Solución Desafío - Carrito de compras

Estructura del proyecto

La clase Artículo

Contiene este constructor y atributos encapsulados:

```
public abstract class Articulo {

    protected int talla;
    protected String marca;
    protected double precio;
    protected String nombre;
    protected String codigo;

    public Articulo(int talla, String marca, double precio, String nombre,
String codigo) {
        super();
        this.talla = talla;
        this.marca = marca;
        this.precio = precio;
        this.nombre = nombre;
        this.codigo = codigo;
    }

    // Encapsulamiento
}
```

La clase Pantalón

Extiende de Artículo y tiene color y cantidadBolsillos.

```
public class Pantalon extends Articulo {

    private String color;
    private int cantidadBolsillos;

    public Pantalon(int talla, String marca, double precio, String nombre,
String codigo) {
        super(talla, marca, precio, nombre, codigo);
    }

    public Pantalon(int talla, String marca, double precio, String nombre,
String codigo, String color,
            int cantidadBolsillos) {
        super(talla, marca, precio, nombre, codigo);
        this.color = color;
        this.cantidadBolsillos = cantidadBolsillos;
    }

    //Encapsulamiento

    @Override
    public String toString() {
        return "Pantalon [color=" + color + ", cantidadBolsillos=" +
cantidadBolsillos + ", talla=" + talla + ", marca="
                + marca + ", precio=" + precio + ", nombre=" + nombre + ",
codigo=" + codigo + "]";
    }


}
```

La clase Zapato

Extiende de Artículo y tiene el atributo modelo:

```
public class Zapato extends Articulo {

    private String modelo;

    public Zapato(int talla, String marca, double precio, String nombre,
String codigo) {
        super(talla, marca, precio, nombre, codigo);
        // TODO Auto-generated constructor stub
    }

    public Zapato(int talla, String marca, double precio, String nombre,
String codigo, String modelo) {
        super(talla, marca, precio, nombre, codigo);
        this.modelo = modelo;
    }

    //Encapsulamiento

    @Override
    public String toString() {
        return "Zapato [modelo=" + modelo + ", talla=" + talla + ", marca=" +
marca + ", precio=" + precio + ", nombre="
                + nombre + ", codigo=" + codigo + "]";
    }
}
```

La clase Polerón

Extiende de Artículo y tiene el atributo color:

```
public class Poleron extends Articulo {

    private String color;

    public Poleron(int talla, String marca, double precio, String nombre,
String codigo) {
        super(talla, marca, precio, nombre, codigo);
        // TODO Auto-generated constructor stub
    }

    public Poleron(int talla, String marca, double precio, String nombre,
String codigo, String color) {
        super(talla, marca, precio, nombre, codigo);
        this.color = color;
    }

    //Encapsulamiento

    @Override
    public String toString() {
        return "Poleron [color=" + color + ", talla=" + talla + ", marca=" +
marca + ", precio=" + precio + ", nombre="
                + nombre + ", codigo=" + codigo + "]";
    }

}
```

La clase Orden

Utilizada para las ordenes de despacho, contiene los siguientes atributos encapsulados y un constructor con todos ellos:

```
public class Orden {

    private String direccion;
    private String nombreCliente;
    private List<Articulo> articulos;
    private double montoTotal;
    private Date fechaCompra;
    public Orden(String direccion, String nombreCliente, List<Articulo>
articulos, double montoTotal,
            Date fechaCompra) {
        super();
        this.direccion = direccion;
        this.nombreCliente = nombreCliente;
        this.articulos = articulos;
        this.montoTotal = montoTotal;
        this.fechaCompra = fechaCompra;
    }
    //Encapsulamiento

}
```

La interfaz MedioPago

Contiene el método pagar, que está pensado para devolver true si el pago se realiza correctamente en sus implementaciones.

```
public interface MedioPago {

    boolean pagar(double valor);

}
```

La clase Crédito que implementa MedioPago

Contiene el comportamiento de recibir la cantidad de cuotas.

```
public class Credito implements MedioPago{

    @Override
    public boolean pagar(double valor) {

        int cicloPago = 1;
        String codigo;
        while(cicloPago == 1) {
            codigo = Escaner.leerRespuestaStringConMensaje("¿Está seguro que
desea pagar con credito el total de $"+valor+"?");
            if(codigo.equals("Y")) {
                Integer cuotas = Escaner.leerNumeroConMensaje(24, 0, "¿Con
cuantas cuotas desea pagar ? (0-24)");
                codigo = Escaner.leerRespuestaStringConMensaje("¿Está seguro
que desea pagar con "+cuotas+" cuotas?");
                if(codigo.equals("Y")) {
                    return true;
                }
            }else {
                cicloPago = 0;
            }
        }
        return false;
    }
}
```

La clase Débito que implementa MedioPago

Tiene la implementación del método pagar, para recibir un pago por un valor con Debito.

```
public class Debito implements MedioPago{

    @Override
    public boolean pagar(double valor) {
        String codigo = Escaner.leerRespuestaStringConMensaje("¿Está seguro
que desea pagar con débito el total de $"+valor+"?");
        if(codigo.equals("Y")) {
            return true;
        }
        return false;
    }
}
```

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

```
public abstract class Escaner {

    //Escaner estático
    static Scanner sc = new Scanner(System.in);

    //Recibe un mensaje para imprimir en el output y luego se encarga de que
el usuario ingrese un numero valido
    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;
    }

    //Recibe un mensaje para imprimir en el output y luego se encarga de que
el usuario ingrese un texto valido
    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;
    }

    //Recibe un mensaje para imprimir en el output y luego se encarga de que
el usuario ingrese Y o N
    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;
    }
}
```

La clase XmlHandler

Esta clase se encarga de manejar las interacciones con los archivos XML que se creen por la aplicación. El método exportar contiene la creación de un archivo con el nombre "ordenes_fecha.xls", el mismo recibe una Orden por parámetro, la que contiene la información que se escribira en la primera fila en blanco del archivo xls. Si el archivo xls no existe, se arrojará un FileNotFoundException al leer el archivo. Cuando esto ocurra, se llamará al método crearArchivo(String ruta), pasándole por parámetro la ruta de creación de un nuevo archivo. Si esto no falla, se continuará ejecutando el método exportar y se podrá editar el archivo creado.

```
public class XmlHandler implements Exportador{

    @Override
    public HandlerResponse exportar(Orden orden) {
        SimpleDateFormat SDF = new SimpleDateFormat("dd-MM-yyyy");
        String fechaHoyString = SDF.format(new Date());
        String excelFilePath = ".\\ordenes_"+fechaHoyString+".xls";

        try {
            FileInputStream inputStream = null;
            try {
                inputStream = new FileInputStream(new File(excelFilePath));
            }catch(FileNotFoundException fnfe) {
                crearArchivo(excelFilePath);
            }
            inputStream = new FileInputStream(new File(excelFilePath));
            Workbook workbook = WorkbookFactory.create(inputStream);

            Sheet sheet = workbook.getSheetAt(0);

            Object[][] bookData = {
                    {orden.getDireccion(), orden.getNombreCliente(),
Arrays.toString(orden.getArticulos().toArray()), orden.getMontoTotal(),
                        SDF.format(orden.getFechaCompra())}
            };

            int rowCount = sheet.getLastRowNum();

            for (Object[] aBook : bookData) {
                Row row = sheet.createRow(++rowCount);

                int columnCount = 0;
```

```
                Cell cell = row.createCell(columnCount);
                cell.setCellValue(rowCount);

                for (Object field : aBook) {
                    cell = row.createCell(++columnCount);
                    if (field instanceof String) {
                        cell.setCellValue((String) field);
                    } else if (field instanceof Integer) {
                        cell.setCellValue((Integer) field);
                    }
                }

            }

            inputStream.close();

            FileOutputStream outputStream = new
FileOutputStream(excelFilePath);
            workbook.write(outputStream);
            workbook.close();
            outputStream.close();
            return new HandlerResponse("Orden de despacho generada
correctamente", true);

        } catch (Exception ex) {
            ex.printStackTrace();
            return new HandlerResponse("Error al procesar orden, intente
nuevamente", false);
        }
    }

    public void crearArchivo(String ruta) {
        HSSFWorkbook libro = new HSSFWorkbook();
        libro.createSheet();
        try {
           FileOutputStream elFichero = new FileOutputStream(ruta);
           libro.write(elFichero);
           elFichero.close();
           libro.close();
        } catch (Exception e) {
           e.printStackTrace();
        }
        try {
            libro.close();
        } catch (Exception e) {
           e.printStackTrace();
        }
    }
}
```

La clase HandlerResponse

Esta clase se utiliza para que el XMLHandler devuelva una respuesta con información legible, devolverá un mensaje de error para el usuario y un booleano indicando si se pudo crear la fila con la orden de despacho en el archivo xls:

```
public class HandlerResponse {

    String mensaje;
    Boolean resultado;

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

    //Encapsulamiento
}
```

La clase Ejecutable

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

```
public static List<Articulo> artDisponibles = new ArrayList<>();
public static List<Articulo> artParaComprar = new ArrayList<>();
public static XmlHandler hdlr = new XmlHandler();

private static void generarArticulos() {
        artDisponibles.add(new Pantalon(30, "Variant", 15000, "Variant Caqui
Corto", "1000", "Caqui", 2));
        artDisponibles.add(new Pantalon(30, "Variant", 19000, "Variant Negro
Largo", "1001", "Negro", 2));
        artDisponibles.add(new Poleron(29, "Variant", 9000, "Variant Rojo",
"2002", "Rojo"));
        artDisponibles.add(new Zapato(34, "Mega", 25000, "Mega CZ50", "3003",
"CZ50"));
        artDisponibles.add(new Zapato(40, "Mega", 20000, "Mega CX50", "3004",
"CX50"));
    }

    private static void imprimirProductos() {
        System.out.println("-----------------------------------------------
");
        for(Articulo articulo : artDisponibles) {
            System.out.print("- ");
            System.out.print(articulo.toString());
            System.out.println(" ");
        }
```

```
        System.out.println("-----------------------------------------------
");
    }
```

El primer método llena la lista estática artDisponibles de la misma clase con los articulos del enunciado. El segundo ejecuta la opción 2 del menú, que muestra cada artículo dentro de la lista artDisponibles.

El tercer método es el main, que contiene un ciclo while, que finaliza al seleccionar la opción Salir del menú. Mientras se está dentro del while, se utilizará un java.util.Scanner gracias a la clase Escaner, el que leerá las opciones relativas al menú, ejecutando el algoritmo correspondiente a cada opción:

```
public static void main(String[] args) {
        generarArticulos();
        int valor = 4;
        String codigo = null;
        while(valor > 0) {
            valor = Escaner.leerNumeroConMensaje(3,0,"¿Qué desea hacer?\n1-
Agregar un articulo al carrito\n2- Ver productos disponibles\n3- Pagar\n0-
Salir");
            if(valor == 1) {
                codigo = Escaner.leerStringConMensaje("Ingrese el código del
articulo que desea agregar");
                if(null != codigo) {
                    for(Articulo articulo : artDisponibles) {
                        System.out.println(articulo.getCodigo());
                        if(codigo.equals(articulo.getCodigo())) {
                            codigo =
Escaner.leerRespuestaStringConMensaje("¿Está seguro de agregar
"+articulo.getNombre()+" al carrito por $"+articulo.getPrecio()+" ?");
                            if(codigo.equals("Y")) {
                                artParaComprar.add(articulo);
                                System.out.println(articulo.getNombre()+"
agregado correctamente.");
                                valor = 4;
                            }else {
                                System.out.println("No se agregó el
producto");
                            }
                            break;
                        }
                    }
                }
            }else if(valor == 2) {
                imprimirProductos();
            }else if(valor == 3) {
                MedioPago mp = null;
                double valorTotal = 0;
                for(Articulo articulo : Ejecutable.artParaComprar) {
                    valorTotal += articulo.getPrecio();
                }
                boolean pagado = false;
                boolean cancelado = false;
                while(!pagado && !cancelado) {
                    valor = Escaner.leerNumeroConMensaje(3, 1, "¿Cuál es su
medio de pago?\n1- Credito\n2- Debito\n3- Cancelar");
                    if(valor==1) {
```

```
                        mp = new Credito();
                        pagado = mp.pagar(valorTotal);
                    }else if(valor==2){
                        mp = new Debito();
                        pagado = mp.pagar(valorTotal);
                    }else {
                        cancelado = true;
                    }
                }
                if(pagado) {
                    String direccion = Escaner.leerStringConMensaje("¿Cuál es
la dirección de despacho?");
                    String nombreCliente = Escaner.leerStringConMensaje("¿Cuál
es el nombre de quien recibe el producto?");

                    HandlerResponse resp = hdlr.exportar(new
Orden(direccion,nombreCliente,artParaComprar,valorTotal,new Date()));
                    if(resp.getResultado()) {
                        artParaComprar = new ArrayList<>();
                    }
                    System.out.println(resp.getMensaje());
                }
            }else if(valor == 0) {
                System.out.println("Hasta pronto.");
            }
            codigo = null;
        }


    }
```

Resultado por consola de una interacción completa:

```
¿Qué desea hacer?
1- Agregar un articulo al carrito
2- Ver productos disponibles
3- Pagar
0- Salir
1
Ingrese el código del articulo que desea agregar
1000
1000
¿Está seguro de agregar Variant Caqui Corto al carrito por $15000.0 ?
Ingrese Y para si o N para no
Y
Variant Caqui Corto agregado correctamente.
¿Qué desea hacer?
1- Agregar un articulo al carrito
2- Ver productos disponibles
3- Pagar
0- Salir
3
¿Cuál es su medio de pago?
1- Credito
2- Debito
3- Cancelar
2
¿Está seguro que desea pagar con débito el total de $15000.0?
Ingrese Y para si o N para no
Y
¿Cuál es la dirección de despacho?
TYHT
¿Cuál es el nombre de quien recibe el producto?
JUYJ
Orden de despacho generada correctamente
¿Qué desea hacer?
1- Agregar un articulo al carrito
2- Ver productos disponibles
3- Pagar
0- Salir
2
-------------------------------------------------
- Pantalon [color=Caqui, cantidadBolsillos=2, talla=30, marca=Variant,
precio=15000.0, nombre=Variant Caqui Corto, codigo=1000]
- Pantalon [color=Negro, cantidadBolsillos=2, talla=30, marca=Variant,
precio=19000.0, nombre=Variant Negro Largo, codigo=1001]
- Poleron [color=Rojo, talla=29, marca=Variant, precio=9000.0, nombre=Variant
Rojo, codigo=2002]
- Zapato [modelo=CZ50, talla=34, marca=Mega, precio=25000.0, nombre=Mega CZ50,
codigo=3003]
- Zapato [modelo=CX50, talla=40, marca=Mega, precio=20000.0, nombre=Mega CX50,
codigo=3004]
-------------------------------------------------
¿Qué desea hacer?
1- Agregar un articulo al carrito
```

```
2- Ver productos disponibles
3- Pagar
0- Salir
1
Ingrese el código del articulo que desea agregar
1000
1000
¿Está seguro de agregar Variant Caqui Corto al carrito por $15000.0 ?
Ingrese Y para si o N para no
Y
Variant Caqui Corto agregado correctamente.
¿Qué desea hacer?
1- Agregar un articulo al carrito
2- Ver productos disponibles
3- Pagar
0- Salir
3
¿Cuál es su medio de pago?
1- Credito
2- Debito
3- Cancelar
1
¿Está seguro que desea pagar con credito el total de $15000.0?
Ingrese Y para si o N para no
Y
¿Con cuantas cuotas desea pagar ? (0-24)
10
¿Está seguro que desea pagar con 10 cuotas?
Ingrese Y para si o N para no
Y
¿Cuál es la dirección de despacho?
Casa 123
¿Cuál es el nombre de quien recibe el producto?
Juan Perez
Orden de despacho generada correctamente
¿Qué desea hacer?
1- Agregar un articulo al carrito
2- Ver productos disponibles
3- Pagar
0- Salir
0
Hasta pronto.
```