public void mostrarAmplitud()
```

Este método mostrará por pantalla todos los nodos del árbol, recorriéndolo **iterativamente en amplitud**.

Nota: para realizar este método se deberá recorrer la lista de nodos hijos de cada nodo usando un iterador. Además, **solo se puede utilizar una cola de nodos** como estructura de datos adicional.

El resultado de ejecutar el código de prueba de la clase *Pruebas* debe ser el siguiente:

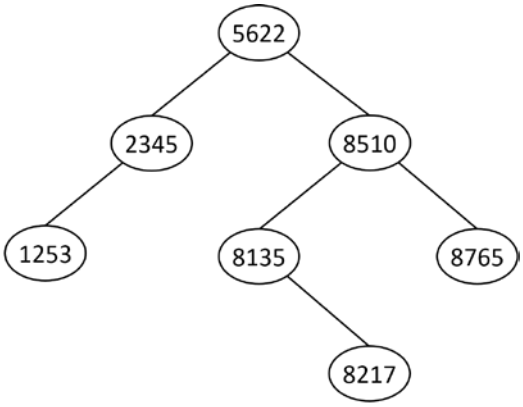
```
Profundidad Recursivo: 1  2  3  4  5  6  7  8  9  10  11
Profundidad Iterativo: 1  2  3  4  5  6  7  8  9  10  11
Amplitud: 1  2  5  7  8  3  4  6  9  10  11
```

3. Definición del TAD Árbol Búsqueda.

3.1 El paquete arbolBusqueda en el proyecto "ED 4 Practica. Arboles".

En esta segunda parte de la práctica trabajaremos con los archivos del paquete *ArbolBusqueda*, que son los siguientes:

- **Arbol**, **NodoArbol** son las clases que implementan el TAD Árbol Búsqueda de objetos de la clase **Alumno** mediante un Árbol Binario de Búsqueda. La clave, es decir, el número de matrícula, viene incluida dentro del objeto de la clase *Alumno* (en su atributo *matricula*). Un ejemplo de un árbol correspondiente a un grupo de alumnos se muestra en la siguiente figura, en la que en los nodos se representa solo el número de matrícula de cada alumno:



- **Pruebas** es una clase con un método *main* que contiene ya codificadas, **sin posible modificación**, las pruebas que se piden en esta práctica sobre el árbol de ejemplo mostrado en la figura anterior.

3.2. Método getNumElementos.

Se pide codificar el siguiente método en la clase *Arbol*:

```
public int getNumElementos()
```

Este método deberá devolver el número de nodos del árbol, recorriéndolo **recursivamente**.

Nota: para realizar este método **no se puede utilizar ninguna estructura de datos adicional**.

3.3. Método getNumMenores.

Se pide codificar el siguiente método en la clase *Arbol*:

```
public int getNumMenores(int clave)
```

Este método deberá devolver el número de nodos del árbol con una clave menor que la proporcionada como parámetro, recorriéndolo **recursivamente**.

Nota: para realizar este método **no se puede utilizar ninguna estructura de datos adicional y se recorrerán el menor número posible de nodos**.

3.4. Método getMenorElemento.

Se pide codificar el siguiente método en la clase *Arbol*:

```
public Alumno getMenorElemento()
```

Este método deberá devolver el alumno del árbol con la menor clave, recorriéndolo **recursivamente**.

Nota: para realizar este método **no se puede utilizar ninguna estructura de datos adicional y se recorrerán el menor número posible de nodos**.

3.5. Método getNumIntermedios.

Se pide codificar el siguiente método en la clase *Arbol*:

```
public int getNumIntermedios(int claveMinimo, int claveMaximo)
```

Este método deberá devolver el número de nodos del árbol con una clave mayor que claveMinimo y menor que claveMaximo, recorriéndolo **recursivamente**.

Nota: para realizar este método **no se puede utilizar ninguna estructura de datos adicional y se recorrerán el menor número posible de nodos**.

3.6. Pruebas de los métodos añadidos.

El resultado de ejecutar el código de prueba de la clase *Pruebas* debe ser el siguiente:

```
----------------- Arbol ----------------
Hay 7 alumnos:
          1253. Felipe Garcia Gomez
      2345. Adriana Torres Pardo
  5622. Alicia Blazquez Martin
          8135. Diego Perez Gonzalez
              8217. Mar Hernando Lopez
      8510. Pedro Jimenez del Pozo
          8765. Eduardo Parra Martin
----------------------------------------

El alumno con menor clave es: 1253. Felipe Garcia Gomez
----------------------------------------

Hay 3 alumnos con clave menor que 5623
Hay 6 alumnos con clave menor que 8511
Hay 2 alumnos con clave menor que 2346
Hay 1 alumnos con clave menor que 1254
Hay 7 alumnos con clave menor que 8766
Hay 4 alumnos con clave menor que 8136
```

```
Hay 5 alumnos con clave menor que 8218
Hay 0 alumnos con clave menor que 1253
---------------------------------------

Hay 0 alumnos con clave mayor que 1000 y menor que 1253
Hay 1 alumnos con clave mayor que 1000 y menor que 2000
Hay 2 alumnos con clave mayor que 1000 y menor que 4000
Hay 3 alumnos con clave mayor que 1000 y menor que 8135
Hay 7 alumnos con clave mayor que 1000 y menor que 9000
Hay 0 alumnos con clave mayor que 2000 y menor que 2200
Hay 4 alumnos con clave mayor que 2344 y menor que 8500
Hay 5 alumnos con clave mayor que 3000 y menor que 9000
Hay 0 alumnos con clave mayor que 9000 y menor que 10000
---------------------------------------
```

4. Entrega de la práctica.

Se entregará el proyecto *IntelliJ* resultante: "ED 4 Practica. Arboles", **que tendrá que tener exactamente ese nombre**. Para ello, se comprimirá el proyecto en un archivo, preferentemente ZIP, y se subirá a la plataforma. El nombre de dicho archivos ZIP será el mismo: "ED 4 Practica. Busqueda.zip".

No olvide que los proyectos deben incluir las pruebas proporcionadas en el código de los métodos *main*.