

Guía 2: Introducción al modelado de clases en UML.

Objetivos

- **Conocer el modelo de clases en UML.**
- **Diseñar modelos de clases utilizando UML.**
- **Aplicar el modelo de clases UML en código.**

¿Qué es UML?

UML proviene de las siglas en inglés: *Unified Modeling Language*, que en español es: **Lenguaje Unificado de Modelado**.

Básicamente es un modelo de representar sistemas de software, es uno de los más conocidos y utilizados, dado su sencillez y su rápida comprensión visual.

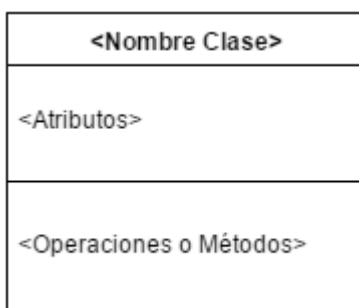
Diagramas de clases

Un diagrama de clases sirve para visualizar las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, de herencia, de uso y de contenimiento, y está compuesto por los siguientes elementos:

- **Clase:** Atributos, métodos y viabilidad.
- **Relaciones:** Herencia, Composición, Agregación, Asociación y Uso.

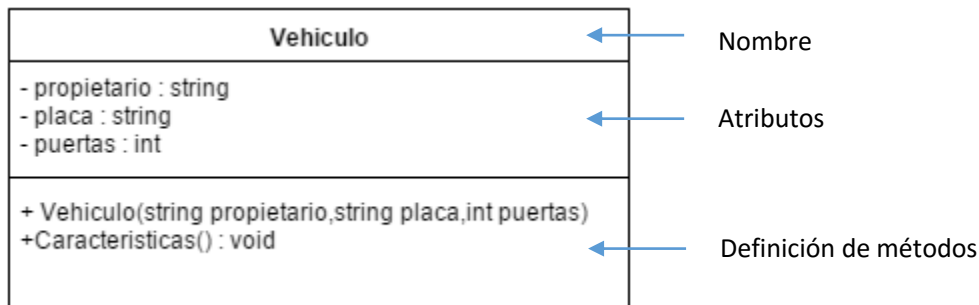
Clase:

Es la unidad básica que encapsula toda la información de un Objeto. En diagrama UML una clase está dividida en tres secciones:



Programación II

Ejemplo:



Identificamos las características de los atributos y métodos mediante lo siguiente:

- **Public (+):** Indica que el atributo será visible tanto dentro como fuera de la clase, es decir, es accesible desde cualquier parte.
- **Private (-):** Indica que el atributo sólo será accesible desde dentro de la clase (sólo sus métodos lo pueden acceder).
- **Protected (#):** Indica que el atributo no será accesible desde fuera de la clase, pero si podrá ser accedido por métodos de la clase además de las subclases que se deriven.

Clases en C++:

Teniendo en cuenta el diseño de clases en UML se procede a la codificación de clases.

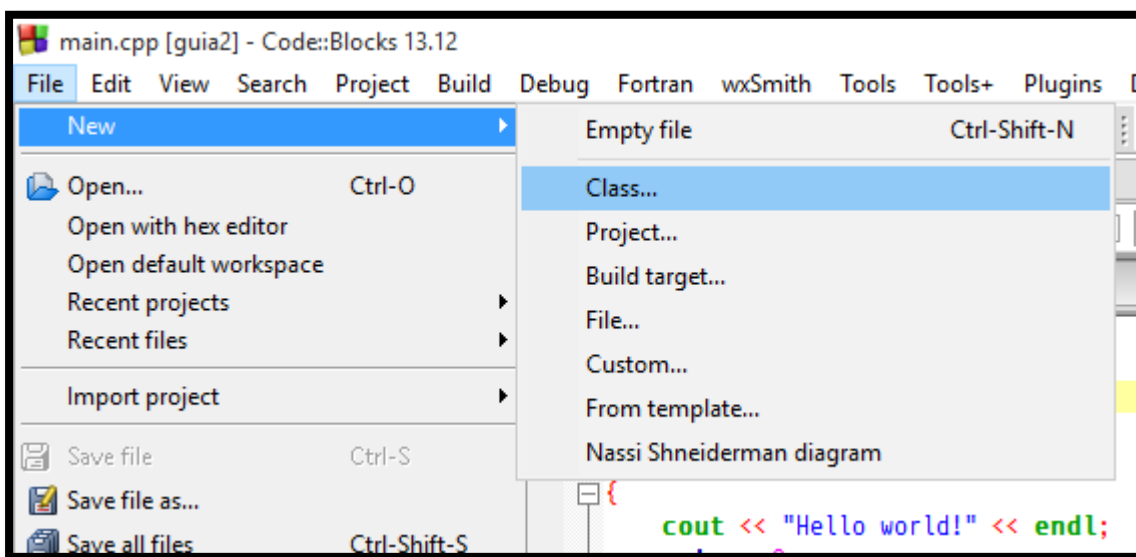
Paso 1:

Ejecutar nuestra ide codeblock, luego crear un nuevo proyecto en consola utilizando c++, con el nombre de guía 2.

Programación II

Paso 2:

Seleccionamos File -> New -> Class...



Paso 3:

Escribimos el nombre de nuestra clase, en este caso “Vehiculo” y deseccionamos *generate implementation file*, esto sirve para generar el archivo de implementación de la clase y todo el contenido de los método o funciones de las clases, pero dado que estamos generando la clase solo con fines demostrativos omitiremos este paso.

Programación II

Create new class

Class definition

Class name:

Arguments:

☒ Has destructor ☐ Has copy ctor
☒ Virtual destructor ☐ Has assignment op.

Inheritance

☐ Inherits another class

Ancestor:

Ancestor's include filename:

Scope:

Member variables

Add new:

☒ Add "Getter" method
☒ Add "Setter" method
☒ Remove prefix:

Documentation

☐ Add documentation where appropriate

File policy

☒ Add paths to project ☒ Use relative path

☐ Header and implementation file shall be in same folder

Folder:

☐ Header and implementation file shall always be lower case

Header file

Folder:

Filename:

☒ Add guard block in header file

Guard block:

Implementation file

☐ Generate implementation file

Folder:

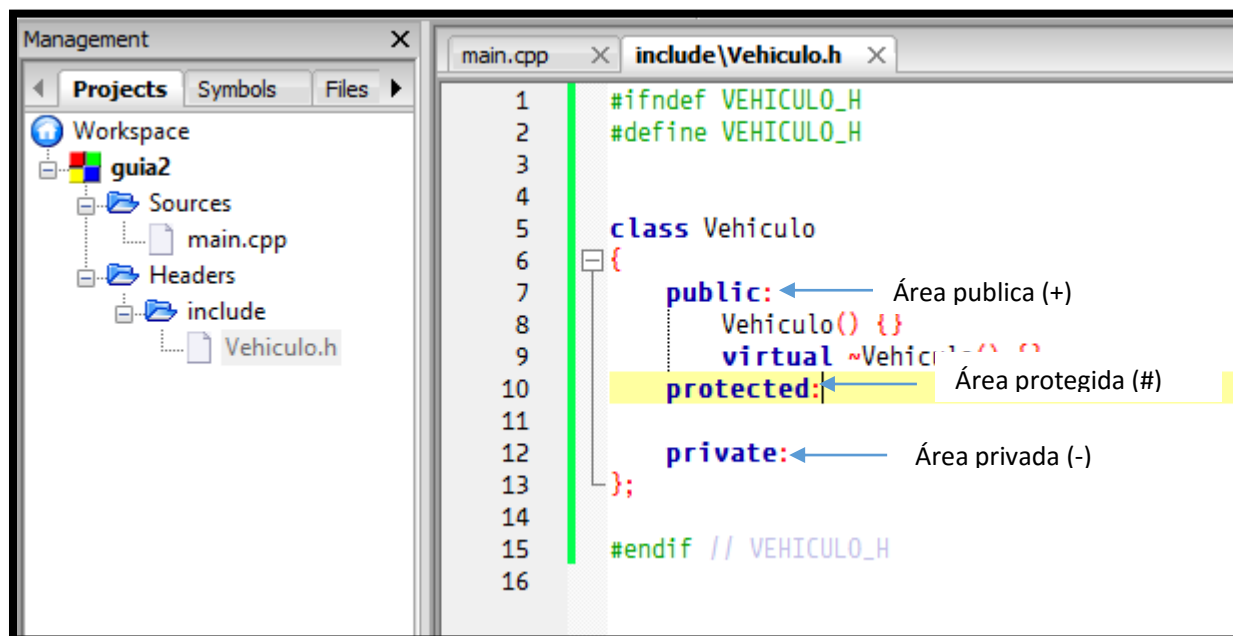
Filename:

Header include:

Luego seleccionamos create. Y nos pregunta si añadimos la clase al proyecto, Seleccionamos ok.

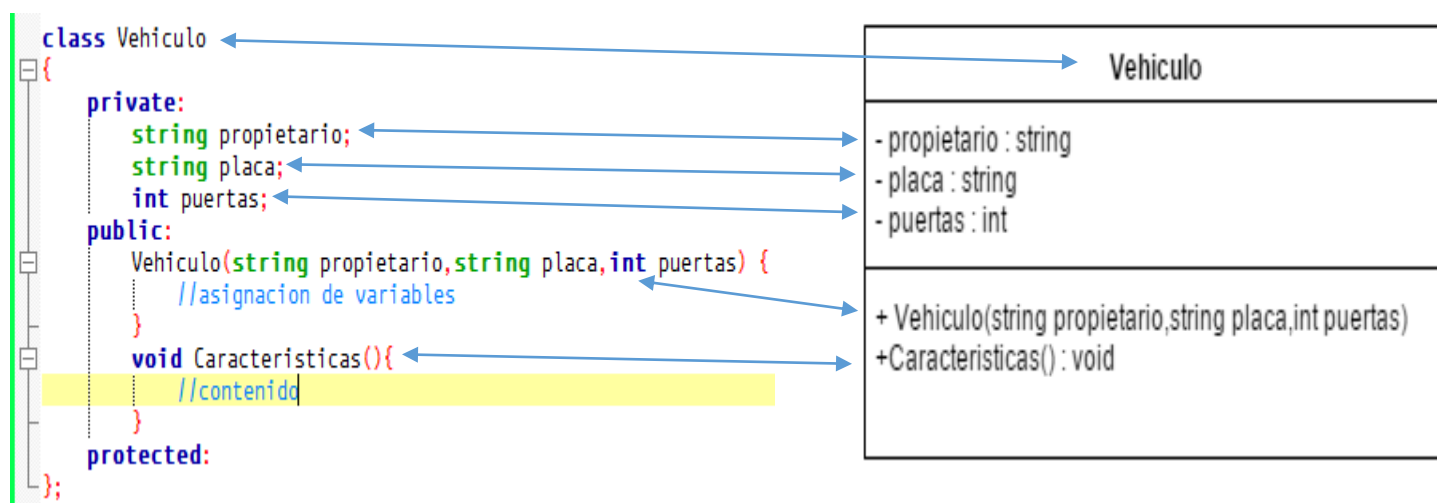
Programación II

Como resultado tenemos una plantilla básica de nuestra clase “Vehiculo” la cual tiene área publica, protegida y privada. Así como el constructor y destructor de dicha clase.



Paso 4:

Ahora que tenemos nuestra plantilla de la clase trasladaremos el contenido del diagrama UML de “Vehiculo” a código siguiendo los lineamientos establecidos:



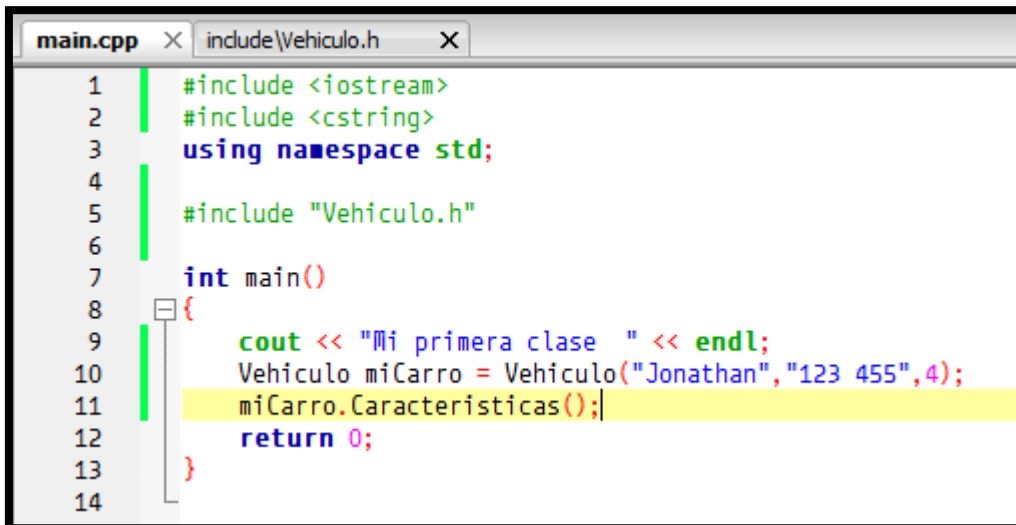
Programación II

Para finalizar completamos los métodos que se nos presentan:

```
2  #define VEHICULO_H
3
4  class Vehiculo
5  {
6  private:
7      string propietario;
8      string placa;
9      int puertas;
10 public:
11     Vehiculo(string propietario,string placa,int puertas) {
12         //asignacion de variables
13         this->propietario = propietario;
14         this->placa = placa;
15         this->puertas = puertas;
16     }
17     void Caracteristicas(){
18         cout << "Vehiculo de : "<< this->propietario << " Placa : "<< this->placa << " puertas : "<< this->puertas<<endl;
19     }
20 protected:
21 };
22
```

Paso 5:

Una vez finalizada la construcción de la clase procedemos a incluirla en el archivo main para que podamos usarla. Y dado que estamos usando el tipo string es necesario importar la librería.



```
main.cpp X include/Vehiculo.h X
1  #include <iostream>
2  #include <cstring>
3  using namespace std;
4
5  #include "Vehiculo.h"
6
7  int main()
8  {
9      cout << "Mi primera clase " << endl;
10     Vehiculo miCarro = Vehiculo("Jonathan","123 455",4);
11     miCarro.Caracteristicas();
12     return 0;
13 }
14
```

Programación II

Ejercicios

- Dado los siguientes diagramas de clase, generar la clase correspondiente en c++, siguiendo los pasos presentados en la guía :

Casa
- propietario : string - direccion : string - habitaciones: int
+ Casa(string propietario,string direccion,int habitaciones) +Caracteristicas() : void

Persona
- nombre: string - apellido: string - edad: int
+ Persona(string nombre,string apellido,int edad) + getNombres() : string + getEdad() : int

Cuenta
- propietario: string - balance: float = 0.0 - idCuenta:int
+ Cuenta(string propietario,int idCuenta) + getID() : int + getBalance() : float + getPropietario() : string + depositar(float cantidad) : void + retirar (float cantidad) : void

Programación II

- Dada la clase “Punto” generar el diagrama UML correspondiente, puede usar la página <https://www.draw.io/> y exportarla como imagen.

```
4
5  class Punto
6  {
7      private:
8          char id;
9          float x;
10         float y;
11         float z;
12     public:
13         Punto(char id){
14             this->id = id;
15         }
16         Punto(char id,float x,float y,float z) {
17             this->id = id;
18             this->x = x;
19             this->y = y;
20             this->z = z;
21         }
22         float getX(){
23             return x;
24         }
25         float getY(){
26             return y;
27         }
28         float getZ(){
29             return z;
30         }
31     };
32
```