

## Guía 4: Aplicación de Clases.

### Objetivos

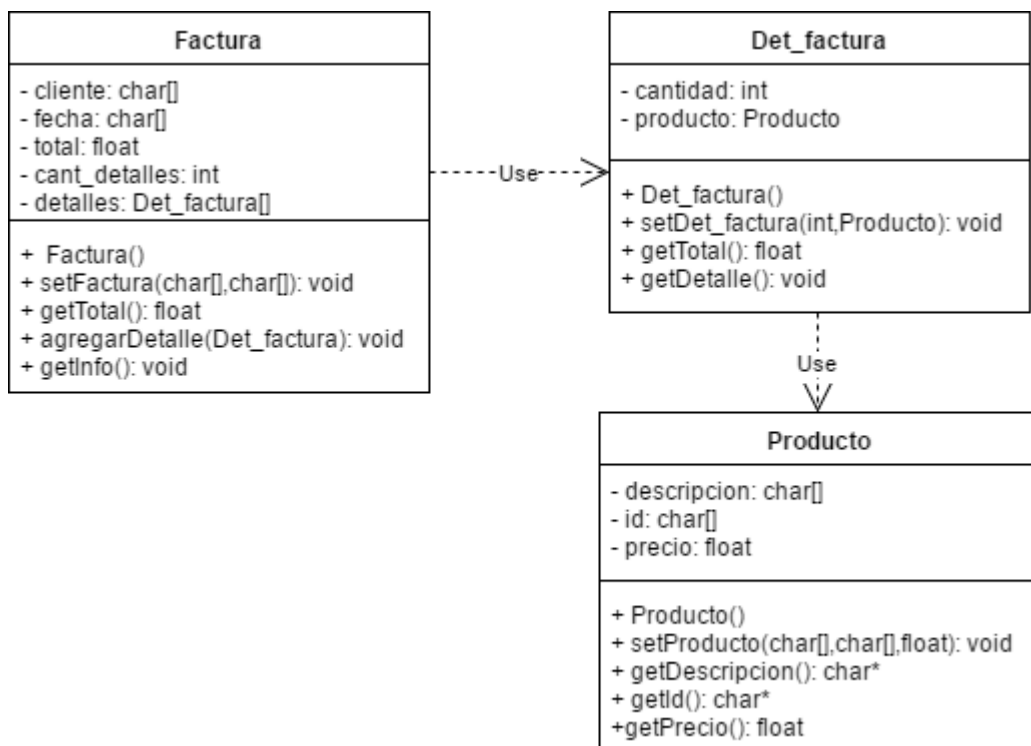
- Conocer las diferentes maneras de resolver problemas de programación usando clases y sus propiedades.
- Codificar clases utilizando diagramas UML con relación de asociación.

### Problema:

Una empresa que vende productos varios desea llevar una gestión ordenada y sistemática de sus ventas, por lo que ha pedido su colaboración para resolver el problema. El administrador del departamento le menciona que las ventas se realizan mediante **facturas**, y estas poseen **detalles de facturas**, a su vez estas contienen **productos**, también menciona que con 20 registros de facturas y detalles está bien para mostrarle el primer prototipo.

### Planteamiento de la solución:

Luego de comprender lo que se nos está pidiendo, se procede a identificar las clases que estarían involucradas como lo son:



# Programación II

Cada una de las clases tendrá una tarea específica que se muestra en el diagrama de clases, además de prototipos de funciones que contienen, con esto se procede a codificar:

## Implementación:

Crear un nuevo proyecto c++ en CodeBlock bajo el nombre de guía 4, luego codificar agregar las siguiente clases y su contenido.

### Producto.h:

```
1  #ifndef PRODUCTO_H_INCLUDED
2  #define PRODUCTO_H_INCLUDED
3  class Producto{
4  private:
5      char descripcion[TAM];
6      char id[TAM];
7      float precio;
8  public:
9      Producto(){
10         strcpy(id, "");
11         strcpy(descripcion, "");
12         precio=0;
13     }
14     void setProducto(char i[], char desc[], float pre){
15         strcpy(id, i);
16         strcpy(descripcion, desc);
17         precio=pre;
18     }
19     char* getDescripcion(){return descripcion;}
20     char* getId(){return id;}
21     float getPrecio(){return precio;}
22 };
23
24 #endif // PRODUCTO_H_INCLUDED
25
```

# Programación II

## Det\_factura.h :

```

1  #ifndef DET_FACTURA_H_INCLUDED
2  #define DET_FACTURA_H_INCLUDED
3
4  #include "Producto.h"
5
6  class Det_factura{
7  private:
8      int cantidad;
9      Producto producto;
10 public:
11     Det_factura(){
12         cantidad=0;
13     }
14     void setDet_factura(int cant,Producto prod){
15         cantidad=cant;
16         producto=prod;
17     }
18     float getTotal(){return producto.getPrecio()*cantidad;}
19     void getDetalle(){
20         cout << cantidad << " " << producto.getDescripcion() << " " << producto.getPrecio() << " c/u " << getTotal() << endl;
21     }
22 };
23
24 #endif // DET_FACTURA_H_INCLUDED
25

```

## Factura.h :

```

1  #ifndef FACTURA_H_INCLUDED
2  #define FACTURA_H_INCLUDED
3  #include "Det_factura.h"
4  class Factura{
5  private:
6      char fecha[TAM];
7      char cliente[TAM];
8      float total;
9      int cant_detalle;
10     Det_factura detalles[TAM];
11 public:
12     Factura(){
13         strcpy(fecha,"");
14         strcpy(cliente,"");
15         total=0;
16         cant_detalle=0;
17     }
18     void setFactura(char cli[],char fe[]){
19         strcpy(cliente,cli);
20         strcpy(fecha,fe);
21     }
22     void agregarDetalle(Det_factura detalle){
23         detalles[cant_detalle] = detalle;
24         total+=detalle.getTotal();
25     }
26     float getTotal(){return total;}

```

# Programación II

```

27
28 void getInfo(){
29     cout << "\n-----\n";
30     cout << "\t\tVenta de productos varios US0";
31     cout << "\nCliente : "<< cliente;
32     cout << "\nFecha   : "<< fecha;
33     cout << "\nDetalles : "<<endl;
34     //modificar aqui para mostrar la lista de detalles
35     cout << "TOTAL A PAGAR : " << getTotal();
36     cout << "\nGracias por su compra!";
37 }
38 };
39
40 #endif // FACTURA_H_INCLUDED
41

```

## Main.cpp:

```

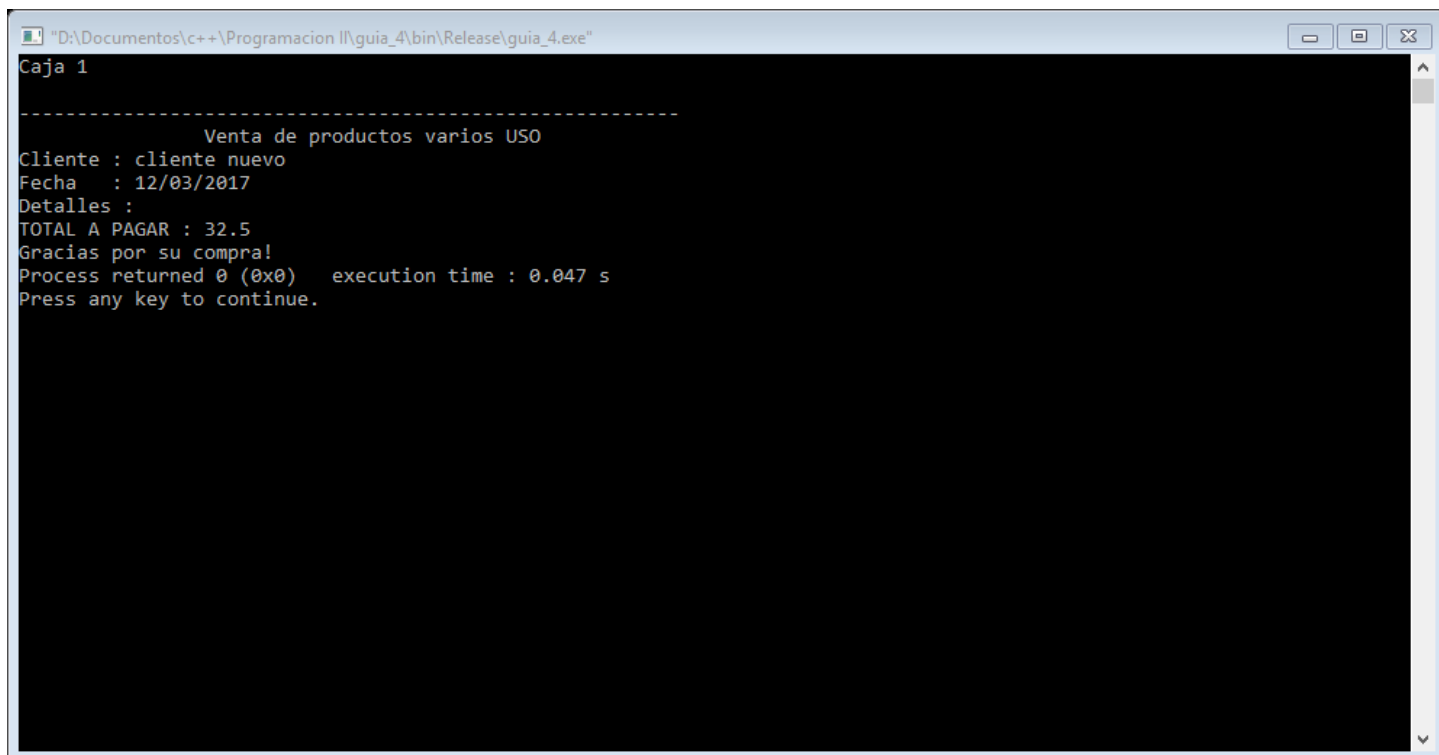
1  #include <iostream>
2  #include <cstring>
3  using namespace std;
4  #define TAM 20
5
6  #include "Factura.h"
7  #include "Det_factura.h"
8  #include "Producto.h"
9
10 Factura facturas[TAM];
11 Producto productos[TAM];
12 int cant_facturas=0;
13 int cant_productos=0;
14
15 void nuevaVenta(){
16     facturas[cant_facturas].setFactura("cliente nuevo","12/03/2017");
17     //agregando el detalle
18     Det_factura detalle;
19     detalle.setDet_factura(3,productos[0]);
20     facturas[cant_facturas].agregarDetalle(detalle);
21     detalle.setDet_factura(5,productos[1]);
22     facturas[cant_facturas].agregarDetalle(detalle);
23     detalle.setDet_factura(2,productos[2]);
24     facturas[cant_facturas].getInfo();
25     cant_facturas++;
26 }
27

```

# Programación II

```
28  int main()
29  {
30      cout << "Caja 1" << endl;
31      productos[cant_productos].setProducto("PROD1", "producto 1", 5);
32      cant_productos++;
33      productos[cant_productos].setProducto("PROD2", "producto 2", 3.5);
34      cant_productos++;
35      productos[cant_productos].setProducto("PROD3", "producto 3", 2);
36      cant_productos++;
37      productos[cant_productos].setProducto("PROD4", "producto 4", 10);
38      cant_productos++;
39      nuevaVenta();
40      return 0;
41  }
42
```

Una vez finalizada la codificación se compilara y se podrá observar el siguiente resultado:



```
"D:\Documentos\c++\Programacion II\guia_4\bin\Release\guia_4.exe"
Caja 1
-----
                Venta de productos varios USO
Cliente : cliente nuevo
Fecha   : 12/03/2017
Detalles :
TOTAL A PAGAR : 32.5
Gracias por su compra!
Process returned 0 (0x0)   execution time : 0.047 s
Press any key to continue.
```

# Programación II

## Ejercicios:

Realizar los siguientes ejercicios:

1. Implementar un menú con las siguientes funciones:
  - Nueva venta: se le pedirá los datos correspondientes: cliente, fecha, y los detalles de facturas, estos detalles se pedirán dentro de un while que se le preguntara si desea agregar otro detalle.
  - Ver la información de todas las facturas realizadas.
  - Agregar nuevos productos.
  - Salir.
2. Modificar la función **getInfo()** de la clase Factura para que muestre en un ciclo todos los detalles que estén en esa factura.