INFORME DE PROYECTO

B2

“SUPERMAXI”

Carrera: Computación

Asignatura: Introducción a la Programación

Docente:

- Irene Robalino Pedro Daniel

Grupo: 10

Integrantes:

- Jesús Rivas
- Carlos Carrillo

Periodo Académico:

- Octubre 2024 – Febrero 2025

Ejercicio:

1. TEMA DE PROYECTOS NRO. 1: Sistema de facturación del SuperMaxi - Loja
El objetivo del proyecto es desarrollar un sistema de facturación para el SuperMaxi en Loja. Este sistema deberá permitir la facturación de N productos, considerando precios normales y promocionales cuando existan muchos productos en stock o su fecha de caducidad esté próxima. Además, se deberá realizar una factura que resuma los totales de impuestos a la renta deducibles por productos en las siguientes categorías: Vivienda, Educación, Alimentación, Vestimenta y Salud. Al final del día, se generará una estadística de ventas totales, por productos y categorías, que ayudará a los gerentes del SuperMaxi en la toma de decisiones.
Características a considerar:
 Gestión de Productos: Implementar métodos para agregar y gestionar productos en el sistema, considerando su cantidad en stock, fecha de caducidad y precios normales y promocionales.
 Facturación: Desarrollar un sistema que calcule el monto total de la factura, teniendo en cuenta los precios normales y promocionales, y que muestre un resumen de los impuestos a la renta deducibles por cada categoría de producto.
 Estadísticas de Ventas: Generar métodos para recopilar estadísticas de ventas diarias, que incluyan las ventas totales y desgloses por productos y categorías, para tomar decisiones gerenciales.

Fases de Análisis y Diseño:

Datos de Entrada: Para los datos de entrada, utilizamos en nuestro primer menú el numero ingresado para hacer diferentes procesos según lo quiera el usuario, asi mismo dentro de las funciones/procedimientos:

- **GestionProductos(String Productos[][]):** Se abre un menú en el cual se pide un numero para ingresar en invocar los procesos que quiera el usuario.
- **AgregarProducto(String Productos[][]):** Se pide como entrada el nombre del Producto y así mismo sus demás características, como lo son el precio, stock, fecha de vencimiento y su categoría.
- **ModStock(String Productos[][]):** Se pide al usuario que ingrese el nombre del producto el cual desee modificar, y así mismo se verifica su existencia y se pide que ingrese el nuevo Stock del producto.
- **Facturacion(String Productos[][], String productProms[], double ventas[], double deducibles[]):** Aquí se pide al usuario ingresar el nombre del producto o “F” para finalizar la facturación, si el usuario ingresa un nombre existente, se le pedirá la cantidad y a partir de eso se calculará los elementos de la facturación, y se comprobará si el producto está en promoción es decir, si se encuentra el nombre dentro del arreglo productProms[], y así se calculará el descuento y así mismo se guardara y se restara,

para conseguir el subtotal, luego se sacara el IVA y este se sumara junto con el subtotal en una variable llamada total. Así mismo guardaremos las ventas en nuestro arreglo ventas[], y sacaremos los deducibles en nuestro arreglo deducibles[].

Procesos: Para los procesos usaremos procedimientos en los cuales veremos cuales son los productos que puedan estar aptos para estar en promoción, y asi mismo en el procedimiento de Facturación:

- **copyPro(String Productos[][]) :** En esta función se copia la variable global products[][] en la variable local Productos[][], para que luego la podamos usar y así mismo imprimirla.
- **generarActualiProms(String Productos[][], String productProms[]):** En este procedimiento se verifica con un ciclo el stock y la fecha de vencimiento que si es <= 30 aplicará para promoción, y este se guardará el nombre en el arreglo productProms[].

Salida: Para los datos de salida los cuales se imprimirán por pantalla usaremos funciones/procedimientos.

- **mostrarProduct(String Productos[][]):** En esta función se guarda en una String el diseño y cada iteración para poder imprimirla luego con un sout, cuando el usuario escoja la opción de imprimirla, así mismo esta se actualizará cuando se agregue un producto o se modifique el stock.
- **presentarProms(String Productos[][], String productProms[]):** En este procedimiento se imprimirá los porductos en promoción, usando un ciclo for, para imprimir los productos.
- **dbEstadis(String Productos[][], double ventas[], double ventasCate[]):** En este procedimiento se imprimirá las ventas por productos, usando un ciclo for, en el cual se pondrá una condición para que se imprima, luego usaremos un switch para acumular en función de a que categoría pertenezca, y luego lo imprimiremos por categoría, así mismo acumularemos todas esas ventas y las guardaremos en una variable aparte y lo imprimiremos al final.

Diseño Modular:

Modulo Principal:

public static void main(String[] args)

Función: Controla los datos de entrada y de salida, mediante el menú creado y su correcto funcionamiento.

Llamado de Funciones/Procedimientos:

- copyPro(Productos);
- (mostrarProduct(Productos));
- generarActualiProms(Productos, productProms);
- GestionProductos(Productos);
- presentarProms(Productos, productProms);
- Facturacion(Productos, productProms, ventas, deducibles);
- dbEstadis(Productos, ventas, ventasCate);

Módulos:

Modulo: copyPro(Productos);

Función: Copia la matriz global "products" en la matriz local "Productos"

Procesos: Recorre la matriz global "products" con un ciclo for y la guarda en la matriz local "Productos"

Modulo: mostrarProduct(String Productos[][])

Función: Guardar en una String, el diseño y contenido de la matriz local "Productos", para así imprimirla en el main

Procesos: Recorre la matriz local "Productos" con un ciclo for, y lo guarda en la String "mostrar", así mismo coloca un diseño y retorna la String, para que con un "sout" sea prsentada al usuario.

Modulo: GestionProductos(String Productos[][])

Función: Mostrar un submenú en el cual así mismo como en el main, llame según el numero ingresado por el usuario a otros procedimientos.

Procesos: Llamar a los procedimientos: AgregarProducto(Productos);, ModStock(Productos); , .

Modulo: AgregarProducto(String Productos[][])

Función: Agregar productos a la matriz local "Productos", dándole valores y así pueda ser mostrada en las demás funciones

Procesos: Verifica la disponibilidad de espacio en la matriz local "Productos", en función de eso le permite al usuario poder ingresar un producto y sus características (precio, stock, fecha de vencimiento, categoría) y lo guarda gracias a un ciclo for en la "i" posición.

Modulo: ModStock(String Productos[][])

Función: Modificar el stock de un producto que contenga la matriz local "Productos"

Procesos: Pide al usuario que ingrese el nombre del producto, lo guarda en una variable y la compara en un ciclo for, si se cumple la condición de que el nombre coincida con el nombre del producto en su posición "i", le muestra el usuario el stock actual, y le pide ingresar el nuevo stock, el cual reemplaza al anterior.

Modulo: generarActualiProms(String Productos[][], String productProms[])

Función: Evaluar que productos cumplen las condiciones para ser promociones (%10) y guardarlos en un arreglo llamado "productProms"

Procesos: Con un ciclo for ingresa a la matriz local "Productos", convierte los datos String en datos enteros, y de fecha, usa un condicional el cual tiene la siguiente condición: if (stock > 100 || diasParaVencer <= 30), si se cumple, el nombre del producto en su posición "i" se guardará en el arreglo "productProms", y así, el producto se pondrá en promoción.

Modulo: presentarProms(String Productos[][], String productProms[])

Función: Presentar el arreglo "productProms".

Procesos: Recorrer el arreglo "productProms", usando un ciclo for, con una condición if (productProms[i] != null) si es que contiene valores, estos se presentaran, pero si no tiene un mensaje se presentara en su lugar, el cual es: "NO HAY PROMOCIONES!!"

Modulo: Facturacion(String Productos[][], String productProms[], double ventas[], double deducibles[])

Función: Procesar la información del producto elegido por el usuario y presenta su subtotal, iva, total, y deducibles.

Procesos: Pedir al usuario el nombre del producto o en su defecto ingresar “F” para terminar el proceso, al ingresar un nombre, este mediante un ciclo for, verifica la existencia del producto, por consiguiente pide ingresar la cantidad de producto que desea adquirir, esta cantidad se verifica, si se excede pedirá que ingrese otra, ya que la excede, pero si está dentro del rango, seguirá verificando si el producto es parte del arreglo “productProms”, y si es así, le asignará un descuento, por consiguiente se transformara el precio a double y esto se multiplicara por la cantidad y el precio, se guarda en subtotal; subtotalfinal se le asignara la resta de subtotal y descuento, luego se acumulara en el arreglo “ventas”, por consiguiente se actualizara el stock del producto, posterior a esto se calculara los deducibles con un switch donde Productos[i][4], deberá coincidir con una de las categorías, y el resultado se guardara en un arreglo “deducibles”, finalmente una vez ingresado “f”, se mostrara la factura con todos estos valores.

Modulo: dbEstadis(String Productos[][], double ventas[], double ventasCate[])

Función: Mostrar las estadísticas en función de: ventas por productos, ventas por categorías, y ventas totales, principalmente para decisiones gerenciales.

Procesos: Con el uso de un ciclo for y un condicional se imprimirá el nombre del producto es decir Productos[i][0] y sus ventas es decir ventas[i], así mismo en este ciclo for se irán acumulando los valores de ventas[i] en la variable totalVentas; posteriormente con ayuda de un switch, se verificara la coincidenia de Productos[i][4] con las categorías, de ser así, estas ventas se acumularan en el arreglo ventasCate[], finalmente estos valores se imprimirán y se podrán ver las estadísticas de las ventas y así poder tomar decisiones gerenciales, tal como lo requiere el problema.