ACTIVIDAD 02: **LA ANIDACIÓN ESTRUCTURAL: REGISTROS CON ARREGLOS, ARREGLOS DE REGISTROS Y ARREGLOS DE OBJETOS**

ESTUDIANTE: **EFRAIN ROBLES PULIDO**

CODIGO: **221350095**

NOMBRE DE LA MATERIA: **ESTRUCTURAS DE DATOS I**

SECCIÓN: ***D12***

CLAVE: ***I5886***

FECHA: **DOMINGO 30 DE ENERO DE 2022**

Tabla de autoevaluación:

Autoevaluación			
Concepto	**Sí**	**No**	**Acumulación**
Bajé el trabajo de internet o alguien me lo pasó (aunque sea de forma parcial)	-100 pts	0 pts	0 pts
Incluí el código fuente **en formato de texto** **(sólo si funciona cumpliento todos los requerimientos)**	+25 pts	0 pts	25 pts
Incluí las **impresiones de pantalla** **(sólo si funciona cumpliento todos los requerimientos)**	+25 pts	0 pts	25 pts
Incluí una **portada** que identifica mi trabajo (nombre, código, materia, fecha, título)	+25 pts	0 pts	25 pts
Incluí una **descripción y conclusiones** de mi trabajo	+25 pts	0 pts	25 pts
		Suma:	100 pts

Introducción:

Luego de leer las instrucciones detenidamente, tuve que realizar los archivos .h y .cpp necesarios para la actividad, un par por clase (fecha, producto, colección y el menú), en cada clase se le introdujo sus atributos necesarios según la actividad como privados, y los métodos como públicos que serían los setters y getters de cada atributo, también se le introdujo un método para pasar los valores a cadena, con ayuda de funciones y también de funciones de librería del string, para que en el código solo se impriman. Para los constructores se les asigna en donde se guardarán, así que también se usara la sobrecarga del operador de asignación para que se guarden los valores según el objeto creado.

En la clase producto se declaro para la fecha una variable tipo Fecha para la composición de clases y se utilizó la misma metodología de los setters y getters, y con el método string de la clase producto regresamos las variables tipo real a cadena con la función sprintf, y los reales con el to_string y se hacia suma de cadenas para el producto.

En la clase colección se le definió un arreglo de 500 y su índice para el almacenamiento de los datos del producto y que se usarían en los métodos que contendrá la clase, que serían los que manejaran la información como el agregar, buscar, incrementar o decrementar la existencia, así como el getter y setter del índice del arreglo de la clase colección y su método de conversión de valores a cadenas. Cuando se inicializa el objeto con el constructor, utilizamos instrucciones para guardar los valores en el arreglo con un ciclo iterativo for.

Y para la clase menú, se le escribirán los métodos necesarios para manejar la información de forma que pida la práctica, en donde en su constructor se le escribí la base del código de funcionamiento de la actividad. Y ya solo en el main se le hace un llamado al constructor para correr el programa completo.

Código fuente:

```
#ifndef FECHA_H_INCLUDED
#define FECHA_H_INCLUDED

#include <string>

class Fecha {
    private:
        int dia, mes, anio;

    public:
        Fecha();//Constructores
        Fecha(const Fecha&);

        int getDia()const;
        int getMes()const;
        int getAnio()const;

        std::string toString() const;//Funcion para pasar variables a
cadena

        void setDia(const int&);
        void setMes(const int&);
        void setAnio(const int&);
    };

#endif // FECHA_H_INCLUDED


#include "fecha.h"

using namespace std;
Fecha::Fecha() {} //Constructores
Fecha::Fecha(const Fecha& date):
dia(date.dia),mes(date.mes),anio(date.anio) {}

int Fecha::getDia() const {
    return dia;
    }

int Fecha::getMes() const {
    return mes;
    }

int Fecha::getAnio() const {
    return anio;
    }
```

```cpp
string Fecha::toString() const { //to_string es para convertir del VALOR
(x) a una CADENA
    return to_string(dia) + "/" + to_string(mes) + "/" + to_string(anio);
    }

void Fecha::setDia(const int& d) {
    dia = d;
    }

void Fecha::setMes(const int& m) {
    mes = m;
    }

void Fecha::setAnio(const int& a) {
    anio = a;
    }



#ifndef PRODUCTO_H_INCLUDED
#define PRODUCTO_H_INCLUDED

#include "Fecha.h"
#include <string>

class Producto {
    private:
        std::string codigoBarrasP;
        std::string nombreP;
        float pesoP;
        Fecha fechaIn;
        float precioMayoreoP;
        float precioMenudeoP;
        int existA;

    public:
        Producto(); //Constructores
        Producto(const Producto&);

        std::string getcodigoBarrasP() const;
        std::string getNombreP() const;
        float getPesoP() const;
        Fecha getFechaIn() const;
        float getPrecioMayoreoP() const;
        float getPrecioMenudeoP() const;
        int getExistP() const;

        std::string toString() const;//Regresa los valores como cadena

        void setcodigoBarrasP(const std::string&);
        void setNombreP(const std::string&);
        void setPesoP(const float&);
        void setFechaIn(const Fecha&);
        void setPrecioMayoreoP(const float&);
        void setPrecioMenudeoP(const float&);
        void setExistP(const int&);
```

```
        Producto& operator = (const Producto&);//Para asignar el valor
entrante a nuestro objeto

    };

#endif // PRODUCTO_H_INCLUDED



#include "producto.h"
#include "fecha.h"
#include <string>

using namespace std;

Producto::Producto() {} //Constructores
Producto::Producto(const Producto& otro):
codigoBarrasP(otro.codigoBarrasP),
    nombreP(otro.nombreP), pesoP(otro.pesoP), fechaIn(otro.fechaIn),
    precioMayoreoP(otro.precioMayoreoP),
precioMenudeoP(otro.precioMenudeoP),
    existA(otro.existA) {};

string Producto::getcodigoBarrasP() const {
    return codigoBarrasP;
    }

string Producto::getNombreP() const {
    return nombreP;
    }

float Producto::getPesoP() const {
    return pesoP;
    }

Fecha Producto::getFechaIn() const {
    return fechaIn;
    }

float Producto::getPrecioMayoreoP() const {
    return precioMayoreoP;
    }

float Producto::getPrecioMenudeoP() const {
    return precioMenudeoP;
    }

int Producto::getExistP() const {
    return existA;
    }

string Producto::toString() const {
    char str_precioMayoreoP[10], str_precioMenudeoP[10], str_peso[10];
    sprintf (str_peso, "%.2f", pesoP);
    sprintf (str_precioMayoreoP, "%.2f", precioMayoreoP);
    sprintf (str_precioMenudeoP, "%.2f", precioMenudeoP);
```

```
    //sprintf sirve para guardar como una cadena en el bÃºfer seÃ±alado
por str, con formato de un printf de un valor

    return "Codigo de barras: " + codigoBarrasP + "\n" +
           "Nombre del producto: " + nombreP + "\n" +
           "Peso: " + str_peso + " kg" + "\n" +
           "Fecha de entrada: " + fechaIn.toString() + "\n" +
           "Precio de mayoreo: $" + str_precioMayoreoP + "\n" +
           "Precio de menudeo: $" + str_precioMenudeoP + "\n" +
           "Existencia actual: " + to_string(existA) + "\n";
    }


void Producto::setcodigoBarrasP(const std::string& codigoBarrasNuevo) {
    codigoBarrasP = codigoBarrasNuevo;
    }

void Producto::setNombreP(const std::string& nombreNuevo) {
    nombreP = nombreNuevo;
    }

void Producto::setPesoP(const float& pesoNuevo) {
    pesoP = pesoNuevo;
    }

void Producto::setFechaIn(const Fecha& fechaNuevo) {
    fechaIn = fechaNuevo;
    }

void Producto::setPrecioMayoreoP(const float& precioMayoreoNuevo) {
    precioMayoreoP = precioMayoreoNuevo;
    }

void Producto::setPrecioMenudeoP(const float& precioMenudeoNuevo) {
    precioMenudeoP = precioMenudeoNuevo;
    }

void Producto::setExistP(const int& existNuevo) {
    existA = existNuevo;
    }

Producto& Producto::operator =(const Producto& otro) {
    codigoBarrasP = otro.codigoBarrasP;
    nombreP = otro.nombreP;
    pesoP = otro.pesoP;
    fechaIn = otro.fechaIn;
    precioMayoreoP = otro.precioMayoreoP;
    precioMenudeoP = otro.precioMenudeoP;
    existA = otro.existA;

    return *this;
    }


#ifndef COLECCION_H_INCLUDED
#define COLECCION_H_INCLUDED
```

```
#include "producto.h"
#include "Fecha.h"
#include <string>

class Coleccion {
    private:
        Producto inventario[500];
        int indexInventario;

    public:
        Coleccion();//Constructores
        Coleccion(const Coleccion&);

        int getIndexInventario() const;
        void setIndexInventario(const int& indice);

        std::string toString(const int& index) const;//Regresa los
valores como cadena

        //Metodos
        void agregarProducto(const Producto& );
        int buscarProducto(const std::string&) const;
        void incrementaExistencia(const int&, const int&);
        void decrementaExistencia(const int&, const int&);

    };


#endif // COLECCION_H_INCLUDED



#include "coleccion.h"

using namespace std;

Coleccion::Coleccion() : indexInventario(0) {}//Inicializo en 0 en indice

Coleccion::Coleccion(const Coleccion& otro):
indexInventario(otro.indexInventario) { //Inicializo el arreglo para
empezar a guardar datos
    for(int x(0); x < indexInventario ; x++) {
        inventario[x] = otro.inventario[x];
        }
    }

int Coleccion::getIndexInventario() const {
    return indexInventario;
    }

void Coleccion::setIndexInventario(const int& indice) {
    indexInventario = indice;
    }

string Coleccion::toString(const int& index) const {
    string mystr;//Variable temporal
```

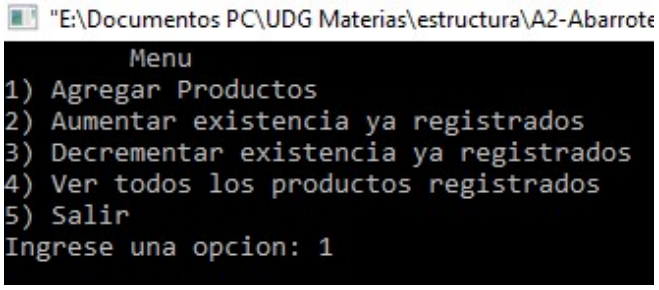
```
    mystr = "\tProducto #" + to_string(index+1) + "\n";
    mystr += inventario[index].toString() + "\n";
    return mystr;
    }

void Coleccion::agregarProducto(const Producto& otro) {
    inventario[indexInventario] = otro;
    indexInventario++;
    }

int Coleccion::buscarProducto(const std::string& otro) const {
    int indexNew(-5);
    for(int x(0); x < indexInventario; x++) {
        if (otro == inventario[x].getcodigoBarrasP()) {
            indexNew = x;
            break;
            }
        }
    return indexNew;
    }

void Coleccion::incrementaExistencia(const int& index, const int&
existActual) {
    int existNueva(0);
    existNueva = inventario[index].getExistP() + existActual;
    inventario[index].setExistP(existNueva);
    }

void Coleccion::decrementaExistencia(const int& index, const int&
existActual) {
    int existNueva(0);
    existNueva = inventario[index].getExistP() - existActual;
    if(existNueva < 0) {
        inventario[index].setExistP(0);
        }
    else {
        inventario[index].setExistP(existNueva);
        }
    }


#ifndef MENU_H_INCLUDED
#define MENU_H_INCLUDED
#include "fecha.h"
#include "producto.h"
#include "coleccion.h"
#include <iostream>
#include <string>
#include <conio.h>
#include <cctype>

class Menu {
    private:
        //Metodos
        int buscarIdProducto(const Coleccion&);
        Producto agregarProductoNuevo() const;
        void imprimirMenu()const;
```

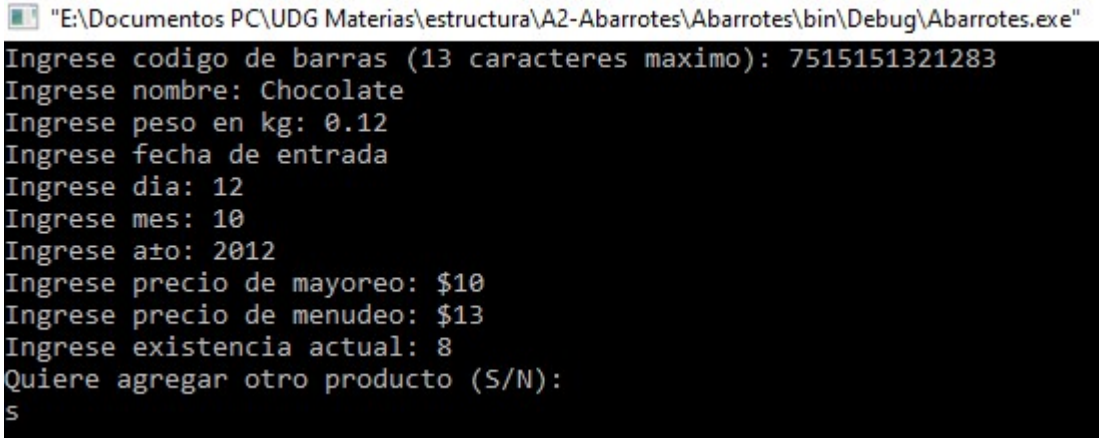
```
        void limpiarPantalla()const;
        void enterMenu()const;
    public:
        //Constructor
        Menu();
    };

#endif // MENU_H_INCLUDED


#include "menu.h"

using namespace std;

int Menu::buscarIdProducto(const Coleccion& otro) {
    string codigo;
    int idCodigo;

    limpiarPantalla();
    cout<<"Ingrese codigo de barras a buscar: "<<endl;
    cin>>codigo;
    idCodigo = otro.buscarProducto(codigo);

    return idCodigo;
    }

Producto Menu::agregarProductoNuevo() const {
    limpiarPantalla();

//Inicializacion de Objetos
    Producto nuevoProducto;
    Fecha nuevaFecha;

//Variables temporales
    string myStr;
    float myfloat;
    int myInt;

//Pedir Datos
    cout<<"Ingrese codigo de barras (13 caracteres maximo): ";
    cin.ignore();
    getline(cin,myStr);
    nuevoProducto.setcodigoBarrasP(myStr);

    cout<<"Ingrese nombre: ";
    getline(cin,myStr);
    nuevoProducto.setNombreP(myStr);

    cout<<"Ingrese peso en kg: ";
    cin>>myfloat;
    nuevoProducto.setPesoP(myfloat);

    cout<<"Ingrese fecha de entrada "<<endl;
    cout<<"Ingrese dia: ";
    cin>>myInt;
    nuevaFecha.setDia(myInt);
```

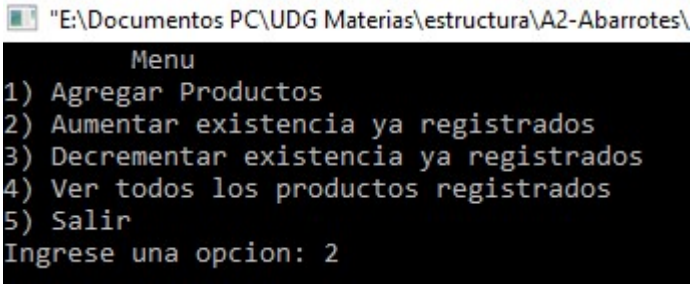
```
    cout<<"Ingrese mes: ";
    cin>>myInt;
    nuevaFecha.setMes(myInt);
    cout<<"Ingrese aÃ±o: ";
    cin>>myInt;
    nuevaFecha.setAnio(myInt);
    nuevoProducto.setFechaIn(nuevaFecha);

    cout<<"Ingrese precio de mayoreo: $";
    cin>>myfloat;
    nuevoProducto.setPrecioMayoreoP(myfloat);

    cout<<"Ingrese precio de menudeo: $";
    cin>>myfloat;
    nuevoProducto.setPrecioMenudeoP(myfloat);

    cout<<"Ingrese existencia actual: ";
    cin>>myInt;
    nuevoProducto.setExistP(myInt);

    return nuevoProducto;
    }

void Menu::imprimirMenu() const {
    limpiarPantalla();
    cout << "\tMenu" << endl;
    cout << "1) Agregar Productos" << endl;
    cout << "2) Aumentar existencia ya registrados" << endl;
    cout << "3) Decrementar existencia ya registrados" << endl;
    cout << "4) Ver todos los productos registrados" << endl;
    cout << "5) Salir" << endl;
    cout << "Ingrese una opcion: ";
    }


void Menu::limpiarPantalla() const {
    system("cls");//Limpia pantalla
    }

void Menu::enterMenu() const {
    cout << "\nPresiona cualquier tecla para continuar ";
    getch();//  Detecta un teclazo
    }

Menu::Menu() {
    Coleccion nuevaColeccion;
    int opc(0);
    while (opc!=5) {
        int indice, cantidad;
        char opcionNuevoP;

        imprimirMenu();
        cin>>opc;

        switch(opc) {
            case 1:
                do {
```

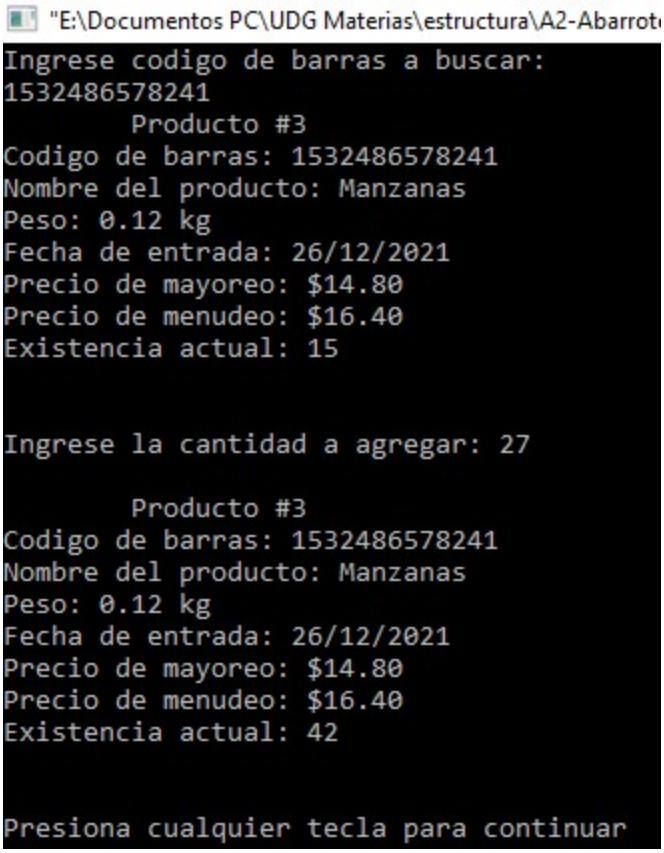
```
nuevaColeccion.agregarProducto(agregarProductoNuevo());
                    cout<<"Quiere agregar otro producto (S/N): "<<endl;
                    cin>> opcionNuevoP;
                    opcionNuevoP = toupper(opcionNuevoP);
                    }
                while(opcionNuevoP == 'S');

                if(nuevaColeccion.getIndexInventario() >= 500) {
                    cout<<"Ya no hay espacio en la memoria (arreglo
Inventario)"<<endl;
                    }
                break;

            case 2:
                indice = buscarIdProducto(nuevaColeccion);
                if (indice!=-5) {
                    cout<< nuevaColeccion.toString(indice);
                    cout<< "\nIngrese la cantidad a agregar: ";
                    cin>>cantidad;
                    cout<<endl;
                    nuevaColeccion.incrementaExistencia(indice,cantidad);
                    cout<<nuevaColeccion.toString(indice);
                    }
                else {
                    cout<< "NO EXISTE codigo de barras";
                    }
                enterMenu();
                break;

            case 3:
                indice = buscarIdProducto(nuevaColeccion);
                if (indice!=-5) {
                    cout<< nuevaColeccion.toString(indice);
                    cout<< "Ingrese la cantidad a quitar: ";
                    cin>>cantidad;
                    cout<<endl;
                    nuevaColeccion.decrementaExistencia(indice,cantidad);
                    cout<<nuevaColeccion.toString(indice);
                    }
                else {
                    cout<< "NO EXISTE codigo de barras";
                    }
                enterMenu();
                break;

            case 4:
                limpiarPantalla();
                cout<<endl;
                for(int i(0); i < nuevaColeccion.getIndexInventario();
i++) {
                    cout<<nuevaColeccion.toString(i);
                    }
                enterMenu();
                break;

            case 5:
```

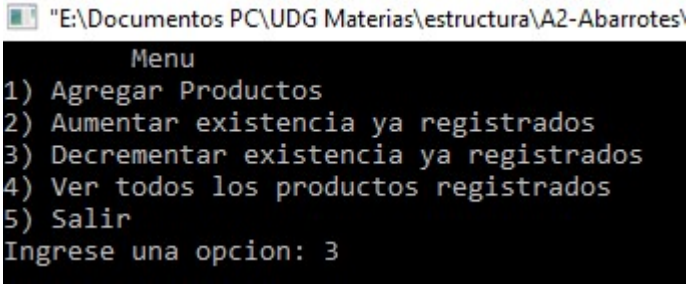
```
            limpiarPantalla();
            cout << "Fin del programa" << endl;
            break;

        default:
            limpiarPantalla();
            cout << "VALOR INVALIDO: Ingrese valor valida" << endl;
            enterMenu();
        }
    }
}
```

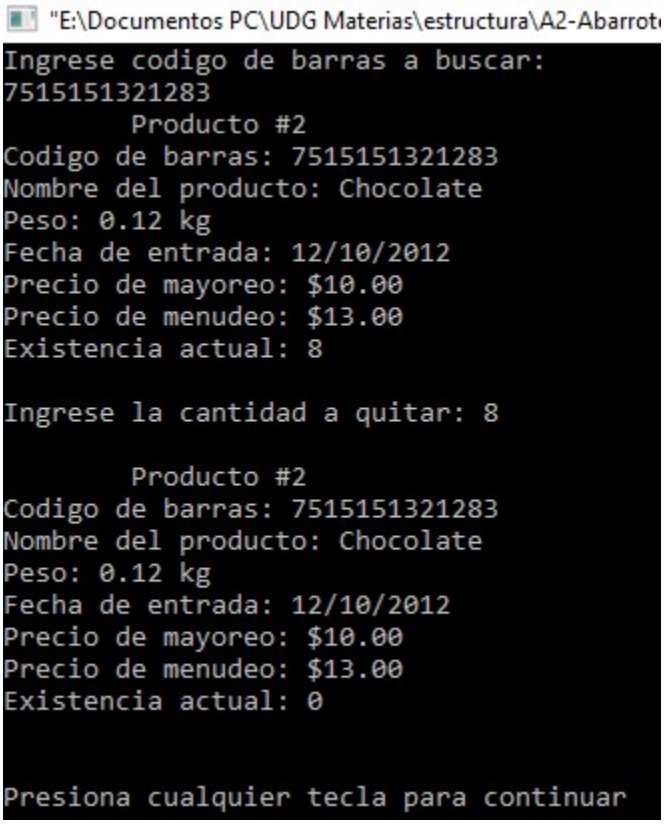


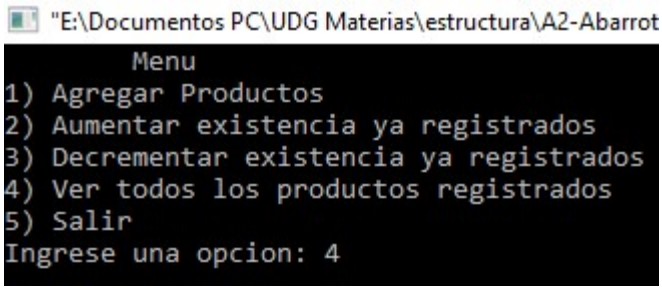


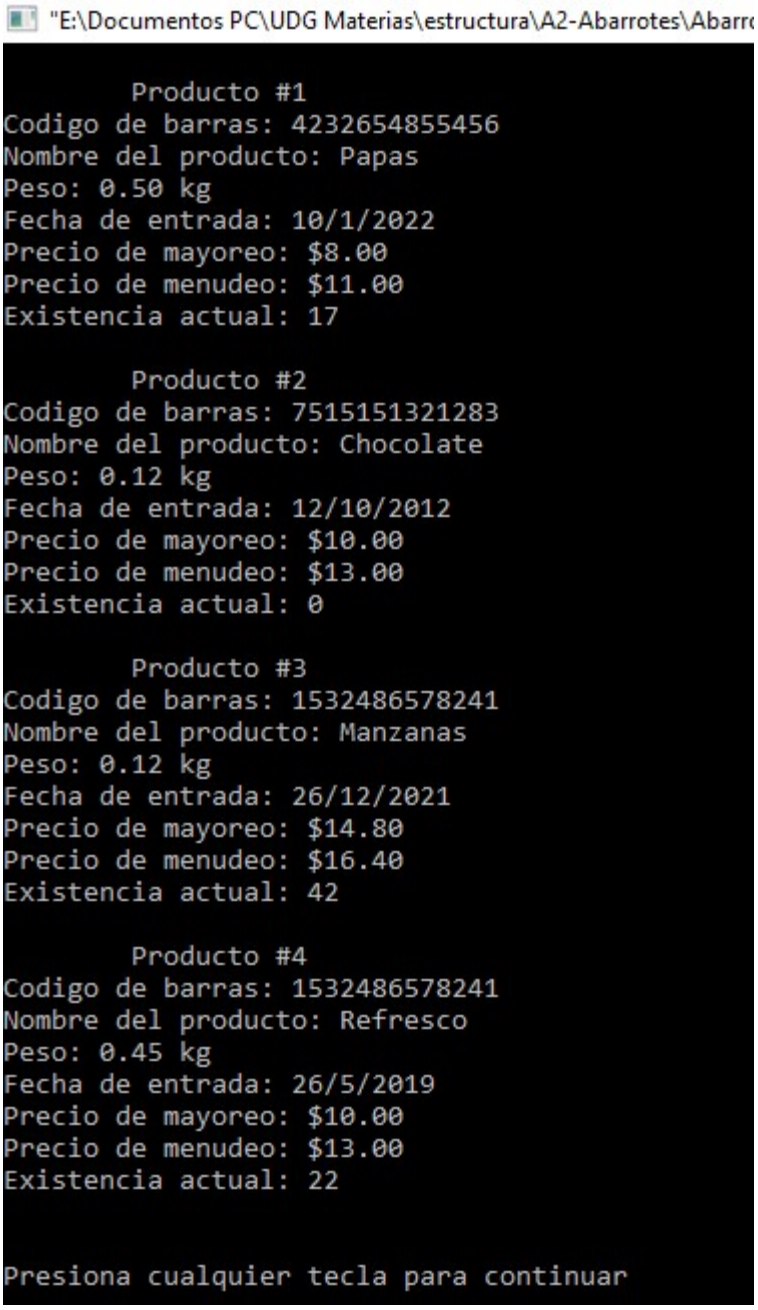


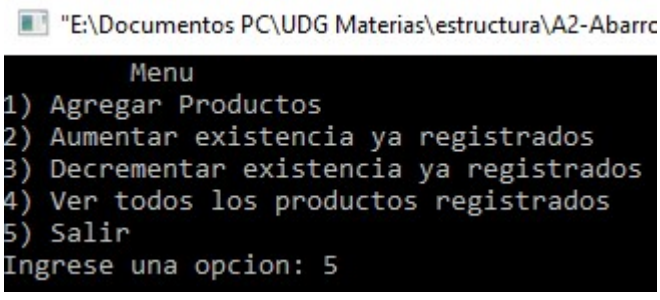

"E:\Documentos PC\UDG Materias\estructura\A2-Abarrot
Ingrese codigo de barras a buscar:
1532486578241
Producto #3
Codigo de barras: 1532486578241
Nombre del producto: Manzanas
Peso: 0.12 kg
Fecha de entrada: 26/12/2021
Precio de mayoreo: $14.80
Precio de menudeo: $16.40
Existencia actual: 15
Ingrese la cantidad a agregar: 27
Producto #3
Codigo de barras: 1532486578241
Nombre del producto: Manzanas
Peso: 0.12 kg
Fecha de entrada: 26/12/2021
Precio de mayoreo: $14.80
Precio de menudeo: $16.40
Existencia actual: 42
Presiona cualquier tecla para continuar

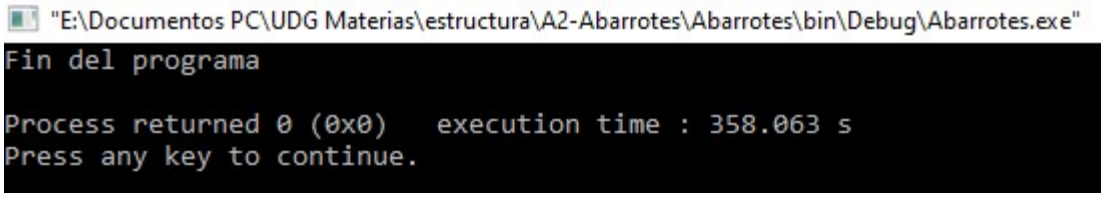

"E:\Documentos PC\UDG Materias\estructura\A2-Abarrotes\
Menu
1) Agregar Productos
2) Aumentar existencia ya registrados
3) Decrementar existencia ya registrados
4) Ver todos los productos registrados
5) Salir
Ingrese una opcion: 3


"E:\Documentos PC\UDG Materias\estructura\A2-Abarrot
Ingrese codigo de barras a buscar:
7515151321283
Producto #2
Codigo de barras: 7515151321283
Nombre del producto: Chocolate
Peso: 0.12 kg
Fecha de entrada: 12/10/2012
Precio de mayoreo: $10.00
Precio de menudeo: $13.00
Existencia actual: 8
Ingrese la cantidad a quitar: 8
Producto #2
Codigo de barras: 7515151321283
Nombre del producto: Chocolate
Peso: 0.12 kg
Fecha de entrada: 12/10/2012
Precio de mayoreo: $10.00
Precio de menudeo: $13.00
Existencia actual: 0
Presiona cualquier tecla para continuar


"E:\Documentos PC\UDG Materias\estructura\A2-Abarrot
Menu
1) Agregar Productos
2) Aumentar existencia ya registrados
3) Decrementar existencia ya registrados
4) Ver todos los productos registrados
5) Salir
Ingrese una opcion: 4


"E:\Documentos PC\UDG Materias\estructura\A2-Abarrotes\Abarr
Producto #1
Codigo de barras: 4232654855456
Nombre del producto: Papas
Peso: 0.50 kg
Fecha de entrada: 10/1/2022
Precio de mayoreo: $8.00
Precio de menudeo: $11.00
Existencia actual: 17
Producto #2
Codigo de barras: 7515151321283
Nombre del producto: Chocolate
Peso: 0.12 kg
Fecha de entrada: 12/10/2012
Precio de mayoreo: $10.00
Precio de menudeo: $13.00
Existencia actual: 0
Producto #3
Codigo de barras: 1532486578241
Nombre del producto: Manzanas
Peso: 0.12 kg
Fecha de entrada: 26/12/2021
Precio de mayoreo: $14.80
Precio de menudeo: $16.40
Existencia actual: 42
Producto #4
Codigo de barras: 1532486578241
Nombre del producto: Refresco
Peso: 0.45 kg
Fecha de entrada: 26/5/2019
Precio de mayoreo: $10.00
Precio de menudeo: $13.00
Existencia actual: 22
Presiona cualquier tecla para continuar


"E:\Documentos PC\UDG Materias\estructura\A2-Abarrc
Menu
1) Agregar Productos
2) Aumentar existencia ya registrados
3) Decrementar existencia ya registrados
4) Ver todos los productos registrados
5) Salir
Ingrese una opcion: 5



Conclusión:

En esta práctica fue difícil y tediosa de realizar ya que se tuvo que repetir muchos métodos para cada atributo a utilizar que serian los getter, setters y el método para pasarlos a una cadena. Al principio no sabía ni como empezar la actividad solo que tenia que definir las clases, con sus atributos y sus métodos ya mencionados, después de ver varios videos de apoyo de la plataforma y otros videos de internet y de mucha investigación en páginas y libros de POO pude desarrollar la actividad adecuadamente. Me llamo la atención que al momento de programar en POO al principio me costo mucho saber como empezar a programar debido a que no sabia que escribir, pero con la ayuda de las instrucciones y los videos pude empezar con los métodos principales de los getters y setters y toString, y después de programar los métodos para el almacenamiento y acceso de los datos, ya se volvió más sencillo de programar, ya que solo se usaban los métodos o funciones correspondientes para cada acción requerida de la actividad.