Universidad Mariano Gálvez
Facultad de Ingeniería en Sistemas de Información y Ciencias de la Computación
Boca del Monte, Villa Canales
Programación II
Ing. Luis Fernando Alvarado Cruz
Sección: “B”

PROYECTO FINAL
PROGRAMACION II

Daniel Estuardo Revolorio Alvarez 7690-22-4444

Guatemala 07 de noviembre del 2023

PATRONES DE DISEÑO UTILIZADO

- Factory Method
- Abstract Factory

Descripción de la Clase Cliente:

La clase Cliente representa la información de un cliente y su pedido en una aplicación Java.

Atributos:

nombres (Cadena): Almacena el nombre del cliente.

apellidos (Cadena): Almacena los apellidos del cliente.

cui (Cadena): Almacena el CUI (Código Único de Identificación) del cliente.

nit (Cadena): Almacena el NIT (Número de Identificación Tributaria) del cliente.

pedido (Map<String, Integer>): Almacena el pedido del cliente como un mapeo de productos a la cantidad solicitada.

total a pagar(doble): Almacena el total a pagar por el cliente.

Métodos:

Cliente (): Constructor por defecto de la clase Cliente que inicializa un nuevo cliente con un pedido vacío y un total a pagar de 0.0.

getNombres(): Obtiene el nombre del cliente.

setNombres(String nombres): Establece el nombre del cliente.

getApellidos(): Obtiene los apellidos del cliente.

setApellidos(String apellidos): Establece los apellidos del cliente.

getCui(): Obtiene el CUI (Código Único de Identificación) del cliente.

setCui(String cui): Establece el CUI del cliente.

getNit(): Obtiene el NIT (Número de Identificación Tributaria) del cliente.

setNit(String nit): Establece el NIT del cliente.

getPedido(): Obtiene el pedido del cliente como un mapeo de productos y cantidades.

actualizarTotalAPagar(double precioProducto, int cantidad): Actualiza el total a pagar del cliente al agregar un producto al pedido, multiplicando el precio del producto por la cantidad.

getTotalAPagar(): Obtiene el total a pagar del cliente.

Descripción de la Conexión de Clase:

La clase Conexionse utiliza para gestionar la conexión a una base de datos MySQL y realizar operaciones relacionadas con productos en una tienda de videojuegos.

Atributos:

conni(Conexión): Almacena la conexión a la base de datos.

host(Cadena): Almacena la dirección del servidor de la base de datos.

port(Cadena): Almacena el puerto de la base de datos.

dbName(Cadena): Almacena el nombre de la base de datos.

userName(Cadena): Almacena el nombre de usuario para acceder a la base de datos.

password(Cadena): Almacena la contraseña para acceder a la base de datos.

instance(Conexión estática): Almacena una instancia única de la clase Conexion.

Métodos:

Conexion(): Constructor de la clase que establece una conexión a la base de datos MySQL utilizando los valores de conexión proporcionados. Si se produce un error en la conexión, se mostrará un mensaje de error.

cerrarConexion(): Cierra la conexión a la base de datos, si está abierta. Puede generar un mensaje de error si la conexión no se puede cerrar.

getProductoPorNombre(String nombreProducto): Recupera un producto de la base de datos con el nombre especificado y lo devuelve como un objeto de la clase Producto. Si no se encuentra el producto, devuelve null.

obtenerPrecioProducto(String nombreProducto): Obtiene el precio de un producto de la base de datos basado en su nombre. Retorna el precio del producto o -1.0 si el producto no se encuentra.

getProductosFromDatabase(): Recupera todos los productos de la base de datos y los devuelve como una lista de objetos Producto.

agregarProductoALaBaseDeDatos(String productoNombre, int cantidadInicial): Agrega un nuevo producto a la base de datos con un nombre y una cantidad inicial especificada. El precio se establece en 0.

aumentarCantidadEnBaseDeDatos(String productoNombre, int cantidadAumentar): Aumente la cantidad de un producto existente en la base de datos en función de su nombre. Esto se utiliza para registrar nuevas existencias de productos.

Descripción de la Clase VideojuegosApp:

La clase VideojuegosApp es una parte de una aplicación de videojuegos en Java que extiende la clase JFrame. Esta clase se utiliza para gestionar la interfaz de usuario de la aplicación y establecer una conexión con la base de datos de la tienda de videojuegos.

Atributos:

conexion(Conexión): Almacena una instancia de la clase Conexión que se utiliza para conectarse a la base de datos de la tienda de videojuegos.

Métodos:

VideojuegosApp (): Constructor de la clase VideojuegosApp se encarga de configurar el manejo del evento de cierre de la ventana. Cuando la ventana se cierra, verifique si existe una conexión y la cierra de manera apropiada.

main(String[] args): El método main el punto de entrada de la aplicación. Aquí se inicia la aplicación de la tienda de videojuegos. Dentro de este método, se realiza lo siguiente:

Se inicia la interfaz de usuario de la aplicación y se crea una instancia de la clase VideojuegosApp.

Se crea una conexión a la base de datos utilizando la clase Conexión y se muestra un mensaje de conexión exitosa si la conexión se establece correctamente.

Se crea una instancia de la clase VideojuegosGUI que maneja la interfaz gráfica de la aplicación y se hace visible.

Se configura un KeyEventDispatcherpara que, al presionar la tecla "Enter", se simule un clic en el botón actualmente enfocado en la interfaz de usuario, lo que facilita la interacción del usuario con la aplicación.

Descripción de la Clase Producto:

La clase Producto representa un artículo en una tienda de videojuegos. Cada instancia de esta clase contiene información sobre un producto, incluyendo su nombre, precio y cantidad disponible en el inventario.

Atributos:

nombre (Cadena): Almacena el nombre del producto.

precio(doble): Almacena el precio del producto.

cantidad(int): Almacena la cantidad de unidades disponibles del producto en el inventario.

Métodos:

Producto (String nombre, double precio, int cantidad): Constructor de la clase videojuegos. Se utiliza para crear una nueva instancia de producto proporcionando el nombre, el precio y la cantidad inicial disponible. Este constructor inicializa los atributos de la instancia con los valores proporcionados.

getNombre(): Método que permite obtener el nombre del videojuego. Retorna una cadena de caracteres que representa el nombre del producto.

getPrecio(): Método que permite obtener el precio del videojuego. Retorna un valor decimal (doble) que representa el precio del producto.

getCantidad(): Método que permite obtener la cantidad disponible del videojuego en el inventario. Retorna un valor entero (int) que representa la cantidad de unidades del videojuego disponibles.

Clase VideojuegosGUI

La clase VideojuegosGUI representa la interfaz gráfica de usuario (GUI) para una aplicación de videojuegos en Java. La clase es una extensión de JFrame, lo que significa que crea una ventana principal para la aplicación. A continuación, se proporciona una descripción detallada de esta clase y sus métodos:

Atributos:

windows(Map<String, JFrame>): Un mapa que almacena ventanas secundarias creadas dinámicamente, donde la clave es el título de la ventana y el valor es la instancia de JFrame asociada.

Conexión (Conexión): Una instancia de la clase Conexión que se utiliza para conectarse a la base de datos de la tienda de videojuegos.

tableModel(DefaultTableModel): Un modelo de tabla utilizado para mostrar datos tabulares en la interfaz de usuario.

Table (JTable): Una tabla que muestra datos tabulares en la interfaz de usuario.

Cliente (Cliente): Una instancia de la clase Cliente que representa los datos de un cliente.

productosAgregados (List <String>): Una lista que almacena los nombres de productos agregados.

Métodos:

VideojuegosGUI (Conexión conexión): Constructor de la clase que recibe una instancia de Conexión. En este constructor, se configura la interfaz de usuario de la aplicación, que consta de pestañas ("Ventas", "Compras", "Inventario" y "Reportes") que contienen diferentes funcionalidades.

abrirVentana(String título): Un método que abre una ventana secundaria según el título proporcionado. Dependiendo del título, puede mostrar información relacionada con la interacción con el cliente, productos en existencia u otras ventanas específicas.

mostrarPanelFactura(): Un método que crea y muestra un panel de facturación en la interfaz de usuario. Permite al buscar usuario y generar facturas, mostrando detalles de las mismas.

generarFactura(String nombresApellidos): Un método que genera una factura en PDF basada en los nombres o formatos del cliente proporcionados. Extrae datos de la base de datos y crea un PDF con la información de la factura.

mostrarVentanaFacturaGenerada(JFrame ventanaFactura, JPanel panelFactura, DefaultTableModel facturaTableModel): Un método que muestra una ventana que contiene los detalles de la factura generada en formato PDF. También incluye un botón para generar el PDF y abrirlo en el visor predeterminado.

obtenerDatosClienteDesdeBD(): Este método se utiliza para recuperar información del cliente desde la base de datos. Realiza una consulta SQL para seleccionar los campos "NOMBRES", "APELLIDOS", "CUI" y "NIT" de la tabla "clientes". Luego, crea un objeto Clientey establece sus atributos con los valores recuperados de la base de datos. Finalmente, retorna el objeto Cliente.

obtenerTotalAPagarDesdeBD(String nombresApellidos): Este método recibe el nombre y apellidos del cliente como parámetro y busca en la base de datos la suma de "total_pagar" de la tabla "factura" donde el nombre o los apellidos coinciden con el valor proporcionado. Retorna el total a pagar como un valor doble.

buscarFacturasEnBaseDeDatos(String nombresApellidos, DefaultTableModel facturaTableModel): Este método busca facturas en la base de datos que coinciden con un nombre o apellidos proporcionados. Luego, llena una tabla (facturaTableModel) con los resultados de la consulta y muestra los resultados en una interfaz gráfica.

actualizarInventario(): Este método actualiza una tabla de productos en la interfaz gráfica al obtener la lista de productos desde la base de datos y mostrarlos en la tabla.

mostrarVentanaDatosCliente(): Este método muestra una ventana en la que el usuario puede ingresar datos del cliente, como nombres, apellidos, CUI y NIT. Luego, guarda estos datos en la base de datos al hacer clic en el botón "Continuar".

guardarClienteEnBaseDeDatos(Cliente cliente): Este método toma un objeto Clientecomo argumento y guarda sus datos en la tabla "clientes" de la base de datos.

mostrarVentanaPedido(): Este método muestra una ventana donde se pueden agregar productos al pedido de un cliente. También muestra una lista de productos disponibles en la base de datos y permite al usuario seleccionar la cantidad de productos que desea agregar al pedido.

obtenerCantidadDisponible(String productoNombre): Este método recibe el nombre del producto y busca en la base de datos la cantidad disponible en el inventario. Retorna la cantidad disponible.

disminuirCantidadEnBaseDeDatos(String productoNombre, int cantidad): Este método disminuye la cantidad de un producto en la base de datos después de que se agrega al pedido de un cliente.

mostrarVentanaFactura(Cliente cliente, double totalAPagar): Este método muestra una factura para el cliente con los detalles de los productos en su pedido y el precio total a pagar. También proporciona la opción de generar un archivo PDF de la factura.

GestionarPedidoConProveedores(): Este método permite gestionar un pedido a proveedores, mostrando una ventana donde se pueden agregar productos al pedido y aumentar la cantidad disponible en la base de datos. También proporciona la opción de finalizar el pedido a proveedores.

guardarFacturaEnBaseDeDatos(Cliente cliente, Map<String, Integer> pedidoCliente, double totalAPagar): Este método guarda una factura en la base de datos, registrando los detalles del cliente, los productos en el pedido y el precio total.