UNIVERSIDAD SIGLO 21

TRABAJO PRÁCTICO

Estudiante: Daiana Micaela Arena

Cursada: EDH

DNI: 38629115

Descripción de la situación problemática (contexto/escenario):

Cozy Books es una tienda en línea que utiliza el modelo de negocio de dropshipping para vender libros. Este modelo de negocio implica que la empresa **no tiene un inventario físico de libros**, en su lugar, trabaja con proveedores que envían los libros directamente a los clientes.

El surgimiento de Cozy Books puede atribuirse a la creciente demanda de compras en línea y la popularidad de los libros electrónicos y los dispositivos de lectura electrónica. La empresa se beneficia de la comodidad que brinda la compra en línea, al mismo tiempo que ofrece una amplia variedad de títulos y géneros a precios asequibles.

Además, el modelo de negocio de dropshipping permite a Cozy Books reducir los costos operativos y evitar la necesidad de invertir en un gran inventario de libros. Esto le permite ofrecer precios competitivos y mejorar su rentabilidad.

En resumen, Cozy Books es una empresa de comercio electrónico que utiliza el modelo de negocio de dropshipping para vender libros. Su éxito se debe a la popularidad de las compras en línea y a la eficiencia que ofrece el modelo de dropshipping para reducir los costos y mejorar la rentabilidad.

Requerimientos por resolver en la situación problemática y descripción de las funcionalidades:

El dueño de “Cozy Books” necesita un programa para gestionar su lista de libros. Para resolver esto, el programa deberá:

- Registrar Autor: esta funcionalidad permitirá al usuario ingresar los datos de un autor y agregarlo al listado de autores disponibles de la tienda.
- Mostrar Autores: permitirá al usuario ver el listado de todos los autores que se encuentran disponibles de la tienda, con sus respectivos datos.

- Registrar Cliente: esta funcionalidad permitirá al usuario ingresar los datos y agregar a la tienda a los clientes de la misma.

- Mostrar Clientes: permite ver el listado de todos los clientes y sus compras realizadas.

- Registrar un libro: esta funcionalidad permitirá al usuario ingresar los datos de un libro (físico, digital o audio) y agregarlo al listado de libros correspondiente a su tipo disponibles de la tienda.
- Mostrar el listado de libros disponibles: permitirá al usuario ver el listado de todos los libros que se encuentran disponibles de la tienda, con sus respectivos datos y ordenados por categoría (físico, digital o audio).
- Buscar un libro por autor: permitirá al usuario ingresar el nombre de un autor y ver los libros que coinciden con ese autor en el listado de la tienda.
- Registrar una venta: esta funcionalidad permitirá al usuario ingresar los datos de una venta. Dentro de la misma, se asocia 1 o más libros. Pueden ser de distintas categorías. Además, crea un documento .txt con el ticket final del cliente. Estos tickets tienen numeración automática, consecutiva y ascendente (1, 2, 3..). Los mismos se guardan en el directorio base.
- Mostrar ventas: permitirá al usuario ver el listado de todas las ventas que se realizaron en la tienda, con sus respectivos datos. Además, mostrará el total facturado.
- La clase Shop: representa al negocio que vende los libros.

Funcionalidades agregadas en este trabajo:

- Polimorfismo: En la clase Libro tenemos el método mostrarLibro() para imprimir los datos. En las clases hijas (audiolibro, digital y físico) escribimos el mismo método para adaptarlo a los nuevos datos que agrega la subclase. Por ejemplo, en audiolibro, agregamos duración, plataforma y narrador:

```java
public void mostrarLibro() {

    System.out.println("Titulo: " + this.getTitulo() );
    System.out.println("Autor: " + this.getAutor().getNombre() );
    System.out.println("Editorial: " + this.getEditorial());
    System.out.println("Año: " + this.getAño());
    System.out.println("Precio: " + this.getPrecio());
    System.out.println("Duracion: " + this.getDuracion() + " minutos");
    System.out.println("Plataforma: " + this.getPlataforma());
    System.out.println("Narrador: " + this.getNarrador());


}
```

- Manejo de excepciones: Validación cuando se solicita el ingreso de DNI del cliente por teclado al usuario. Se revisa que sean alfanuméricos y que tenga 8 caracteres:

```java
String documento="";

//Valida que el DNI sea un numero entero de 8 cifras
boolean documentoValido = false;
while (!documentoValido) {
    System.out.println(x:"Ingrese el documento del nuevo cliente (8 dígitos):");
    documento = lector.nextLine();
    try {

        //Chequea que los dni tengan 8 digitos alfanumericos
        if (documento.length() == 8) {
            documentoValido = true;
        } else {
            System.out.println(x:"El documento debe tener exactamente 8 dígitos.");

            // Genera una excepcion
            throw new NumberFormatException();
        }
    } catch (NumberFormatException e) {
        System.out.println(x:"El documento es inválido.");
    }
}
```

También en el método escribirArchivo() se valida que no ocurran errores con la creación de los tickets:

```
try {
    int num = 1;
    String numTicket = "ticket"+ num + ".txt";
    File ticket = new File(numTicket);
    while(ticket.exists()) {
        numTicket = "ticket"+ (num++) + ".txt";
        ticket = new File(numTicket);
    }


    if (ticket.createNewFile()) {···
    } else {
        System.out.println(x:"File already exists.");
    }
} catch (IOException e) {
    System.out.println(x:"An error occurred.");
    e.printStackTrace();
}
```

- Interfaces:La interface IArchivo implementada en Archivo determina el método para escribir archivos.

```
package main.java.com.app.interfaceArchivo;

import main.java.com.app.venta.Venta;

public interface IArchivo {

    public void escribirArchivo(Venta factura);
}
```

- Manipulación de archivos en Java: Se utiliza la manipulación de archivos para crear tickets numerados de forma automática, consecutiva y ascendente con cada venta registrada en el sistema. Si bien las ventas no se registran en el sistema una vez levantado nuevamente el programa, al iniciar desde cero los nuevos tickets generados siguen la numeración inicial. Esto es para que “cerrar el programa” implique un nuevo ciclo fiscal de facturación en el sistema pero no en el historial de tickets.

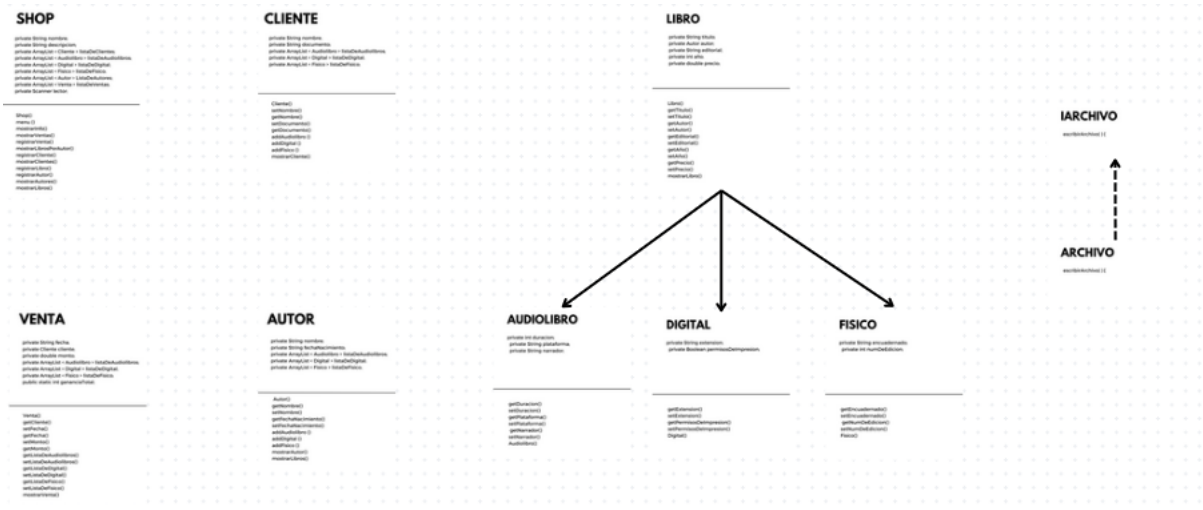
```java
public void escribirArchivo(Venta factura) {



    //factura.mostrarVenta();
    try {
        int num = 1;
        String numTicket = "ticket"+ num + ".txt";
        File ticket = new File(numTicket);
        while(ticket.exists()) {
            numTicket = "ticket"+ (num++) + ".txt";
            ticket = new File(numTicket);
        }


        if (ticket.createNewFile()) {···
        } else {
            System.out.println(x:"File already exists.");
        }
    } catch (IOException e) {
        System.out.println(x:"An error occurred.");
        e.printStackTrace();
    }
}
```

Lista de clases utilizadas en el trabajo:



(adjunto también cada clase por separado para que sea más legible)

LIBRO

private String titulo;
private Autor autor;
private String editorial;
private int año;
private double precio;

Libro()
getTitulo()
setTitulo()
getAutor()
setAutor()
getEditorial()
setEditorial()
getAño()
setAño()
getPrecio()
setPrecio()
mostrarLibro()

AUDIOLIBRO

private int duracion;
private String plataforma;
private String narrador;

---

getDuracion()
setDuracion()
getPlataforma()
setPlataforma()
getNarrador()
setNarrador()
Audiolibro()

DIGITAL

private String extension;
private Boolean permisosDeImpresion;

---

getExtension()
setExtension()
getPermisosDeImpresion()
setPermisosDeImpresion()
Digital()

FISICO

private String encuadernado;
private int numDeEdicion;

---

getEncuadernado()
setEncuadernado()
getNumDeEdicion()
setNumDeEdicion()
Fisico()

SHOP

private String nombre;
private String descripcion;
private ArrayList < Cliente > listaDeClientes;
private ArrayList < Audiolibro > listaDeAudiolibros;
private ArrayList < Digital > listaDeDigital;
private ArrayList < Fisico > listaDeFisico;
private ArrayList < Autor > ListaDeAutores;
private ArrayList < Venta > listaDeVentas;
private Scanner lector;

Shop()
menu ()
mostrarInfo()
mostrarVentas()
registrarVenta()
mostrarLibrosPorAutor()
registrarCliente()
mostrarClientes()
registrarLibro()
registrarAutor()
mostrarAutores()
mostrarLibros()

VENTA

private String fecha;
private Cliente cliente;
private double monto;
private ArrayList < Audiolibro > listaDeAudiolibros;
private ArrayList < Digital > listaDeDigital;
private ArrayList < Fisico > listaDeFisico;
public static int gananciaTotal;

Venta()
getCliente()
setFecha()
getFecha()
setMonto()
getMonto()
getListaDeAudiolibros()
setListaDeAudiolibros()
getListaDeDigital()
setListaDeDigital()
getListaDeFisico()
setListaDeFisico()
mostrarVenta()

CLIENTE

private String nombre;
private String documento;
private ArrayList < Audiolibro > listaDeAudiolibros;
private ArrayList < Digital > listaDeDigital;
private ArrayList < Fisico > listaDeFisico;

Cliente()
setNombre()
getNombre()
setDocumento()
getDocumento()
addAudiolibro ()
addDigital ()
addFisico ()
mostrarCliente()

AUTOR

private String nombre;
private String fechaNacimiento;
private ArrayList < Audiolibro > listaDeAudiolibros;
private ArrayList < Digital > listaDeDigital;
private ArrayList < Fisico > listaDeFisico;

Autor()
getNombre()
setNombre()
getFechaNacimiento()
setFechaNacimiento()
addAudiolibro ()
addDigital ()
addFisico ()
mostrarAutor()
mostrarLibros()

IARCHIVO

escribirArchivo( ){

ARCHIVO

escribirArchivo( ){