Práctica 3: TAD AB (Árbol Binario)

Objetivos:

- Diseñar, implementar y usar TADs jerárquicos.

PARTE 1: DISEÑO DE TADs

1) TAD AB:

a. Realizar la **especificación lógica** del **TAD Árbol Binario (AB)** siguiendo las distintas etapas de la "técnica de abstracción de datos" que hemos estudiado.

b. Implementar el TAD AB :

- Paquete modelos: Interfaz I_AB
- Paquete jerárquicos: Clase NodoAB (nodo), Clase AB (árbol binario)

2) TAD ABEnteros:

a. Realizar la **especificación lógica** del **TAD Árbol Binario de elementos enteros (ABEnteros)** siguiendo las distintas etapas de la "técnica de abstracción de datos" que hemos estudiado.

b. Implementar el **TAD ABEnteros** haciendo uso del TAD básico AB - será un AB pero considerando solo claves enteras -

- Paquete modelos: Interfaz I_ABEnteros
- Paquete jerárquicos: Clase ABEnteros

Importante:

- Debéis hacer un **estudio previo** para determinar qué operaciones son básicas en un AB. Repetir el proceso para el AB de Enteros.
- Importante: Tened en cuenta que:
  - Vamos a considerar básicas todas las operaciones que se llevan a cabo en las opciones 2..11 del menú, por tanto, debéis decidir a qué TAD pertenecerán (TAD AB y/o TAD ABEnteros).
  - Vamos a considerar NO básicas, las opciones 0, 1, 12 y 13 del menú. Estas opciones estarán implementadas fuera de los TADs en métodos estáticos.
- Considerad la posibilidad de aplicar herencia entre interfaces y herencia entre clases y justificad en la memoria de la práctica la decisión que habéis tomado al respecto al realizar el diseño de las interfaces y clases consideradas.

PARTE 2: USO DE TADs

3) Implementar una aplicación que presente en pantalla, de forma repetitiva, el siguiente menú de opciones:

MENÚ AB de enteros

1. Crear los AB de enteros

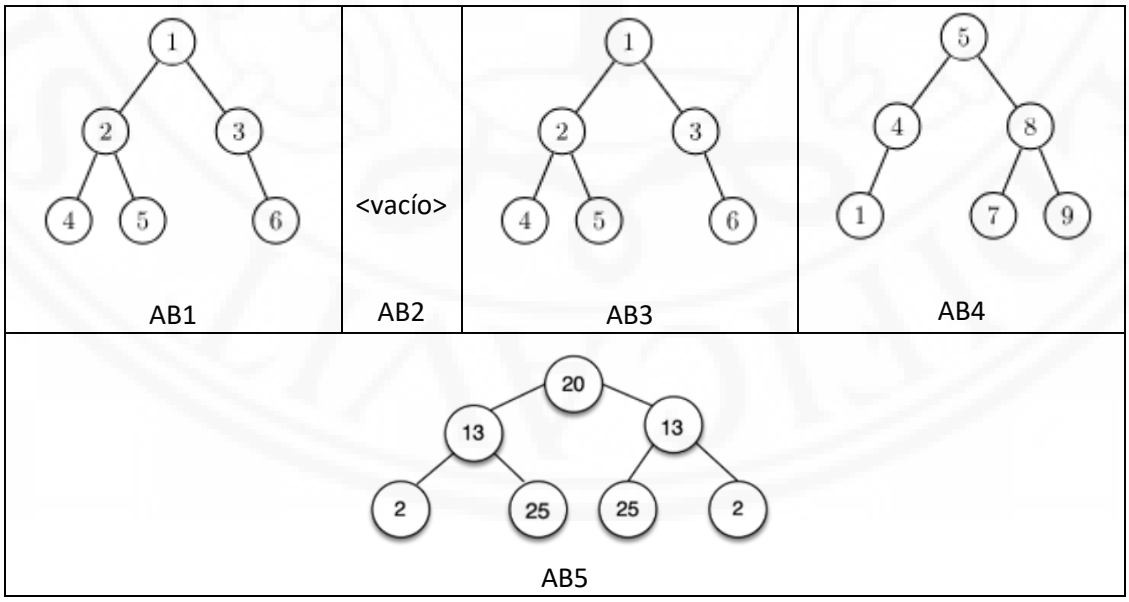
2. Listado de claves en InOrden
3. Listado de claves en InOrden Converso
4. Listado de claves en PreOrden
5. Listado de claves en PostOrden
6. Listado de claves por Niveles
7. Comprobar si dos árboles son iguales
8. Comprobar Simetría de un AB
9. Comprobar Parientes
10. Valor mínimo de cada nivel
11. Comprobar suma de claves
12. Vaciar Árbol Modo 1 (sin recorrer el árbol)
13. Vaciar Árbol Modo 2 (recorriendo todos sus nodos)

0. Salir

A continuación se detalla la tarea que se debe llevar a cabo al ejecutar cada una de las distintas opciones del menú de la aplicación:

Opción 1.- ***"Crear AB de enteros"***: Al ejecutar esta opción se crean los siguientes AB de claves enteras: **(Método static crearABEnteros)**

*** Debéis controlar que se ha ejecutado esta opción al menos una vez, antes de poder ejecutar las opciones restantes del menú.*



Opción 2.- ***"Listado de claves en InOrden"***: Solicita al usuario que indique cuál es el árbol que desea considerar (1..5), y a continuación, realiza un recorrido del AB considerado mostrando en pantalla las claves en InOrden. **(Método recursivo inOrden)**

Opción 3.- ***"Listado de claves en InOrden Converso"***: Solicita al usuario que indique cuál es el árbol que desea considerar (1..5), y a continuación, realiza un recorrido del AB considerado mostrando en pantalla las claves en InOrden Converso. **(Método recursivo inOrdenConverso)**

Opción 4.- ***"Listado de claves en PreOrden"***: Solicita al usuario que indique cuál es el árbol que desea considerar (1..5), y a continuación, realiza un recorrido del AB considerado mostrando en pantalla las claves en PreOrden. **(Método recursivo preOrden)**

Opción 5.- ***"Listado de claves en PostOrden"***: Solicita al usuario que indique cuál es el árbol que desea considerar (1..5), y a continuación, realiza un recorrido del AB considerado mostrando en pantalla las claves en PostOrden. **(Método recursivo postOrden)**

Opción 6.- ***"Listado de claves por Niveles"***: Solicita al usuario que indique cuál es el árbol que desea considerar (1..5), y a continuación, realiza un recorrido del AB considerado mostrando en pantalla las claves por niveles *(recorrido en amplitud)*. **(Método iterativo recorridoPorNiveles)**
(Importante: Haced uso del TAD Cola diseñado en la práctica anterior)

Ejemplos de ejecución de las opciones 2..6 ejecutadas considerando el árbol AB4:

Recorrido en INORDEN (árbol AB4):	1 4 5 7 8 9
Recorrido en INORDEN CONVERSO (árbol AB4):	9 8 7 5 4 1
Recorrido en PREORDEN (árbol AB4):	5 4 1 8 7 9
Recorrido en POSTORDEN (árbol AB4):	1 4 7 9 8 5
Recorrido por NIVELES (árbol AB4):	5 4 8 1 7 9

Opción 7.- ***"Comprobar si dos árboles son iguales"***: Solicita al usuario que indique los dos árboles que desea comparar (1..5), y a continuación, comprueba si ambos árboles son iguales en relación a la disposición de sus nodos y el contenido de éstos. Tened en cuenta que si alguno de los árboles está vacío también se mostrará en pantalla un mensaje indicándolo. Al terminar el proceso de comparación, se debe mostrar en pantalla un mensaje con el resultado de la comparación (IGUALES o DISTINTOS). **(Método recursivo igualesAB que devuelve un booleano)**

Ejemplos de ejecución:

Ejemplo 1:

```
COMPARAR DOS ÁRBOLES

- Indique primer árbol a considerar (1..5):   1
- Indique segundo árbol a considerar (1..5):  2

** Resultado de la comparación:

El árbol AB2 está vacío.
Los árboles AB1 y AB2 son DISTINTOS.
```

Ejemplo 2:

COMPARAR DOS ÁRBOLES

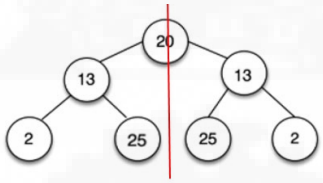
- Indique primer árbol a considerar (1..5): 1
- Indique segundo árbol a considerar (1..5): 3

** Resultado de la comparación:

Los árboles AB1 y AB3 son IGUALES.

Opción 8.- ***"Comprobar simetría"***: Solicita al usuario que indique cuál es el árbol que desea considerar (1..5), y a continuación, comprueba si dicho árbol es simétrico respecto al eje vertical que pasa por su raiz, y muestra en pantalla el correspondiente mensaje "SI es SIMÉTRICO", "NO es SIMÉTRICO" **(Método recursivo esSimetrico que devuelve un booleano)**

Ejemplo: Si consideramos el árbol AB5, podemos observar que SI es simétrico según el criterio anterior (observad cómo están colocadas las claves).



En pantalla se mostrará el siguiente mensaje: "El árbol AB5 SI es SIMÉTRICO"
(Observación: El resto de árboles considerados como ejemplo NO son simétricos respecto al eje vertical que pasa por su raiz).

Opción 9.- ***"Comprobar Parientes"*****:** Solicita al usuario que indique cuál es el árbol que desea considerar (1..5), y también solicita un número entero positivo (X). A continuación, comprueba si el número de ascendientes del nodo X es igual al número de descendientes del mismo. **(Método iterativo comprobarParientes** => devolverá un valor booleano (true/false)

Ejemplos de ejecución:

Ejemplo 1

COMPROBAR PARIENTES

- Indique árbol a considerar (1..5): 1
- ¿Qué clave desea considerar? (nº entero positivo): 2

Resultados

Número de ASCENDIENTES del nodo 2 ➔ 1
Número de DESCENDIENTES del nodo 2 ➔ 2

En el árbol AB1, el nodo 2 NO TIENE el mismo número de ascendientes y descendientes.

Ejemplo 2

COMPROBAR PARIENTES

- Indique árbol a considerar (1..5): 1
- ¿Qué clave desea considerar? (nº entero positivo): 3

Resultados

Número de ASCENDIENTES del nodo 3 ➔ 1
Número de DESCENDIENTES del nodo 3 ➔ 1

En el árbol AB1, el nodo 3 SI TIENE el mismo número de ascendientes y descendientes.

Opción 10.- ***"Valor mínimo de cada nivel"***: Solicita al usuario que indique cuál es el árbol que desea considerar (1..5), y a continuación, crea una lista enlazada con el valor mínimo de cada uno de los niveles del árbol. **(Método iterativo obtenerMinimosPorNivel que devuelve una lista enlazada que contiene los valores mínimos).** Finalmente, se muestra en pantalla los resultados obtenidos.

Observación: Debéis usar LinkedList de Java *(buscad información sobre ello)*.

Ejemplo de ejecución:

VALOR MÍNIMO DE CADA NIVEL

¿Qué arbol desea considerar? (1..5): 1

Mínimos por nivel:

1 2 4

Opción 11.- ***"Comprobar suma de claves"***: Pide al usuario que indique con qué árbol desea trabajar (1..5), y también, un número entero positivo (n). A continuación, comprueba si existe algún camino desde el nodo raíz hasta una hoja del AB cuya suma de claves sea igual al valor n. **(Método recursivo caminoSumaClaves que devuelve un booleano)**

Ejemplo de ejecución 1:

COMPROBAR SUMA DE CLAVES

¿Qué arbol desea considerar? (1..5): 1

Introduzca un número entero positivo: 10

SI existe un camino cuya suma de claves sea igual a 10

Opción 12.- *"Vaciar Árbol Modo 1 (sin recorrer el árbol)"* *:* Solicita al usuario que indique cuál es el árbol que desea considerar (1..5), y a continuación, vacía el árbol considerado sin recorrerlo, haciendo uso exclusivamente del recolector automático de basura que existe en Java. **(Método vaciarArbolModo1)**

Opción 13.- *"Vaciar Árbol Modo 2 (recorriendo todos sus nodos)"*: Solicita al usuario que indique cuál es el árbol que desea considerar (1..5), y a continuación, vacía el árbol considerado realizando un recorrido completo y borrando cada uno de sus nodos *(considerad que la tarea de liberación de espacios de memoria en Java la lleva a cabo el recolector automático de basura).* **(Método recursivo vaciarArbolModo2)**

Pregunta1:

Si estuviésemos trabajando con un lenguaje de programación que no esté dotado de un *"Recolector automático de basura"* (por ej. Lenguaje C), ¿cuáles serían los distintos pasos que debería llevar a cabo el algoritmo que vacía un árbol?.. Escriba el algoritmo en pseudocódigo e inclúyalo en la memoria de la práctica.

Opción 0.- *"Salir"* : Finaliza la ejecución del programa mostrando un mensaje de despedida ("Gracias por utilizar nuestro TAD ABEnteros").

Observaciones:

- Las opciones 2..13 del menú sólo podrán ser ejecutadas si con anterioridad se ha ejecutado la opción 1 *("Crear AB de enteros").*

- Debéis validar los datos que introduce el usuario (X positivo, etc..).

- Podéis diseñar las excepciones que consideréis oportunas.

Pregunta 2:

Una vez finalizada la realización de la práctica, plantearos la siguiente cuestión: ¿Cómo habría cambiado la implementación de la misma si en lugar de considerar AB *(árbol binario)* trabajásemos con ABB *(árbol binario de búsqueda)*?. Justificad vuestra respuesta y anotadlo en la memoria de la práctica.

NORMAS DE REALIZACIÓN DE LA PRÁCTICA:

- Elaborar una memoria completa de la práctica
- Código fuente correctamente comentado (cabeceras descriptivas en cada método, etc.)
- La memoria y el código fuente deberán ser entregados a través del campus virtual de acuerdo a las instrucciones que allí se detallan.

DOCUMENTACIÓN A ENTREGAR

Será preciso realizar una **MEMORIA** que contenga la siguiente información:

1.- **Portada**: Indicando Asignatura, Título de la práctica, Grupo de prácticas, Profesor de prácticas, Autor/es, Curso 2º y Fecha de entrega.
2.- **Especificación lógica** de los TADs considerados (AB, ABEnteros).
3.- **Descripción detallada del funcionamiento de cada método implementado** *(Breve comentario)***.**
4.- **Pruebas de ejecución** de la aplicación: Capturas de pantalla de ejecución de cada una de las opciones del menú de la aplicación.
5.- **Respuesta a las preguntas 1 y 2** incluídas en el enunciado de la práctica.
6.- **Anexo final:** Código fuente completo.

FECHAS:

- Publicación de la práctica: Viernes, 11 de Abril de 2025
- **Entrega de la práctica:** **Domingo, 25 de Mayo de 2025, 22 h** => Pasada esta fecha NO se recogerán más prácticas. No se admitirán entregas por correo electrónico.
