

Practica No.1 Lenguajes Formales

11/02/2025

Equipo 7

Integrantes de equipo:

Angel Orlando Niño Noriega – 22050705

David Sebastián de la Fuente Monjaras – 22050778

Mauro Rodrigo Ruiz Alvarez - 22050727

Objetivo

El alumno identifica y clasifica traductores, así como ejerce la utilización de archivos para el almacenamiento y lectura de información.

Material a utilizar

Internet

Bibliografía

Computadora

Procesador de texto

Lenguaje de programación / herramientas

Distribución :

Programa en Java (Capturas):

Mauro Rodrigo Ruiz Alvarez:

1. Creación de la clase principal Almacen:

Mauro fue el responsable de crear la clase principal Almacen, que gestiona la interacción del usuario con el sistema. Esta clase contiene el ciclo principal del programa, donde el usuario puede agregar artículos, consultar artículos y salir del sistema.

2. Gestión de la entrada de datos:

Mauro también se encargó de permitir que el usuario ingresara los datos de los artículos (nombre, existencia y costo). Además, fue el responsable de validar que los datos ingresados fueran correctos. Específicamente, validó que el nombre del artículo no estuviera vacío y que la existencia fuera un número entero y el costo un número decimal.

Tareas específicas de Mauro:

Nombre del artículo: Validó que no fuera vacío y solicitó al usuario que lo ingresara de nuevo si no lo proporcionaba.

Existencia y costo: Validó que la existencia fuera un número entero y que el costo fuera un número decimal. Si se ingresaba un valor incorrecto, pidió al usuario que lo intentara nuevamente.

```
case 1:
    String nombre;
    while (true) {
        System.out.print("Nombre del artículo: ");
        nombre = scanner.nextLine().trim();
        if (!nombre.isEmpty()) {
            break;
        }
        System.out.println("El nombre no puede estar vacío. Inténtalo de nuevo.");
    }

    System.out.print("Existencia: ");
    int existencia;
    try {
        existencia = Integer.parseInt(scanner.nextLine());
    } catch (NumberFormatException e) {
        System.out.println("Debe ser un número entero.");
        continue;
    }

    System.out.print("Costo: ");
    double costo;
    try {
        costo = Double.parseDouble(scanner.nextLine());
    } catch (NumberFormatException e) {
        System.out.println("Debe ser un número decimal.");
        continue;
    }
}
```

Orlando Noriega:

1. Creación de la clase Artículo:

Orlando fue el responsable de crear la clase Artículo. Esta clase contiene los atributos nombre, existencia y costo para cada artículo, y también se encargó de definir el constructor que inicializa estos atributos.

```
class Artículo {
    private String nombre;
    private int existencia;
    private double costo;

    public Artículo(String nombre, int existencia, double costo) {
        this.nombre = nombre;
        this.existencia = existencia;
        this.costo = costo;
    }
}
```

2. Persistencia de datos en el archivo almacen.txt:

Orlando fue el encargado de asegurarse de que los datos ingresados por el usuario se guardaran correctamente en un archivo de texto. Utilizó un

BufferedWriter para escribir la información de cada artículo en el archivo almacen.txt.

```
try (BufferedWriter bw = new BufferedWriter(new FileWriter(archivo, true))) {  
    // Formato alineado  
    String lineaFormateada = String.format("%-15s %-12d $%-7.2f", nombre, existencia, costo);  
    bw.write(lineaFormateada);  
    bw.newLine();  
} catch (IOException e) {  
    System.out.println("Error al escribir en el archivo.");  
}  
break;
```

3. Menú de opciones para el usuario:

Orlando fue también el responsable de diseñar el menú que permite al usuario elegir entre agregar un artículo, consultar artículos o salir del sistema.

```
public static void main(String[] args) throws IOException {  
    String nombreArchivo = "almacen.txt";  
    String ruta = "C:\\Users\\Acer\\Desktop\\Auto\\";  
    File archivo = new File(ruta + nombreArchivo);  
  
    Scanner scanner = new Scanner(System.in);  
    ArrayList<Articulo> articulos = new ArrayList<>();  
  
    while (true) {  
        System.out.println("\n--- Sistema de Artículos ---");  
        System.out.println("1. Agregar Artículo");  
        System.out.println("2. Consultar Artículos");  
        System.out.println("3. Salir");  
        System.out.print("Selecciona una opción: ");
```

Tareas específicas de Orlando:

Guardar los artículos en el archivo: Se encargó de escribir los datos de los artículos en el archivo de manera ordenada, asegurándose de que cada artículo estuviera formateado correctamente (con nombre, existencia y costo alineados).

Verificación del archivo: Se encargó de verificar que el archivo existiera y de que no se generaran errores al escribir los datos. En caso de un error, implementó un manejo básico de excepciones.

David de la Fuente:

1. Lectura de los datos del archivo almacen.txt:

David fue el responsable de leer los artículos almacenados en el archivo `almacen.txt` y mostrar esa información en consola. Implementó un sistema que leía el archivo línea por línea y lo mostraba al usuario de manera estructurada.

Tareas específicas de David:

Verificación del archivo antes de leer: Se encargó de verificar que el archivo existiera y no estuviera vacío antes de intentar leerlo.

Lectura y visualización de los datos: Utilizó un `BufferedReader` para leer el archivo y mostrar los artículos guardados en la consola de manera organizada. Los artículos se presentaban con un formato alineado que permitía ver claramente el nombre, existencia y costo de cada uno.

Manejo de errores al leer: Si ocurría algún error al leer el archivo (por ejemplo, si el archivo no existía o estaba vacío), David implementó un manejo de excepciones para mostrar un mensaje adecuado al usuario.

2. Salida del sistema:

David también fue el responsable de implementar la salida del sistema. Al seleccionar la opción correspondiente en el menú, David aseguró que el programa terminara correctamente, cerrando los recursos como el `Scanner` y mostrando un mensaje final al usuario, como "Saliendo del sistema...".

```
case 2:
    if (archivo.exists() && archivo.length() > 0) {
        System.out.println("\n--- Artículos Registrados ---");
        System.out.printf("%-15s %-12s %-8s\n", "Nombre", "Existencia", "Costo");
        System.out.println("-----");

        try (BufferedReader br = new BufferedReader(new FileReader(archivo))) {
            String linea;
            while ((linea = br.readLine()) != null) {
                System.out.println(linea);
            }
        } catch (IOException e) {
            System.out.println("Error al leer el archivo.");
        }
    } else {
        System.out.println("El archivo no existe o está vacío.");
    }
    break;

case 3:
    System.out.println("Saliendo del sistema...");
    scanner.close();
    return;

default:
    System.out.println("Opción inválida. Inténtalo de nuevo.");
}
}
```

Programa en Java (Código completo):

Clase Artículo:

```
Articulo.java x Almacen.java
1 package U1;
2
3 public class Articulo {
4     private String Nombre;
5     private int Existencia;
6     private double costo;
7
8     public Articulo(String nombre, int Existencia, double costo) {
9         super();
10        Nombre = nombre;
11        this.Existencia = Existencia;
12        this.costo = costo;
13    }
14
15    public Articulo() {
16        super();
17        // TODO Auto-generated constructor stub
18    }
19
20    public String getNombre() {
21        return Nombre;
22    }
23
24    public void setNombre(String nombre) {
25        Nombre = nombre;
26    }
27
28    public int getExistencia() {
29        return Existencia;
30    }
31
32    public void setExistencia(int existencia) {
33        Existencia = existencia;
34    }
35
36    public double getCosto() {
37        return costo;
38    }
39
40    ,
41
42    public void setCosto(double costo) {
43        this.costo = costo;
44    }
45
46    @Override
47    public String toString() {
48        return "Articulo: \n" + "-----Nombre-----" + Nombre
49            + "-----Existencia-----" + Existencia + "-----costo-----" + costo;
50    }
51
52 }
```

Clase almacén:

```

1 package UI;
2
3 import java.io.BufferedReader;
4
5
6
7
8
9
10
11
12
13 /*
14  *
15  * Autores: Mauro Rodrigo Ruiz Alvarez
16  *          Orlando Noriega
17  *          David de la Fuente
18  *
19  * PROGRAMA
20  * Realice una aplicación en el lenguaje de programación JAVA, que almacene la siguiente información en un archivo de texto.
21  * Para cada Artículo de papelería la siguiente información:
22  *
23  * Nombre del artículo Existencia Costo
24  *
25  * El usuario podrá ingresar la cantidad de artículos que desee, cuando el usuario ya no ingrese más información, deberá guardarse en un archivo de texto.
26  * Además deberá mostrar la información almacenada, leyendo el archivo de texto que almacena.
27  */
28 public class Almacen {
29
30     public static void main(String[] args) throws IOException {
31         String nombreArchivo = "almacen.txt";
32         String ruta = "C:\\Users\\Mauro\\Desktop\\Automatas\\";
33         File archivo = new File(ruta + nombreArchivo);
34
35         Scanner scanner = new Scanner(System.in);
36         ArrayList<Articulo> articulos = new ArrayList<>();
37
38         while (true) {
39             System.out.println("\n--- Sistema de Artículos ---");
40             System.out.println("1. Agregar Artículo");
41             System.out.println("2. Consultar Artículos");
42             System.out.println("3. Salir");
43             System.out.print("Selecciona una opción: ");
44
45             int opcion;

```

```

45         int opcion;
46         try {
47             opcion = Integer.parseInt(scanner.nextLine());
48         } catch (NumberFormatException e) {
49             System.out.println("Por favor, introduce un número válido.");
50             continue;
51         }
52
53         switch (opcion) {
54             case 1:
55                 String nombre;
56                 while (true) {
57                     System.out.print("Nombre del artículo: ");
58                     nombre = scanner.nextLine().trim();
59                     if (nombre.isEmpty()) {
60                         break;
61                     }
62                     System.out.println("El nombre no puede estar vacío. Inténtalo de nuevo.");
63                 }
64
65                 System.out.print("Existencia: ");
66                 int existencia;
67                 try {
68                     existencia = Integer.parseInt(scanner.nextLine());
69                 } catch (NumberFormatException e) {
70                     System.out.println("Debe ser un número entero.");
71                     continue;
72                 }
73
74                 System.out.print("Costo: ");
75                 double costo;
76                 try {
77                     costo = Double.parseDouble(scanner.nextLine());
78                 } catch (NumberFormatException e) {
79                     System.out.println("Debe ser un número decimal.");
80                     continue;
81                 }
82

```

```

82         articulos.add(new Articulo(nombre, existencia, costo));
83         System.out.println("Artículo registrado correctamente.");
84
85         try (BufferedWriter bw = new BufferedWriter(new FileWriter(archivo, true))) {
86             // Formato alineado
87             String lineaFormateada = String.format("%-15s %-12d $%-7.2f", nombre, existencia, costo);
88             bw.write(lineaFormateada);
89             bw.newLine();
90         } catch (IOException e) {
91             System.out.println("Error al escribir en el archivo.");
92         }
93         break;
94
95     case 2:
96         if (archivo.exists() && archivo.length() > 0) {
97             System.out.println("\n--- Artículos Registrados ---");
98             System.out.printf("%-15s %-12s %-8s\n", "Nombre", "Existencia", "Costo");
99             System.out.println("-----");
100
101             try (BufferedReader br = new BufferedReader(new FileReader(archivo))) {
102                 String linea;
103                 while ((linea = br.readLine()) != null) {
104                     System.out.println(linea);
105                 }
106             } catch (IOException e) {
107                 System.out.println("Error al leer el archivo.");
108             }
109         } else {
110             System.out.println("El archivo no existe o está vacío.");
111         }
112         break;
113
114     case 3:
115         System.out.println("Saliendo del sistema...");
116         scanner.close();
117         return;
118
119     default:

```

```
        case 3:
            System.out.println("Saliendo del sistema...");
            scanner.close();
            return;

        default:
            System.out.println("Opción inválida. Inténtalo de nuevo.");
    }
}
}
```

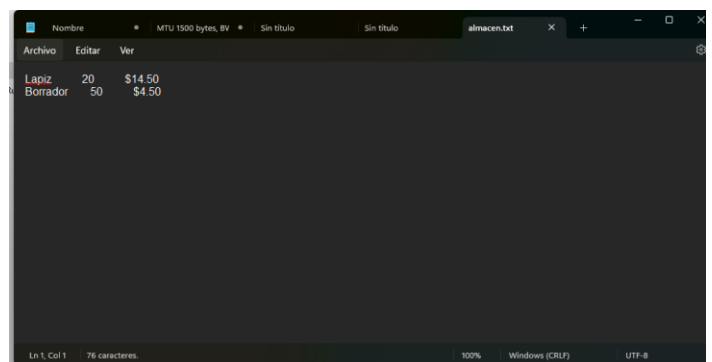
Código ejecutado:

```
--- Sistema de Artículos ---
1. Agregar Artículo
2. Consultar Artículos
3. Salir
Selecciona una opción: 1
Nombre del artículo: Lapis
Existencia: 20
Costo: 14.5
Artículo registrado correctamente.

--- Sistema de Artículos ---
1. Agregar Artículo
2. Consultar Artículos
3. Salir
Selecciona una opción: 1
Nombre del artículo: Borrador
Existencia: 50
Costo: 4.5
Artículo registrado correctamente.
```

```
--- Sistema de Artículos ---
1. Agregar Artículo
2. Consultar Artículos
3. Salir
Selecciona una opción: 2
|
--- Artículos Registrados ---
Nombre          Existencia  Costo
-----
Lapiz           20          $14.50
Borrador        50          $4.50

--- Sistema de Artículos ---
1. Agregar Artículo
2. Consultar Artículos
3. Salir
Selecciona una opción:
```



The screenshot shows a text editor window with the file 'almacen.txt' open. The window has a menu bar with 'Archivo', 'Editar', and 'Ver'. The text content is as follows:

Lapiz	20	\$14.50
Borrador	50	\$4.50

The status bar at the bottom indicates 'Ln 1, Col 1', '76 caracteres', '100%', 'Windows (CRLF)', and 'UTF-8'.

Conclusiones:

- David Sebastian de la Fuente Monjaras - 22050778

Yo pienso que la identificación y clasificación de traductores es fundamental para comprender cómo los lenguajes de programación se convierten en instrucciones ejecutables por una computadora. Al mismo tiempo, se me hizo muy interesante la manipulación de archivos para el almacenamiento y lectura de información permite organizar y gestionar datos de manera eficiente.

- Mauro Rodrigo Ruiz Alvarez - 22050727

Reconocer el papel de los traductores nos permite ver cómo se transforman las ideas en instrucciones para la computadora, mientras que el manejo adecuado de archivos facilita la organización y acceso a la información de manera clara.

- Angel Orlando Niño Noriega – 22050705

El proyecto resultó en una solución funcional y eficiente para el manejo de información de artículos, integrando correctamente el almacenamiento de datos en un archivo de texto. Fue interesante trabajar en la persistencia de información, asegurando que los datos fueran almacenados de manera clara y accesible para futuras consultas. La experiencia permitió aplicar buenas prácticas de manejo de archivos y consolidar conocimientos sobre el manejo de excepciones. El resultado final demuestra la importancia de una arquitectura bien estructurada y un manejo adecuado de la información.

Bibliografías:

- Aho Alfred V., U. J. (2007). Compiladores. Principios, técnicas y herramientas (2da. ed.). México: Pearson Educación
- Alfonseca Moreno, M. (2006). Compiladores e intérpretes: teoría y práctica (1ra ed.). España: Pearson/ Prentice Hall.
- Ruíz, J. (2009). Compiladores -Teoría e implementación. México: Alfaomega.
- Grune, Dick. (2007). Diseño de compiladores modernos. McGraw-Hill.