



**UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO**

**FACULTAD DE ESTUDIOS SUPERIORES  
ARAGON**



## **PROYECTO FINAL**

**P R E S E N T A**

Alexis Hernández Zamudio

**A PROFESOR**

Jesús Hernández Cabrera

**Gpo:1158**

URL del repositorio:

**<https://github.com/TyrBalder1439/Estructura-de-Datos-/tree/main/Proyecto%20final>**

**Ciudad Nezahualcóyotl, EDOMEX. 19 DE NOVIEMBRE DEL 2024**

LA TEMATICA UTILIZADA EN ESTE PROYECTO FUE DE LA VENTA DE TACOS.

## -----Clase Producto-----

```
GestorDelInventarios.java  Producto.java  ArbolBinarioBusqueda.java
1  public class Producto implements Comparable<Producto> { 14 usages
2      private String nombre; 4 usages
3      private int id; 6 usages
4      private double precio; 4 usages
5      private int cantidad; 4 usages
6      public Producto(String nombre, int id, double precio, int cantidad) { 3u
7          this.nombre = nombre;
8          this.id = id;
9          this.precio = precio;
10         this.cantidad = cantidad;
11     }
12     public void setCantidad(int cantidad) { no usages
13         this.cantidad = cantidad;
14     }
15     public void setPrecio(double precio) { no usages
16         this.precio = precio;
17     }
18     public void setId(int id) { no usages
19         this.id = id;
20     }
21     public void setNombre(String nombre) { no usages
22         this.nombre = nombre;
23     }
24
25     public String getNombre() { no usages
26         return nombre;
27     }
28     public int getId() { no usages
29         return id;
30     }
31     public double getPrecio() { no usages
32         return precio;
33     }
34     public int getCantidad() { no usages
35         return cantidad;
36
37     @Override
38     public int compareTo(Producto otro) {
39         return Integer.compare(this.id, otro.id);
40     }
41     @Override
42     public String toString() {
43         return "Producto{" +
44             "ID= " + id +
45             ", Nombre = '" + nombre + '\'' +
46             ", Precio = $" + precio +
47             ", Cantidad= " + cantidad +
48             '}';
49     }
50 }
51
```

## CLASE PRODUCTO

La clase Producto es una implementación de la interfaz Comparable<Producto>, por lo que permite saber si los objetos de esta clase son capaces de ser comparados entre sí, con lo que es posible ordenarlos.

Atributos: La clase contiene cuatro atributos privados:

- nombre: un String que almacena el nombre del producto.
- id: un int que indica un identificador único para el producto.
- precio: un double que indica el precio del producto.
- cantidad: un int que indica cuántas unidades del producto se encuentran disponibles.

Constructor: El constructor de la clase permite instanciar un objeto Producto inicializando las propiedades de la forma indicada cuando se crea el objeto.

Métodos set y get: Se incluyen métodos para fijar (set) y conseguir (getter) los valores de los atributos. Con esto se consigue encapsular los datos y controlar los accesos a los mismos.

Método compareTo: Este método permite comparar dos objetos Producto mediante el atributo id. El método devuelve un valor negativo, cero o positivo dependiendo de si el id del objeto actual es menor, igual o mayor que el id del objeto pasado como argumento.

Método toString: El método sobrescribe el método toString de la clase Object y proporciona una representación en forma de cadena del objeto Producto mostrando los atributos de forma legible.

En resumen, esta clase es una representación simple de un producto que incluye atributos, métodos para los atributos y la posibilidad de compararse con otros productos. Esto puede ser útil en aplicaciones que gestionan inventarios o catálogos de productos.

## -----Clase ArbolBinarioBusqueda-----

```
GestorDeInventarios.java  Producto.java  ArbolBinarioBusqueda.java x
1  public class ArbolBinarioBusqueda<T extends Comparable<T>> { 6 usages
2      private class Nodo { 11 usages
3          T dato; 13 usages
4          Nodo izquierdo; 11 usages
5          Nodo derecho; 12 usages
6
7          Nodo(T dato) { 1 usage
8              this.dato = dato;
9              izquierdo = derecho = null;
10         }
11     }
12
13     private Nodo raiz; 6 usages
14
15     public void insertar(T elemento) { 1 usage
16         raiz = insertarRec(raiz, elemento);
17     }
18
19     @ private Nodo insertarRec(Nodo nodo, T elemento) { 3 usages
20         if (nodo == null) {
21             return new Nodo(elemento);
22         }
23         if (elemento.compareTo(nodo.dato) < 0) {
24             nodo.izquierdo = insertarRec(nodo.izquierdo, elemento);
25         } else if (elemento.compareTo(nodo.dato) > 0) {
26             nodo.derecho = insertarRec(nodo.derecho, elemento);
27         } else {
28             throw new IllegalArgumentException("El elemento ya existe en el árbol");
29         }
30         return nodo;
31     }
32
33     public void eliminar(T elemento) { 1 usage
34         raiz = eliminarRec(raiz, elemento);
35     }
```

```

36
37 private Nodo eliminarRec(Nodo nodo, T elemento) { 4 usages
38     if (nodo == null) {
39         throw new IllegalArgumentException("El elemento no existe en el árbol");
40     }
41     if (elemento.compareTo(nodo.dato) < 0) {
42         nodo.izquierdo = eliminarRec(nodo.izquierdo, elemento);
43     } else if (elemento.compareTo(nodo.dato) > 0) {
44         nodo.derecho = eliminarRec(nodo.derecho, elemento);
45     } else {
46         if (nodo.izquierdo == null) {
47             return nodo.derecho;
48         } else if (nodo.derecho == null) {
49             return nodo.izquierdo;
50         }
51         nodo.dato = obtenerMinimo(nodo.derecho);
52         nodo.derecho = eliminarRec(nodo.derecho, nodo.dato);
53     }
54     return nodo;
55 }
56
57 private T obtenerMinimo(Nodo nodo) { 1 usage
58     T minimo = nodo.dato;
59     while (nodo.izquierdo != null) {
60         nodo = nodo.izquierdo;
61         minimo = nodo.dato;
62     }
63     return minimo;
64 }
65
66 public T buscar(T elemento) { 1 usage
67     return buscarRec(raiz, elemento);
68 }
69
70 private T buscarRec(Nodo nodo, T elemento) { 3 usages
71     if (nodo == null) {
72         return null;
73     }
74     if (elemento.compareTo(nodo.dato) < 0) {
75         return buscarRec(nodo.izquierdo, elemento);
76     } else if (elemento.compareTo(nodo.dato) > 0) {
77         return buscarRec(nodo.derecho, elemento);
78     } else {
79         return nodo.dato;
80     }
81 }
82
83 public void inOrden() { 1 usage
84     inOrdenRec(raiz);
85 }
86
87 private void inOrdenRec(Nodo nodo) { 3 usages
88     if (nodo != null) {
89         inOrdenRec(nodo.izquierdo);
90         System.out.println(nodo.dato+" ");
91         inOrdenRec(nodo.derecho);
92     }
93 }
94 }
95

```

## CLASE ARBOL BINARIO DE BUSQUEDA (ABB)

La clase ArbolBinarioBusqueda es una clase propia que tiene un tipo genérico T que implementa la interfaz Comparable, lo que significa que aquellos elementos que se almacenan dentro del árbol deben ser comparables entre si.

La clase ArbolBinarioBusqueda contiene un nodo interno que es de tipo Nodo y que representa un nodo del árbol. Cada uno de los nodos tiene un valor dato y, además, tiene tanto un hijo izquierdo como un hijo derecho.

La clase ArbolBinarioBusqueda tiene varios métodos, que son:

- insertar(T elemento): que inserta en el árbol un nuevo elemento. Si el elemento ya existe, lanza una excepción("El elemento ya existe en el árbol");
- eliminar(T elemento): que elimina del árbol un elemento. Si el elemento no existe, lanza una excepción ("El elemento no existe en el árbol");
- buscar(T elemento): que busca en el árbol un elemento y devuelve el valor si existe, o devuelve null si no existe;
- inOrden(): que imprime los elementos del árbol en orden de elementos inorden (izquierdo, raíz y derecho).

El método insertarRec es un método de carácter recursivo que permite insertar un nuevo elemento dentro del árbol, si el nodo es null crea un nuevo nodo cuyo valor será el de este elemento. En el caso que el elemento sea menor que el valor del nodo actual ejecuta insertarRec al hijo izquierdo. Por el contrario en el caso de que el elemento sea mayor que el valor del nodo actual ejecuta insertarRec al hijo derecho.

El método eliminarRec es un método de carácter recursivo que permite eliminar un elemento del árbol en caso de que el nodo en cuestión sea null se lanza una excepción, en caso de que el elemento sea menor que el del nodo actual se ejecuta eliminarRec al hijo izquierdo, si el elemento es mayor que el valor del nodo actual ejecuta eliminarRec a su hijo derecho, y en el caso en que el elemento es igual que el del nodo actual se elimina el nodo y se sustituye su valor por el mínimo valor del subárbol derecho.

El método obtenerMinimo devuelve el mínimo valor de un subárbol.

El método buscarRec es un método recursivo que busca un elemento en el árbol. Si el nodo es null devolverá null, si el elemento es menor que el del nodo actual ejecuta buscarRec al hijo izquierdo si el elemento es mayor que el del nodo actual ejecuta buscarRec al hijo derecho, si el elemento es igual que el dato del nodo devuelvo su valor.

El método inOrdenRec: Este método recursivo realiza el recorrido en orden. Si el nodo no es null, primero llama a sí mismo para el subárbol izquierdo, luego imprime el dato del nodo actual y finalmente llama a sí mismo para el subárbol derecho. Esto resulta en una salida ordenada de los elementos del árbol.

## CLASE GESTION DE INVENTARIO

```
1  import java.util.Scanner;
2
3  public class GestorDeInventarios {
4      public static void main(String[] args) {
5          ArbolBinarioBusqueda<Producto> inventario = new ArbolBinarioBusqueda<>();
6          Scanner scanner = new Scanner(System.in);
7
8          while (true) {
9              try {
10                 System.out.println("\n--- Gestión de Inventario ---");
11                 System.out.println("1. Agregar Producto");
12                 System.out.println("2. Eliminar Producto");
13                 System.out.println("3. Buscar Producto");
14                 System.out.println("4. Listar Inventario");
15                 System.out.println("5. Salir");
16                 System.out.print("Seleccione una opción: ");
17                 int opcion = Integer.parseInt(scanner.nextLine());
18
19                 switch (opcion) {
20                     case 1:
21                         agregarProducto(scanner, inventario);
22                         break;
23                     case 2:
24                         eliminarProducto(scanner, inventario);
25                         break;
26                     case 3:
27                         buscarProducto(scanner, inventario);
28                         break;
29                     case 4:
30                         listarInventario(inventario);
31                         break;
32                     case 5:
33                         System.out.println("Saliendo del sistema...");
34                         return;
35                     default:
36                         System.out.println("Opción inválida. Intente de nuevo.");
37                 }
38             } catch (NumberFormatException e) {
39                 System.out.println("Entrada inválida. Por favor, ingrese un número válido.");
40             } catch (Exception e) {
41                 System.out.println("Error: " + e.getMessage());
42             }
43         }
44     }
45
46     private static void agregarProducto(Scanner scanner, ArbolBinarioBusqueda<Producto> inventario) {
47         try {
48             System.out.print("Nombre del producto: ");
49             String nombre = scanner.nextLine();
50
51             System.out.print("ID del producto: ");
52             int id = Integer.parseInt(scanner.nextLine());
```

```

53
54     System.out.print("Precio del producto: ");
55     double precio = Double.parseDouble(scanner.nextLine());
56
57     System.out.print("Cantidad en inventario: ");
58     int cantidad = Integer.parseInt(scanner.nextLine());
59
60     if (precio < 0 || cantidad < 0) {
61         System.out.println("El precio y la cantidad deben ser positivos.");
62         return;
63     }
64
65     Producto producto = new Producto(nombre, id, precio, cantidad);
66     inventario.insertar(producto);
67     System.out.println("Producto agregado con éxito.");
68 } catch (NumberFormatException e) {
69     System.out.println("Error: Ingrese valores válidos para ID, precio y cantidad.");
70 } catch (Exception e) {
71     System.out.println("Error: " + e.getMessage());
72 }
73 }
74
75 private static void eliminarProducto(Scanner scanner, ArbolBinarioBusqueda<Producto> inventario) {
76     try {
77         System.out.print("ID del producto a eliminar: ");
78
79         int idEliminar = Integer.parseInt(scanner.nextLine());
80
81         Producto productoEliminar = new Producto("", idEliminar, 0, 0);
82         inventario.eliminar(productoEliminar);
83         System.out.println("Producto eliminado con éxito.");
84     } catch (NumberFormatException e) {
85         System.out.println("Error: Ingrese un ID válido.");
86     } catch (Exception e) {
87         System.out.println("Error: " + e.getMessage());
88     }
89 }
90
91 @ private static void buscarProducto(Scanner scanner, ArbolBinarioBusqueda<Producto> inventario) {
92     try {
93         System.out.print("ID del producto a buscar: ");
94         int idBuscar = Integer.parseInt(scanner.nextLine());
95
96         Producto productoBuscar = new Producto("", idBuscar, 0, 0);
97         Producto encontrado = inventario.buscar(productoBuscar);
98         if (encontrado != null) {
99             System.out.println("Producto encontrado: " + encontrado);
100         } else {
101             System.out.println("Producto no encontrado.");
102         }
103     }

```



```

101         }
102     } catch (NumberFormatException e) {
103         System.out.println("Error: Ingrese un ID válido.");
104     }
105 }
106
107 @
108 private static void listarInventario(ArbolBinarioBusqueda<Producto> inventario) {
109     System.out.println("Inventario en orden:");
110     inventario.inOrden();
111 }
112

```

## GESTOR DE INVENTARIO

Importaciones y Clase Principal:

Importamos java.util.Scanner para poder recibir datos por teclado

La clase GestorDelInventarios contiene el método main que es donde inicia el programa.

Estructura del Menú:

Creamos un bucle while(true) para poder estar interactuando de forma continua hasta que el usuario se decida por salir. Dentro de ese bucle se le presenta al usuario un menú con opciones para añadir, eliminar, buscar y listar productos así como para salir del programa.

Manejo de Opciones: Se utiliza un switch para gestionar la opción seleccionada por el usuario. Según la opción elegida, se invocan diferentes métodos (agregarProducto, eliminarProducto, buscarProducto, listarInventario).

Método agregarProducto: Este método permite al usuario ingresar los detalles de un nuevo producto (nombre, ID, precio y cantidad). Se verifica que el precio y la cantidad sean positivos antes de crear un objeto Producto y añadirlo al inventario.

Método eliminarProducto: Permite al usuario eliminar un producto del inventario ingresando su ID. Se crea un objeto Producto con el ID proporcionado y se llama al método eliminar del árbol binario de búsqueda.

Método buscarProducto: Facilita la búsqueda de un producto en el inventario por su ID. Si se encuentra el producto, se muestra su información; de lo contrario, se notifica que no se encontró.

Método listarInventario: Este método imprime todos los productos en el inventario en orden, utilizando el método inOrden del árbol binario de búsqueda.

Manejo de Errores: Se implementan bloques try-catch para gestionar excepciones, como entradas no válidas, asegurando que el programa no se detenga ante errores.

Los datos se almacenarán en la memoria heap, ya que se crean instancias de objetos como Producto y ArbolBinarioBusqueda, que son gestionados dinámicamente.

## EJECUCION DEL PROGRAMA

- AGREGAR PRODUCTO

```
--- Gestión de Inventario ---
1. Agregar Producto
2. Eliminar Producto
3. Buscar Producto
4. Listar Inventario
5. Salir
Seleccione una opción: 1
Nombre del producto: Toco al pastor
ID del producto: 2
Precio del producto: 15
Cantidad en inventario: 200
Producto agregado con éxito.
```

- ELIMINAR PRODUCTO

```
--- Gestión de Inventario ---
1. Agregar Producto
2. Eliminar Producto
3. Buscar Producto
4. Listar Inventario
5. Salir
Seleccione una opción: 2
ID del producto a eliminar: 2
Producto eliminado con éxito.
```

- BUSCAR PRODUCTO

```

--- Gestión de Inventario ---
1. Agregar Producto
2. Eliminar Producto
3. Buscar Producto
4. Listar Inventario
5. Salir
Seleccione una opción: 3
ID del producto a buscar: 6
Producto encontrado: Producto{ID= 6, Nombre = 'TACOS DE SUADERO ', Precio = $20.0, Cantidad= 250}

```

- LISTAR INVENTARIO

```

--- Gestión de Inventario ---
1. Agregar Producto
2. Eliminar Producto
3. Buscar Producto
4. Listar Inventario
5. Salir
Seleccione una opción: 4
Inventario en orden:
Producto{ID= 1, Nombre = 'CAFE DE OLLA', Precio = $20.0, Cantidad= 200}
Producto{ID= 2, Nombre = 'TACO AL PASTOR ', Precio = $20.0, Cantidad= 200}
Producto{ID= 4, Nombre = 'TACO DE TRIPA ', Precio = $20.0, Cantidad= 150}
Producto{ID= 5, Nombre = 'TACO DE CHORIZO ', Precio = $17.0, Cantidad= 200}
Producto{ID= 6, Nombre = 'TACOS DE SUADERO ', Precio = $20.0, Cantidad= 250}
Producto{ID= 20, Nombre = 'TORTA DE TACOS ', Precio = $35.0, Cantidad= 180}

```

- Ejemplo de duplicación de ID

```

--- Gestión de Inventario ---
1. Agregar Producto
2. Eliminar Producto
3. Buscar Producto
4. Listar Inventario
5. Salir
Seleccione una opción: 1
Nombre del producto: tacos de birria
ID del producto: 2
Precio del producto: 12
Cantidad en inventario: 23
Error: El elemento ya existe en el árbol

```

- Eliminación de productos inexistentes.

```
--- Gestión de Inventario ---
1. Agregar Producto
2. Eliminar Producto
3. Buscar Producto
4. Listar Inventario
5. Salir
Seleccione una opción: 2
ID del producto a eliminar: 5
Error: El elemento no existe en el árbol
```

- Intentos de búsqueda de ID inexistentes

```
--- Gestión de Inventario ---
1. Agregar Producto
2. Eliminar Producto
3. Buscar Producto
4. Listar Inventario
5. Salir
Seleccione una opción: 3
ID del producto a buscar: 9
Producto no encontrado.
```

- Salir

```
--- Gestión de Inventario ---
1. Agregar Producto
2. Eliminar Producto
3. Buscar Producto
4. Listar Inventario
5. Salir
Seleccione una opción: 5
Saliendo del sistema... BYE =)
```