





















Solución propuesta para el desafio Reservas de biblioteca

Estructura del proyecto

- ▼  src
 - ▼  dao
 - ▶  XlsHandler.java
 - ▼  main
 - ▶  Ejecutable.java
 - ▼  modelo
 - ▶   Artículo.java
 - ▶  HandlerResponse.java
 - ▶  Libro.java
 - ▶  Pelicula.java
 - ▶  Usuario.java
 - ▼  servicio
 - ▶  SessionHandler.java
 - ▶  Singleton.java
 - ▼  utilidades
 - ▶   Escaner.java
-  articulos.xls
-  usuarios.xls

Las Clases

La clase Artículo

Contiene este constructor y atributos encapsulados:

```
public abstract class Artículo {

    protected int plaxoMaximo;
    protected String nombre;
    protected String codigo;
    protected boolean reservado;

    public Artículo(int plaxoMaximo, String nombre, String codigo, boolean reservado) {
        super();
        this.plaxoMaximo = plaxoMaximo;
        this.nombre = nombre;
        this.codigo = codigo;
        this.reservado = reservado;
    }
    //Encapsulamiento
}
```

La clase Libro

Extiende de Artículo y tiene páginas e imprenta.

```
public class Libro extends Artículo {

    private int paginas;
    private String imprenta;

    public Libro(int plaxoMaximo, String nombre, String codigo, int paginas, String imprenta, boolean reservado) {
        super(plaxoMaximo, nombre, codigo, reservado);
        this.paginas = paginas;
        this.imprenta = imprenta;
    }
    // Encapsulamiento
    @Override
    public String toString() {
        return "Libro [paginas=" + paginas + ", imprenta=" + imprenta + ", plaxoMaximo=" + plaxoMaximo + ", nombre=" + nombre + ", codigo=" + codigo + ", reservado=" + reservado + "];"
    }
}
```

La clase Película

Extiende de Artículo y tiene duración y calidad:

```
public class Pelicula extends Artículo {

    private int duracion;
    private String calidad;

    public Pelicula(int plaxoMaximo, String nombre, String codigo, int
duracion, String calidad, boolean reservado) {
        super(plaxoMaximo, nombre, codigo, reservado);
        this.duracion = duracion;
        this.calidad = calidad;
    }
    // Encapsulamiento
    @Override
    public String toString() {
        return "Pelicula [duracion=" + duracion + ", calidad=" + calidad + ",
plaxoMaximo=" + plaxoMaximo + ", nombre="
            + nombre + ", codigo=" + codigo + ", reservado=" + reservado +
        "]" ;
    }
}
```

La clase Usuario

Se utiliza para mantener activo al usuario:

```
public class Usuario{

    private String nombre;
    private String clave;

    public Usuario(String nombre, String clave) {
        super();
        this.nombre = nombre;
        this.clave = clave;
    }
    //Encapsulamiento
}
```

La clase XlsHandler

Contiene la lógica para leer y escribir los archivos XLS, el método `guardarArticulos()` lee la lista de artículos del singleton y sobrescribe el archivo con los datos de esa lista.

El método `leerArticulos()` obtiene los datos desde `articulos.xls`. El método `leerUsuarios()` obtiene los datos desde `usuarios.xls`

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.util.ArrayList;
import java.util.List;

import org.apache.poi.hssf.usermodel.HSSFCell;
import org.apache.poi.hssf.usermodel.HSSFRow;
import org.apache.poi.hssf.usermodel.HSSFSheet;
import org.apache.poi.hssf.usermodel.HSSFWorkbook;
import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.ss.usermodel.Sheet;
import org.apache.poi.ss.usermodel.Workbook;
import org.apache.poi.ss.usermodel.WorkbookFactory;

import modelo.Articulo;
import modelo.HandlerResponse;
import modelo.Libro;
import modelo.Pelicula;
import modelo.Usuario;

public class XlsHandler {

    String excelFilePathArticulos = ".\\articulos.xls";
    String excelFilePathUsuarios = ".\\usuarios.xls";

    public HandlerResponse guardarArticulos(List<Articulo> lista) {

        HSSFWorkbook libro = new HSSFWorkbook();
        HSSFSheet hoja = libro.createSheet();
        int tipoArticulo = -1;
        int valorCeldaCuatro = 0;
        String valorCeldaCinco = "";
        HSSFCell celda = null;
        for (int i = 0; i < lista.size(); i++) {
            if (lista.get(i).getClass().equals(Libro.class)) {
                tipoArticulo = 0;
                valorCeldaCuatro = ((Libro) lista.get(i)).getPaginas();
                valorCeldaCinco = ((Libro) lista.get(i)).getImprenta();
            } else {
                tipoArticulo = 1;
                valorCeldaCuatro = ((Pelicula) lista.get(i)).getDuracion();
                valorCeldaCinco = ((Pelicula) lista.get(i)).getCalidad();
            }

            HSSFRow fila = hoja.createRow(i);
```

[illegible]

```

        row.getCell(3).getStringCellValue(),
        (int) row.getCell(4).getNumericCellValue(),
        row.getCell(5).getStringCellValue(),
        row.getCell(6).getBooleanCellValue());
    } else {
        articulo = new Pelicula((int)
row.getCell(1).getNumericCellValue(),
        row.getCell(2).getStringCellValue(),
row.getCell(3).getStringCellValue(),
        (int) row.getCell(4).getNumericCellValue(),
row.getCell(5).getStringCellValue(),
        row.getCell(6).getBooleanCellValue());
    }
    lista.add(articulo);
} else {
    break;
}
}
} catch (Exception ioe) {
    ioe.printStackTrace();
    return new ArrayList<Articulo>();
}
try {
    workbook.close();
} catch (Exception e) {
}
try {
    inputStream.close();
} catch (Exception e) {
}
return lista;
}

public List<Usuario> leerUsuarios() {
    Workbook workbook = null;
    FileInputStream inputStream = null;
    List<Usuario> lista = new ArrayList<>();
    try {
        inputStream = new FileInputStream(new
File(excelFilePathUsuarios));
        workbook = WorkbookFactory.create(inputStream);

        Sheet sheet = workbook.getSheetAt(0);
        Row row;

        int cantidadFilas = sheet.getPhysicalNumberOfRows();

        // Leer filas completas
        Usuario user = null;
        for (int r = 0; r < cantidadFilas; r++) {
            row = sheet.getRow(r);
            if (row != null) {

                user = new Usuario(row.getCell(0).getStringCellValue(),
row.getCell(1).getStringCellValue());

                lista.add(user);
            }

```

```

        }
    } catch (Exception ioe) {
        ioe.printStackTrace();
        return new ArrayList<Usuario>();
    }
    try {
        workbook.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
    try {
        inputStream.close();
    } catch (Exception e) {
    }
    return lista;
}
}

```

La clase HandlerResponse

Es utilizado para que el XLSHandler pueda responder con un mensaje y un resultado de la operación, donde true indica un resultado correcto.

```

public class HandlerResponse {

    String mensaje;
    Boolean resultado;

    public HandlerResponse(String mensaje, Boolean resultado) {
        super();
        this.mensaje = mensaje;
        this.resultado = resultado;
    }
    // Encapsulamiento
}

```

La clase SessionHandler

Contiene la información del usuario activo en una variable estática

```
import modelo.Usuario;

public class SessionHandler {

    private static Usuario usuarioActivo = null;

    public static Usuario getUsuarioActivo() {
        return usuarioActivo;
    }

    public static void setUsuarioActivo(Usuario usuarioActivo) {
        SessionHandler.usuarioActivo = usuarioActivo;
    }

}
```


El Singleton

Mantiene Artículos y Usuarios.

```
import java.util.List;

import dao.XlsHandler;
import modelo.Articulo;
import modelo.Usuario;

public class Singleton {

    //Variable encapsulada y estática donde se almacenará la instancia.
    private static Singleton singleton;
    private List<Articulo> articulos;
    private List<Usuario> usuarios;

    public static Singleton getInstance() {
        if (singleton == null) {
            synchronized(Singleton.class) {
                if (singleton == null) {
                    singleton = new Singleton();
                }
            }
        }
        return singleton;
    }

    //Constructor privado
    private Singleton() {}

    public static synchronized List<Articulo> getArticulos() {
        return getInstance().articulos;
    }

    public static synchronized void setArticulos(List<Articulo> articulos) {
        getInstance().articulos = articulos;
    }

    public static List<Usuario> getUsuarios() {
        return getInstance().usuarios;
    }

    public static void setUsuarios(List<Usuario> usuarios) {
        getInstance().usuarios = usuarios;
    }

    public static synchronized boolean guardarCambios() {
        XlsHandler hdlr = new XlsHandler();
        return hdlr.guardarArticulos(getInstance().articulos).getResultado();
    }

}
```

La clase Escaner

Contiene métodos que se encargan de escanear correctamente entradas de datos del usuario, manejando excepciones y enviando mensajes de error y de los datos que se pidan, recibidos por parámetros.

```
import java.util.Scanner;

public abstract class Escaner {

    static Scanner sc = new Scanner(System.in);

    public static Integer leerNumeroConMensaje(int maximo, int minimo, String
mensaje) {
        System.out.println(mensaje);
        int valor = 0;
        while(valor == 0) {
            try {
                valor = Integer.parseInt(sc.nextLine());
                if(valor <= maximo && valor >= minimo) {
                    return valor;
                }else {
                    valor = -1;
                }
            }catch(Exception e) {
            }finally{
                if(valor == -1) {
                    System.out.println("Ingrese un numero valido");
                }
                valor = 0;
            }
        }
        return null;
    }

    public static String leerStringConMensaje(String mensaje) {
        System.out.println(mensaje);
        try {
            return sc.nextLine();
        }catch(Exception e) {
            System.out.println("Ingrese un valor valido");
        }
        return null;
    }

    public static String leerRespuestaStringConMensaje(String mensaje) {
        System.out.println(mensaje);
        int valor = 0;
        while(valor == 0) {
            System.out.println("Ingrese Y para si o N para no");
            try {
                String codigo = sc.nextLine();
                if(codigo.equals("Y") || codigo.equals("N")) {
                    return codigo;
                }
            }catch(Exception e) {
            }finally{
                valor = 0;
            }
        }
        return null;
    }
}
```

```

        valor = 0;
    }
}
return null;
}
}

```

La clase Ejecutable

Excluyendo el método main, contiene cinco métodos y tres atributos estáticos:

```

public static XlsHandler hdlr = new XlsHandler();

private static void cargarDatos() {
    Singleton.setArticulos(hdlr.leerArticulos());
    Singleton.setUsuarios(hdlr.leerUsuarios());
}

private static void buscarArticulos(String nombre) {
    System.out.println("Buscando articulos con "+nombre+" en su nombre");

    for (Articulo articulo : Singleton.getArticulos()) {
        if (articulo.getNombre().contains(nombre)) {
            System.out.println(articulo);
        }
    }
    System.out.println("Buqueda finalizada.");
}

private static void generarReserva(Integer indice) {
    if(!Singleton.getArticulos().get(indice).isReservado()) {
        Singleton.getArticulos().get(indice).setReservado(true);
        Singleton.guardarCambios();
        System.out.println("Reserva realizada.");
    }else {

        System.out.println("El articulo no se encuentra disponible.");
    }

}

private static Integer buscarIndiceArticuloPorCodigo(String
codigoArticulo) {
    List<Articulo> cloneList = Singleton.getArticulos();
    for (int i = 0; i < cloneList.size(); i++) {
        if (cloneList.get(i).getCodigo().equals(codigoArticulo)) {
            return i;
        }
    }
    return null;
}

```

```

        private static Usuario buscarUsuario(String nombre, String clave) {
            List<Usuario> cloneList = Singleton.getUsuarios();
            for (int i = 0; i < cloneList.size(); i++) {
                if (cloneList.get(i).getClave().equals(clave) &&
cloneList.get(i).getNombre().equals(nombre)) {
                    return cloneList.get(i);
                }
            }
            return null;
        }
    }
}

```

cargarDatos() guarda los datos leídos desde articulos.xls y usuarios.xls en el Singleton al iniciar la aplicación. buscarArticulos(String nombre) busca un substring dentro del nombre de todos los artículos e imprime el artículo donde se encuentre el substring. generarReserva(Integer indice) recibe el índice de un artículo de la lista del Singleton para generar la reserva buscarIndiceArticuloPorCodigo(String codigoArticulo) recibe el código de un artículo y busca en la lista del Singleton un artículo con el código para devolver el índice del mismo dentro del arreglo del Singleton. buscarUsuario(String nombre, String clave) busca un usuario en la lista del singleton que tenga el nombre y clave ingresados.

```

public static void main(String[] args) {
    cargarDatos();
    int valor = 100;

    while(SessionHandler.getUsuarioActivo() == null) {
        String nombre = Escaner.leerStringConMensaje("Ingrese su nombre de
usuario.");
        String clave = Escaner.leerStringConMensaje("Ingrese su clave.");
        SessionHandler.setUsuarioActivo(buscarUsuario(nombre, clave));
    }
    System.out.println("Bienvenido
"+SessionHandler.getUsuarioActivo().getNombre());
    while (valor > 0 && SessionHandler.getUsuarioActivo() != null) {
        valor = Escaner.leerNumeroConMensaje(2, 0,
            "--- \n¿Qué desea hacer?\n1- Reservar un articulo\n2-
Buscar articulo\n0- Salir");
        if (valor == 1) {
            Integer indice = null;
            while (indice == null) {
                indice =
buscarIndiceArticuloPorCodigo(Escaner.leerStringConMensaje("Ingrese el código
del articulo.));
                if(indice == null)
                    System.out.println("No encontrado, intente
nuevamente.");
            }
            generarReserva(indice);

        } else if (valor == 2) {
            buscarArticulos(Escaner.leerStringConMensaje("Ingrese un
nombre para buscar en la lista.));
        } else if (valor == 0) {
            System.out.println("Hasta pronto
"+SessionHandler.getUsuarioActivo().getNombre()+".");
        }
    }
}

```

}

Resultado por consola de una interacción completa:

```
Ingrese su nombre de usuario.
a
Ingrese su clave.
a
Bienvenido a
---
¿Qué desea hacer?
1- Reservar un articulo
2- Buscar articulo
0- Salir
1
Ingrese el código del articulo.
a1234
El articulo no se encuentra disponible.
---
¿Qué desea hacer?
1- Reservar un articulo
2- Buscar articulo
0- Salir
2
Ingrese un nombre para buscar en la lista.
pr
Buscando articulos con pr en su nombre
Libro [paginas=30, imprenta=imprenta, plaxoMaximo=2, nombre=pruebita,
codigo=a1234, reservado=true]
Pelicula [duracion=45, calidad=4k, plaxoMaximo=3, nombre=pruebaza,
codigo=a5678, reservado=false]
Buqueda finalizada.
---
¿Qué desea hacer?
1- Reservar un articulo
2- Buscar articulo
0- Salir
1
Ingrese el código del articulo.
a5678
Reserva realizada.
---
¿Qué desea hacer?
1- Reservar un articulo
2- Buscar articulo
0- Salir
2
Ingrese un nombre para buscar en la lista.
p
Buscando articulos con p en su nombre
Libro [paginas=30, imprenta=imprenta, plaxoMaximo=2, nombre=pruebita,
codigo=a1234, reservado=true]
```

Pelicula [duracion=45, calidad=4k, plaxoMaximo=3, nombre=pruebaza,
codigo=a5678, reservado=true]

Buqueda finalizada.

¿Qué desea hacer?

1- Reservar un articulo

2- Buscar articulo

0- Salir

0

Hasta pronto a.