| **Asignatura** | **Programación II** |
| --- | --- |
| **Carrera** | **Ing. en Informática** |
| **Plan** | **Ajuste 2023** |
| **Ciclo** | **2do** |
| **Cuatrimestre** | **1ero** |
| **Tema/Título** | **Practica 6** |
| **Profesor** | **Adrian Tozzi** |

**Grupo de Trabajo**

| **ID/Matrícula** | **APELLIDO, Nombres** | **Correo Electrónico** |
| --- | --- | --- |
| **000-20-2323** | Cannizzaro, Pablo | Pablom.cannizzaro@comunidad.ub.edu.ar |
| **000-20-1394** | Callizaya, Leandro | [leandroa.callizaya@comunidad.ub.edu.ar](mailto:leandro.callizaya@comunidad.ub.edu.ar) |
| **000-20-1953** | Conde Buades, Joaquín | Joaquin.conde@comunidad.ub.edu.ar |
| **000-20-1212** | Franco Dalla Via Oliveros | Franco.dalla@comunidad.ub.edu.ar |

**Grilla de calificación**

| **Concepto** | **Propuesta** | **Marco Teórico** | **Desarrollo propio** | **Conclusiones** | **Fuentes y Referencias** |
| --- | --- | --- | --- | --- | --- |
| **Sobresaliente (10)** |  |  |  |  |  |
| **Distinguido (9-8)** |  |  |  |  |  |
| **Bueno (7-6)** |  |  |  |  |  |
| **Aprobado (5-4)** |  |  |  |  |  |
| **Insuficiente (3-2-1)** |  |  |  |  |  |
| **Reprobado (0)** |  |  |  |  |  |
| **NOTA** |  | | | | |

| Comentario adicional del Profesor: |
| --- |

**PRÁCTICA 6**

**Ejercicio 1:**

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

import java.util.Scanner;

public class ContadorArchivoSimple {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.*in*);

// Crear el archivo con un string fijo

String nombreArchivo = "frase.txt";

String contenidoArchivo = "Hola mundo, esto es una prueba.";

try (BufferedWriter writer = new BufferedWriter(new FileWriter(nombreArchivo))) {

writer.write(contenidoArchivo);

System.*out*.println("Archivo '" + nombreArchivo + "' creado con contenido: " + contenidoArchivo);

} catch (IOException e) {

System.*out*.println("Error al crear el archivo: " + e.getMessage());

return;

}

// Pedir opción al usuario

System.*out*.print("Ingrese opción (c = caracteres, l = líneas, cl = ambos): ");

String opciones = scanner.nextLine().toLowerCase();

boolean contarCaracteres = opciones.contains("c");

boolean contarLineas = opciones.contains("l");

int cantidadCaracteres = 0;

int cantidadLineas = 0;

// Leer el archivo y contar

try (BufferedReader br = new BufferedReader(new FileReader(nombreArchivo))) {

String linea;

while ((linea = br.readLine()) != null) {

if (contarLineas) cantidadLineas++;

if (contarCaracteres) cantidadCaracteres += linea.length();

}

if (contarCaracteres) System.*out*.println("Cantidad de caracteres: " + cantidadCaracteres);

if (contarLineas) System.*out*.println("Cantidad de líneas: " + cantidadLineas);

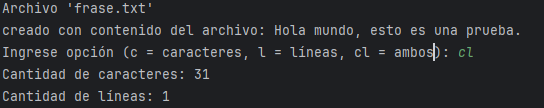
} catch (IOException e) {

System.*out*.println("Error al leer el archivo: " + e.getMessage());

}

}

}

**CONSOLA:  
**

**Ejercicio 2:**

import java.io.\*;

import java.util.Scanner;

public class ConversorTexto {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.*in*);

String rutaEntrada = "entrada.txt";

String rutaSalida = "salida.txt";

String contenidoFijo = "Hola Mundo. Este es un texto de prueba.";

// Crear archivo de entrada con contenido fijo

try (BufferedWriter escritorInicial = new BufferedWriter(new FileWriter(rutaEntrada))) {

escritorInicial.write(contenidoFijo);

} catch (IOException e) {

System.*out*.println("Error al crear archivo de entrada: " + e.getMessage());

return;

}

System.*out*.print("¿Convertir a (M)ayúsculas o (m)inúsculas?: ");

String opcion = scanner.nextLine();

if (opcion.equals("M")) {

// Convertir a MAYÚSCULAS

*convertirArchivo*(rutaEntrada, rutaSalida, true);

} else if (opcion.equals("m")) {

// Convertir a minúsculas

*convertirArchivo*(rutaEntrada, rutaSalida, false);

} else {

System.*out*.println("Opción inválida. Por favor, ingresa solo 'M' o 'm'.");

scanner.close();

return;

}

scanner.close();

}

private static void convertirArchivo(String rutaEntrada, String rutaSalida, boolean aMayusculas) {

System.*out*.println("\nTexto convertido a " + (aMayusculas ? "MAYÚSCULAS" : "minúsculas") + ":\n");

try (

BufferedReader lector = new BufferedReader(new FileReader(rutaEntrada));

BufferedWriter escritor = new BufferedWriter(new FileWriter(rutaSalida))

) {

String linea;

while ((linea = lector.readLine()) != null) {

String lineaConvertida = aMayusculas ? linea.toUpperCase() : linea.toLowerCase();

escritor.write(lineaConvertida);

escritor.newLine();

System.*out*.println(lineaConvertida);

}

System.*out*.println("\nTexto convertido guardado en: " + rutaSalida);

} catch (IOException e) {

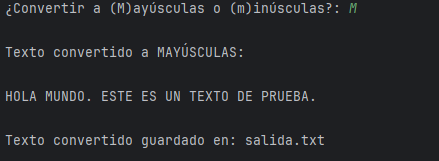
System.*out*.println("Ocurrió un error: " + e.getMessage());

}

}

}

**CONSOLA:**

****

**EJERCICIO 3:**

import java.io.\*;

import java.util.Scanner;

public class SepararPaisesPorPoblacion {

public static void main(String[] args) {

String archivoEntrada = "paises.txt";

String archivoMenos30 = "paises\_menor\_30.txt";

String archivoMayorIgual30 = "paises\_mayor\_igual\_30.txt";

// 1. Crear archivo de entrada con formato legible

String[][] datosPaises = {

{"Argentina", "45.3", "45000.5"},

{"Uruguay", "3.5", "18000.2"},

{"Paraguay", "7.0", "12000.7"},

{"Chile", "19.5", "28000.3"},

{"Brasil", "211.0", "150000.1"},

{"Bolivia", "11.6", "9000.0"}

};

try (BufferedWriter escritorEntrada = new BufferedWriter(new FileWriter(archivoEntrada))) {

for (String[] pais : datosPaises) {

escritorEntrada.write(String.*format*("NOMBRE DEL PAÍS: %s%n", pais[0]));

escritorEntrada.write(String.*format*("POBLACIÓN: %s%n", pais[1]));

escritorEntrada.write(String.*format*("PBI: %s%n", pais[2]));

escritorEntrada.write(System.*lineSeparator*()); // línea vacía entre países

}

System.*out*.println("Archivo de entrada creado con datos fijos y formato legible.");

} catch (IOException e) {

System.*out*.println("Error al crear archivo de entrada: " + e.getMessage());

return;

}

// 2. Leer archivo de entrada y separar en dos archivos según población

try (

BufferedReader lector = new BufferedReader(new FileReader(archivoEntrada));

BufferedWriter escritorMenor30 = new BufferedWriter(new FileWriter(archivoMenos30));

BufferedWriter escritorMayorIgual30 = new BufferedWriter(new FileWriter(archivoMayorIgual30));

) {

String nombrePais = null;

String poblacionStr = null;

String pbiStr = null;

String linea;

while ((linea = lector.readLine()) != null) {

linea = linea.trim();

if (linea.isEmpty()) {

// línea vacía indica que se terminó un registro

if (nombrePais != null && poblacionStr != null && pbiStr != null) {

double poblacion = Double.*parseDouble*(poblacionStr);

BufferedWriter escritor = poblacion < 30 ? escritorMenor30 : escritorMayorIgual30;

// Escribir el bloque completo escritor.write("NOMBRE DEL PAÍS: " + nombrePais);

escritor.newLine();

escritor.write("POBLACIÓN: " + poblacionStr);

escritor.newLine();

escritor.write("PBI: " + pbiStr);

escritor.newLine();

escritor.newLine();

}

// resetear variables para próximo país

nombrePais = null;

poblacionStr = null;

pbiStr = null;

} else {

if (linea.startsWith("NOMBRE DEL PAÍS:")) {

nombrePais = linea.substring("NOMBRE DEL PAÍS:".length()).trim();

} else if (linea.startsWith("POBLACIÓN:")) {

poblacionStr = linea.substring("POBLACIÓN:".length()).trim();

} else if (linea.startsWith("PBI:")) {

pbiStr = linea.substring("PBI:".length()).trim();

}

}

}

// Para el último país si el archivo no termina con línea vacía

if (nombrePais != null && poblacionStr != null && pbiStr != null) {

double poblacion = Double.*parseDouble*(poblacionStr);

BufferedWriter escritor = poblacion < 30 ? escritorMenor30 : escritorMayorIgual30;

escritor.write("NOMBRE DEL PAÍS: " + nombrePais);

escritor.newLine();

escritor.write("POBLACIÓN: " + poblacionStr);

escritor.newLine();

escritor.write("PBI: " + pbiStr);

escritor.newLine();

escritor.newLine();

}

System.*out*.println("Archivos generados correctamente con formato legible.");

} catch (IOException | NumberFormatException e) {

System.*out*.println("Error al procesar archivos: " + e.getMessage());

return;

}

// 3. Preguntar qué archivo mostrar

Scanner scanner = new Scanner(System.*in*);

System.*out*.println("\n¿Qué archivo querés ver?");

System.*out*.println("1 - Países con población MENOR a 30 millones");

System.*out*.println("2 - Países con población MAYOR o IGUAL a 30 millones");

System.*out*.print("Elegí 1 o 2: ");

String opcion = scanner.nextLine();

String archivoAMostrar = null;

if (opcion.equals("1")) {

archivoAMostrar = archivoMenos30;

} else if (opcion.equals("2")) {

archivoAMostrar = archivoMayorIgual30;

} else {

System.*out*.println("Opción inválida. Saliendo...");

scanner.close();

return;

}

// 4. Mostrar archivo con formato legible

System.*out*.println("\nContenido del archivo seleccionado (" + archivoAMostrar + "):\n");

try (BufferedReader lector = new BufferedReader(new FileReader(archivoAMostrar))) {

String linea;

while ((linea = lector.readLine()) != null) {

System.*out*.println(linea);

}

} catch (IOException e) {

System.*out*.println("Error al leer el archivo: " + e.getMessage());

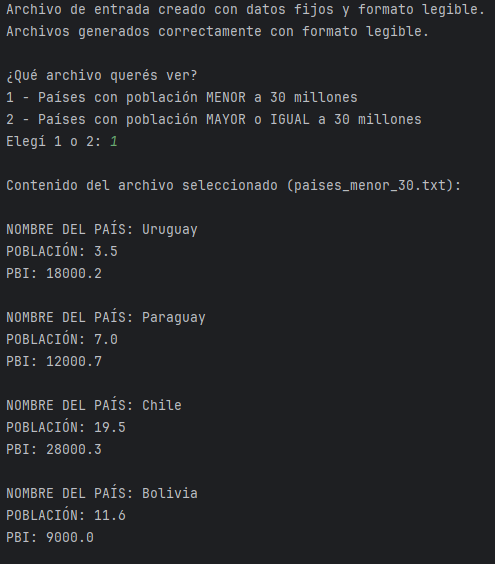
}

scanner.close();

}

}

**CONSOLA:**

****

**EJERCICIO 4:**

import java.io.\*;

import java.util.\*;

public class GestionClientes {

public static void main(String[] args) {

List<String> clientesIniciales = *inicializarClientes*();

String archivoClientes = "Clientes.txt";

String archivoClientes2 = "Clientes2.txt";

// 1. Guardar los 10 clientes iniciales en archivo Clientes.txt y mostrar

*guardarClientesEnArchivo*(clientesIniciales, archivoClientes);

System.*out*.println("Clientes iniciales guardados:");

*mostrarArchivo*(archivoClientes);

// 2. Pedir 5 clientes nuevos por teclado y agregarlos al principio

List<String> nuevosClientes = *pedirClientesPorTeclado*(5);

List<String> todosClientes = new ArrayList<>();

todosClientes.addAll(nuevosClientes); // primero los nuevos

todosClientes.addAll(clientesIniciales); // después los iniciales

*guardarClientesEnArchivo*(todosClientes, archivoClientes);

System.*out*.println("\nArchivo actualizado con 5 nuevos clientes al principio:");

*mostrarArchivo*(archivoClientes);

// 3. Crear archivo Clientes2.txt con número de cuenta y tipo de cuenta

*generarArchivoClientes2*(todosClientes, archivoClientes2);

System.*out*.println("\nArchivo Clientes2.txt generado con más datos:");

*mostrarArchivo*(archivoClientes2);

}

// Inicializa lista con 10 clientes fijos

public static List<String> inicializarClientes() {

return new ArrayList<>(Arrays.*asList*(

"Juan Pérez",

"María López",

"Carlos Sánchez",

"Ana Gómez",

"Luis Fernández",

"Sofía Martínez",

"Pedro Díaz",

"Laura Romero",

"Jorge Herrera",

"Marta Castro"

));

}

// Guarda la lista de clientes en un archivo (uno por línea)

public static void guardarClientesEnArchivo(List<String> clientes, String nombreArchivo) {

try (BufferedWriter escritor = new BufferedWriter(new FileWriter(nombreArchivo))) {

for (String cliente : clientes) {

escritor.write(cliente);

escritor.newLine();

}

} catch (IOException e) {

System.*out*.println("Error al guardar archivo " + nombreArchivo + ": " + e.getMessage());

}

}

// Muestra el contenido de un archivo por pantalla

public static void mostrarArchivo(String nombreArchivo) {

try (BufferedReader lector = new BufferedReader(new FileReader(nombreArchivo))) {

String linea;

while ((linea = lector.readLine()) != null) {

System.*out*.println(linea);

}

} catch (IOException e) {

System.*out*.println("Error al leer archivo " + nombreArchivo + ": " + e.getMessage());

}

}

// Pide por teclado n clientes (nombre y apellido)

public static List<String> pedirClientesPorTeclado(int cantidad) {

Scanner scanner = new Scanner(System.*in*);

List<String> nuevosClientes = new ArrayList<>();

System.*out*.println("\nIngrese " + cantidad + " nuevos clientes (nombre y apellido):");

for (int i = 1; i <= cantidad; i++) {

System.*out*.print("Cliente " + i + ": ");

String cliente = scanner.nextLine().trim();

while (cliente.isEmpty()) {

System.*out*.print("No puede estar vacío. Reingrese Cliente " + i + ": ");

cliente = scanner.nextLine().trim();

}

nuevosClientes.add(cliente);

}

// No cerramos scanner para evitar cerrar System.in (puede causar problemas en IDEs)

return nuevosClientes;

}

// Genera archivo Clientes2.txt con número de cuenta y tipo de cuenta

public static void generarArchivoClientes2(List<String> clientes, String nombreArchivo) {

Random rnd = new Random();

String[] tiposCuenta = {"Caja de ahorro", "Cuenta corriente"};

try (BufferedWriter escritor = new BufferedWriter(new FileWriter(nombreArchivo))) {

for (String cliente : clientes) {

// Generar número de cuenta aleatorio: 10 dígitos

long numCuenta = 1000000000L + (Math.*abs*(rnd.nextLong()) % 9000000000L);

// Elegir tipo de cuenta aleatoriamente

String tipoCuenta = tiposCuenta[rnd.nextInt(tiposCuenta.length)];

escritor.write(String.*format*("Número de cuenta: %d%n", numCuenta));

escritor.write(String.*format*("Cliente: %s%n", cliente));

escritor.write(String.*format*("Tipo de cuenta: %s%n", tipoCuenta));

escritor.write(System.*lineSeparator*()); // línea vacía entre clientes

}

} catch (IOException e) {

System.*out*.println("Error al generar archivo " + nombreArchivo + ": " + e.getMessage());

}

}

}

**consola:**

—---------------------------------------------------------------------------

Clientes iniciales guardados:

Juan Pérez

María López

Carlos Sánchez

Ana Gómez

Luis Fernández

Sofía Martínez

Pedro Díaz

Laura Romero

Jorge Herrera

Marta Castro

Ingrese 5 nuevos clientes (nombre y apellido):

Cliente 1: leandro callizaya

Cliente 2: franco dalla via

Cliente 3: pablo cannizzaro

Cliente 4: joaquin conde

Cliente 5: juan ferro

Archivo actualizado con 5 nuevos clientes al principio:

leandro callizaya

franco dalla via

pablo cannizzaro

joaquin conde

juan ferro

Juan Pérez

María López

Carlos Sánchez

Ana Gómez

Luis Fernández

Sofía Martínez

Pedro Díaz

Laura Romero

Jorge Herrera

Marta Castro

Archivo Clientes2.txt generado con más datos:

Número de cuenta: 9323020786

Cliente: leandro callizaya

Tipo de cuenta: Caja de ahorro

Número de cuenta: 8448667746

Cliente: franco dalla via

Tipo de cuenta: Cuenta corriente

Número de cuenta: 8530232755

Cliente: pablo cannizzaro

Tipo de cuenta: Caja de ahorro

Número de cuenta: 9722129712

Cliente: joaquin conde

Tipo de cuenta: Caja de ahorro

Número de cuenta: 9145233876

Cliente: juan ferro

Tipo de cuenta: Caja de ahorro

Número de cuenta: 1036184823

Cliente: Juan Pérez

Tipo de cuenta: Caja de ahorro

Número de cuenta: 5377090771

Cliente: María López

Tipo de cuenta: Cuenta corriente

Número de cuenta: 7068255537

Cliente: Carlos Sánchez

Tipo de cuenta: Caja de ahorro

Número de cuenta: 8497125893

Cliente: Ana Gómez

Tipo de cuenta: Caja de ahorro

Número de cuenta: 9562212346

Cliente: Luis Fernández

Tipo de cuenta: Caja de ahorro

Número de cuenta: 4923980252

Cliente: Sofía Martínez

Tipo de cuenta: Cuenta corriente

Número de cuenta: 3903652585

Cliente: Pedro Díaz

Tipo de cuenta: Caja de ahorro

Número de cuenta: 4777747112

Cliente: Laura Romero

Tipo de cuenta: Caja de ahorro

Número de cuenta: 9009115951

Cliente: Jorge Herrera

Tipo de cuenta: Caja de ahorro

Número de cuenta: 1114378256

Cliente: Marta Castro

Tipo de cuenta: Caja de ahorro

—----------------------------------------------------------------------------

Ejercicio 5:

import java.io.\*;

import java.util.\*;

public class GestionDVD {

static class DVD {

String id;

String titulo;

String genero;

int año;

boolean eliminado;

public DVD(String id, String titulo, String genero, int año, boolean eliminado) {

this.id = id;

this.titulo = titulo;

this.genero = genero;

this.año = año;

this.eliminado = eliminado;

}

@Override

public String toString() {

return String.*format*("ID: %s | Título: %s | Género: %s | Año: %d | Eliminado: %s",

id, titulo, genero, año, eliminado ? "Sí" : "No");

}

public String toArchivo() {

return String.*join*("|", id, titulo, genero, String.*valueOf*(año), String.*valueOf*(eliminado));

}

}

static final String *ARCHIVO\_DVDS* = "dvds.txt";

public static void main(String[] args) {

Scanner scanner = new Scanner(System.*in*);

List<DVD> listaDVDs = *cargarDVDs*();

boolean salir = false;

while (!salir) {

*mostrarMenu*();

System.*out*.print("Seleccione opción: ");

String opcion = scanner.nextLine();

switch (opcion) {

case "1":

*altaDVD*(listaDVDs, scanner);

break;

case "2":

*bajaLogicaDVD*(listaDVDs, scanner);

break;

case "3":

*modificarDVD*(listaDVDs, scanner);

break;

case "4":

*consultaDVD*(listaDVDs, scanner);

break;

case "5":

*cantidadElementos*(listaDVDs);

break;

case "6":

*cantidadConCondicion*(listaDVDs, scanner);

break;

case "7":

*listarTodos*(listaDVDs);

break;

case "8":

*listadoCondicion*(listaDVDs, scanner);

break;

case "9":

salir = true;

*guardarDVDs*(listaDVDs);

System.*out*.println("Programa finalizado.");

break;

default:

System.*out*.println("Opción inválida, intente de nuevo.");

}

System.*out*.println();

}

scanner.close();

}

// Menú

public static void mostrarMenu() {

System.*out*.println("=== MENU ===");

System.*out*.println("1. Agregar nuevo DVD (alta)");

System.*out*.println("2. Eliminar DVD (baja lógica)");

System.*out*.println("3. Modificar DVD");

System.*out*.println("4. Consultas");

System.*out*.println("5. Cantidad total de elementos");

System.*out*.println("6. Cantidad que cumplen condición");

System.*out*.println("7. Listado de todos los DVDs");

System.*out*.println("8. Listado que cumplen condición");

System.*out*.println("9. Salir");

}

// Cargar DVDs desde archivo

public static List<DVD> cargarDVDs() {

List<DVD> lista = new ArrayList<>();

File archivo = new File(*ARCHIVO\_DVDS*);

if (!archivo.exists()) {

return lista; // Si no existe archivo, devolver lista vacía

}

try (BufferedReader br = new BufferedReader(new FileReader(archivo))) {

String linea;

while ((linea = br.readLine()) != null) {

String[] partes = linea.split("\\|");

if (partes.length == 5) {

DVD dvd = new DVD(

partes[0],

partes[1],

partes[2],

Integer.*parseInt*(partes[3]),

Boolean.*parseBoolean*(partes[4])

);

lista.add(dvd);

}

}

} catch (IOException e) {

System.*out*.println("Error al cargar DVDs: " + e.getMessage());

}

return lista;

}

// Guardar lista DVDs a archivo

public static void guardarDVDs(List<DVD> lista) {

try (BufferedWriter bw = new BufferedWriter(new FileWriter(*ARCHIVO\_DVDS*))) {

for (DVD dvd : lista) {

bw.write(dvd.toArchivo());

bw.newLine();

}

} catch (IOException e) {

System.*out*.println("Error al guardar DVDs: " + e.getMessage());

}

}

// Alta DVD

public static void altaDVD(List<DVD> lista, Scanner scanner) {

System.*out*.println("Agregar nuevo DVD:");

System.*out*.print("ID: ");

String id = scanner.nextLine().trim();

if (*buscarDVD*(lista, id) != null) {

System.*out*.println("ID ya existe. No se puede agregar.");

return;

}

System.*out*.print("Título: ");

String titulo = scanner.nextLine().trim();

System.*out*.print("Género: ");

String genero = scanner.nextLine().trim();

System.*out*.print("Año: ");

int año;

try {

año = Integer.*parseInt*(scanner.nextLine());

} catch (NumberFormatException e) {

System.*out*.println("Año inválido.");

return;

}

lista.add(new DVD(id, titulo, genero, año, false));

System.*out*.println("DVD agregado correctamente.");

}

// Baja lógica DVD

public static void bajaLogicaDVD(List<DVD> lista, Scanner scanner) {

System.*out*.print("Ingrese ID para eliminar (baja lógica): ");

String id = scanner.nextLine().trim();

DVD dvd = *buscarDVD*(lista, id);

if (dvd == null) {

System.*out*.println("DVD no encontrado.");

return;

}

if (dvd.eliminado) {

System.*out*.println("DVD ya está eliminado.");

return;

}

dvd.eliminado = true;

System.*out*.println("DVD marcado como eliminado.");

}

// Modificar DVD

public static void modificarDVD(List<DVD> lista, Scanner scanner) {

System.*out*.print("Ingrese ID para modificar: ");

String id = scanner.nextLine().trim();

DVD dvd = *buscarDVD*(lista, id);

if (dvd == null) {

System.*out*.println("DVD no encontrado.");

return;

}

System.*out*.println("Dejar vacío para no modificar.");

System.*out*.print("Nuevo título (actual: " + dvd.titulo + "): ");

String titulo = scanner.nextLine().trim();

if (!titulo.isEmpty()) {

dvd.titulo = titulo;

}

System.*out*.print("Nuevo género (actual: " + dvd.genero + "): ");

String genero = scanner.nextLine().trim();

if (!genero.isEmpty()) {

dvd.genero = genero;

}

System.*out*.print("Nuevo año (actual: " + dvd.año + "): ");

String añoStr = scanner.nextLine().trim();

if (!añoStr.isEmpty()) {

try {

dvd.año = Integer.*parseInt*(añoStr);

} catch (NumberFormatException e) {

System.*out*.println("Año inválido, no se modificó.");

}

}

System.*out*.println("DVD modificado.");

}

// Buscar DVD por ID

public static DVD buscarDVD(List<DVD> lista, String id) {

for (DVD dvd : lista) {

if (dvd.id.equals(id)) {

return dvd;

}

}

return null;

}

// Consultas

public static void consultaDVD(List<DVD> lista, Scanner scanner) {

System.*out*.print("Ingrese ID para buscar: ");

String id = scanner.nextLine().trim();

DVD dvd = *buscarDVD*(lista, id);

if (dvd == null) {

System.*out*.println("DVD no encontrado.");

return;

}

System.*out*.println(dvd);

}

// Cantidad total elementos (no eliminados)

public static void cantidadElementos(List<DVD> lista) {

long count = lista.stream().filter(dvd -> !dvd.eliminado).count();

System.*out*.println("Cantidad total de DVDs activos: " + count);

}

// Cantidad que cumplen condición (por género o año mínimo)

public static void cantidadConCondicion(List<DVD> lista, Scanner scanner) {

System.*out*.println("Ingrese condición para contar:");

System.*out*.println("1. Género");

System.*out*.println("2. Año mínimo");

System.*out*.print("Opción: ");

String opcion = scanner.nextLine().trim();

long count = 0;

switch (opcion) {

case "1":

System.*out*.print("Ingrese género: ");

String genero = scanner.nextLine().trim();

count = lista.stream()

.filter(dvd -> !dvd.eliminado && dvd.genero.equalsIgnoreCase(genero))

.count();

break;

case "2":

System.*out*.print("Ingrese año mínimo: ");

try {

int añoMin = Integer.*parseInt*(scanner.nextLine());

count = lista.stream()

.filter(dvd -> !dvd.eliminado && dvd.año >= añoMin)

.count();

} catch (NumberFormatException e) {

System.*out*.println("Año inválido.");

return;

}

break;

default:

System.*out*.println("Opción inválida.");

return;

}

System.*out*.println("Cantidad que cumplen condición: " + count);

}

// Listado todos (no eliminados)

public static void listarTodos(List<DVD> lista) {

System.*out*.println("Listado completo de DVDs activos:");

lista.stream()

.filter(dvd -> !dvd.eliminado)

.forEach(System.*out*::println);

}

// Listado con condición

public static void listadoCondicion(List<DVD> lista, Scanner scanner) {

System.*out*.println("Ingrese condición para listado:");

System.*out*.println("1. Género");

System.*out*.println("2. Año mínimo");

System.*out*.print("Opción: ");

String opcion = scanner.nextLine().trim();

List<DVD> filtrados = new ArrayList<>();

switch (opcion) {

case "1":

System.*out*.print("Ingrese género: ");

String genero = scanner.nextLine().trim();

filtrados = lista.stream()

.filter(dvd -> !dvd.eliminado && dvd.genero.equalsIgnoreCase(genero))

.toList();

break;

case "2":

System.*out*.print("Ingrese año mínimo: ");

try {

int añoMin = Integer.*parseInt*(scanner.nextLine());

filtrados = lista.stream()

.filter(dvd -> !dvd.eliminado && dvd.año >= añoMin)

.toList();

} catch (NumberFormatException e) {

System.*out*.println("Año inválido.");

return;

}

break;

default:

System.*out*.println("Opción inválida.");

return;

}

if (filtrados.isEmpty()) {

System.*out*.println("No hay DVDs que cumplan la condición.");

} else {

System.*out*.println("DVDs que cumplen la condición:");

filtrados.forEach(System.*out*::println);

}

}

}

consola:

=== MENU ===

1. Agregar nuevo DVD (alta)

2. Eliminar DVD (baja lógica)

3. Modificar DVD

4. Consultas

5. Cantidad total de elementos

6. Cantidad que cumplen condición

7. Listado de todos los DVDs

8. Listado que cumplen condición

9. Salir

Seleccione opción: 1

Agregar nuevo DVD:

ID: 12345

Título: los increibles

Género: accion

Año: 2002

DVD agregado correctamente.

=== MENU ===

1. Agregar nuevo DVD (alta)

2. Eliminar DVD (baja lógica)

3. Modificar DVD

4. Consultas

5. Cantidad total de elementos

6. Cantidad que cumplen condición

7. Listado de todos los DVDs

8. Listado que cumplen condición

9. Salir

Seleccione opción: 7

Listado completo de DVDs activos:

ID: 12345 | Título: los increibles | Género: accion | Año: 2002 | Eliminado: No

ejercicio 6:

import java.io.\*;

import java.util.\*;

public class GestionElementos {

static class Elemento {

String clave;

String nombre;

String descripcion;

boolean eliminado;

public Elemento(String clave, String nombre, String descripcion, boolean eliminado) {

this.clave = clave;

this.nombre = nombre;

this.descripcion = descripcion;

this.eliminado = eliminado;

}

@Override

public String toString() {

return String.*format*("Clave: %s | Nombre: %s | Descripción: %s | Eliminado: %s",

clave, nombre, descripcion, eliminado ? "Sí" : "No");

}

public String toArchivo() {

return String.*join*("|", clave, nombre, descripcion, String.*valueOf*(eliminado));

}

}

static final String *ARCHIVO* = "elementos.txt";

public static void main(String[] args) {

Scanner scanner = new Scanner(System.*in*);

List<Elemento> lista = *cargarElementos*();

boolean salir = false;

while (!salir) {

*mostrarMenu*();

System.*out*.print("Seleccione opción: ");

String opcion = scanner.nextLine();

switch (opcion) {

case "1":

*altaElemento*(lista, scanner);

break;

case "2":

*bajaLogicaElemento*(lista, scanner);

break;

case "3":

*modificarElemento*(lista, scanner);

break;

case "4":

*consultaElemento*(lista, scanner);

break;

case "5":

*cantidadElementos*(lista);

break;

case "6":

*cantidadCondicion*(lista, scanner);

break;

case "7":

*listadoTodos*(lista);

break;

case "8":

*listadoCondicion*(lista, scanner);

break;

case "9":

salir = true;

*guardarElementos*(lista);

System.*out*.println("Programa finalizado.");

break;

default:

System.*out*.println("Opción inválida, intente de nuevo.");

}

System.*out*.println();

}

scanner.close();

}

public static void mostrarMenu() {

System.*out*.println("=== MENU ===");

System.*out*.println("1. Agregar nuevo elemento (alta)");

System.*out*.println("2. Eliminar elemento (baja lógica)");

System.*out*.println("3. Modificar elemento");

System.*out*.println("4. Consultas (buscar por clave)");

System.*out*.println("5. Cantidad total de elementos activos");

System.*out*.println("6. Cantidad que cumplen condición");

System.*out*.println("7. Listado de todos los elementos activos");

System.*out*.println("8. Listado de elementos que cumplen condición");

System.*out*.println("9. Salir");

}

public static List<Elemento> cargarElementos() {

List<Elemento> lista = new ArrayList<>();

File archivo = new File(*ARCHIVO*);

if (!archivo.exists()) return lista;

try (BufferedReader br = new BufferedReader(new FileReader(archivo))) {

String linea;

while ((linea = br.readLine()) != null) {

String[] partes = linea.split("\\|");

if (partes.length == 4) {

lista.add(new Elemento(

partes[0],

partes[1],

partes[2],

Boolean.*parseBoolean*(partes[3])

));

}

}

} catch (IOException e) {

System.*out*.println("Error al cargar elementos: " + e.getMessage());

}

return lista;

}

public static void guardarElementos(List<Elemento> lista) {

try (BufferedWriter bw = new BufferedWriter(new FileWriter(*ARCHIVO*))) {

for (Elemento e : lista) {

bw.write(e.toArchivo());

bw.newLine();

}

} catch (IOException e) {

System.*out*.println("Error al guardar elementos: " + e.getMessage());

}

}

public static Elemento buscarPorClave(List<Elemento> lista, String clave) {

for (Elemento e : lista) {

if (e.clave.equals(clave)) return e;

}

return null;

}

public static void altaElemento(List<Elemento> lista, Scanner scanner) {

System.*out*.print("Clave: ");

String clave = scanner.nextLine().trim();

if (*buscarPorClave*(lista, clave) != null) {

System.*out*.println("Clave ya existe.");

return;

}

System.*out*.print("Nombre: ");

String nombre = scanner.nextLine().trim();

System.*out*.print("Descripción: ");

String descripcion = scanner.nextLine().trim();

lista.add(new Elemento(clave, nombre, descripcion, false));

System.*out*.println("Elemento agregado.");

}

public static void bajaLogicaElemento(List<Elemento> lista, Scanner scanner) {

System.*out*.print("Ingrese clave para eliminar (baja lógica): ");

String clave = scanner.nextLine().trim();

Elemento e = *buscarPorClave*(lista, clave);

if (e == null) {

System.*out*.println("Elemento no encontrado.");

return;

}

if (e.eliminado) {

System.*out*.println("Elemento ya está eliminado.");

return;

}

e.eliminado = true;

System.*out*.println("Elemento marcado como eliminado.");

}

public static void modificarElemento(List<Elemento> lista, Scanner scanner) {

System.*out*.print("Ingrese clave para modificar: ");

String clave = scanner.nextLine().trim();

Elemento e = *buscarPorClave*(lista, clave);

if (e == null) {

System.*out*.println("Elemento no encontrado.");

return;

}

System.*out*.println("Dejar vacío para no modificar.");

System.*out*.print("Nuevo nombre (actual: " + e.nombre + "): ");

String nombre = scanner.nextLine().trim();

if (!nombre.isEmpty()) e.nombre = nombre;

System.*out*.print("Nueva descripción (actual: " + e.descripcion + "): ");

String descripcion = scanner.nextLine().trim();

if (!descripcion.isEmpty()) e.descripcion = descripcion;

System.*out*.println("Elemento modificado.");

}

public static void consultaElemento(List<Elemento> lista, Scanner scanner) {

System.*out*.print("Ingrese clave para buscar: ");

String clave = scanner.nextLine().trim();

Elemento e = *buscarPorClave*(lista, clave);

if (e == null) {

System.*out*.println("Elemento no encontrado.");

return;

}

System.*out*.println(e);

}

public static void cantidadElementos(List<Elemento> lista) {

long count = lista.stream().filter(e -> !e.eliminado).count();

System.*out*.println("Cantidad total de elementos activos: " + count);

}

public static void cantidadCondicion(List<Elemento> lista, Scanner scanner) {

System.*out*.println("Ingrese condición para contar:");

System.*out*.println("1. Nombre contiene");

System.*out*.println("2. Descripción contiene");

System.*out*.print("Opción: ");

String opcion = scanner.nextLine().trim();

long count = 0;

switch (opcion) {

case "1":

System.*out*.print("Ingrese texto para nombre: ");

String textoNombre = scanner.nextLine().trim().toLowerCase();

count = lista.stream()

.filter(e -> !e.eliminado && e.nombre.toLowerCase().contains(textoNombre))

.count();

break;

case "2":

System.*out*.print("Ingrese texto para descripción: ");

String textoDesc = scanner.nextLine().trim().toLowerCase();

count = lista.stream()

.filter(e -> !e.eliminado && e.descripcion.toLowerCase().contains(textoDesc))

.count();

break;

default:

System.*out*.println("Opción inválida.");

return;

}

System.*out*.println("Cantidad que cumplen condición: " + count);

}

public static void listadoTodos(List<Elemento> lista) {

System.*out*.println("Listado de todos los elementos activos:");

lista.stream()

.filter(e -> !e.eliminado)

.forEach(System.*out*::println);

}

public static void listadoCondicion(List<Elemento> lista, Scanner scanner) {

System.*out*.println("Ingrese condición para listado:");

System.*out*.println("1. Nombre contiene");

System.*out*.println("2. Descripción contiene");

System.*out*.print("Opción: ");

String opcion = scanner.nextLine().trim();

List<Elemento> filtrados = new ArrayList<>();

switch (opcion) {

case "1":

System.*out*.print("Ingrese texto para nombre: ");

String textoNombre = scanner.nextLine().trim().toLowerCase();

filtrados = lista.stream()

.filter(e -> !e.eliminado && e.nombre.toLowerCase().contains(textoNombre))

.toList();

break;

case "2":

System.*out*.print("Ingrese texto para descripción: ");

String textoDesc = scanner.nextLine().trim().toLowerCase();

filtrados = lista.stream()

.filter(e -> !e.eliminado && e.descripcion.toLowerCase().contains(textoDesc))

.toList();

break;

default:

System.*out*.println("Opción inválida.");

return;

}

if (filtrados.isEmpty()) {

System.*out*.println("No hay elementos que cumplan la condición.");

} else {

System.*out*.println("Elementos que cumplen la condición:");

filtrados.forEach(System.*out*::println);

}

}

}

**consola**:

—------------------------------------------------------------------------------------------------------------------------

=== MENU ===

1. Agregar nuevo elemento (alta)

2. Eliminar elemento (baja lógica)

3. Modificar elemento

4. Consultas (buscar por clave)

5. Cantidad total de elementos activos

6. Cantidad que cumplen condición

7. Listado de todos los elementos activos

8. Listado de elementos que cumplen condición

9. Salir

Seleccione opción: 1

Clave: 1234

Nombre: tierra

Descripción: polvo

Elemento agregado.

=== MENU ===

1. Agregar nuevo elemento (alta)

2. Eliminar elemento (baja lógica)

3. Modificar elemento

4. Consultas (buscar por clave)

5. Cantidad total de elementos activos

6. Cantidad que cumplen condición

7. Listado de todos los elementos activos

8. Listado de elementos que cumplen condición

9. Salir

Seleccione opción: 7

Listado de todos los elementos activos:

Clave: 1234 | Nombre: tierra | Descripción: polvo | Eliminado: No