**2.1 Declaración de clases: atributos, métodos, encapsulamiento.**

**Declaracion de clases**

las clases y estructuras son basicamente un conjunto de datos y comportamientos que forman en conjunto una unidad logica.

una declaración de clase se utiliza para crear instancias u objetos en tiempo de ejecucion .

una clase se crea un objeto que se guarda en la clase cuando se crea una nuevo objeto hace referencia al original

la clases se declaran mediante la palabra clave class seguida por un identificador unico,

ejemplo: public class.

atributos

es donde se define la variable

metodos

es un bloque de codigo que contiene una serie de instrucciones

acceso a metodos

es donde se llama un metodo en un objeto , nombre del objeto ,se agrega un punto, el nombre del metodo, y parentesis.= moto.startEngine();

parametros de metodos frente a argumentos

es donde se define los nombres de los metodos, se proporciona valores , los argumentos deben ser compatibles con el parametro y el nombre del argumento no deve ser el mismo que el parametro

pasar por referencia frente a pasar por valor

cuando un tipo de valor se pasa a un metodo se pasa una copia en lugar del propio objeto.los cambios realizados en el argumento no tiene nigun efecto en la copia original del metodo de llamada .

SampleRefType rt = new SampleRefType();  
 rt.value = 44;  
 ModifyObject(rt);

valores devueltos

si antes del nombre del metodo no es void puede devolver el valor mediante la clave return .

metodos asincronicos

cuando se utiliza el asintricronica , no necesita usar definiciones de llamada explicitas ni dividir manual mente el codigo en varios metodos

private async Task DoSomethingAsync()  
{  
 Task<int> delayTask = DelayAsync();  
 int result = await delayTask;  
  
 // The previous two statements may be combined into  
 // the following statement.  
 //int result = await DelayAsync();  
  
 Debug.WriteLine("Result: " + result);

Un método aisncrónico no puede declarar ningún parámetro [ref](https://docs.microsoft.com/es-es/dotnet/csharp/language-reference/keywords/ref) u [out](https://docs.microsoft.com/es-es/dotnet/csharp/language-reference/keywords/out-parameter-modifier) , pero puede llamar a los métodos que tienen estos parámetros.

definicion de cuerpos de expresion

hace devoluciones de forma inmediata con el resultado de una expresion

no usa get

iterators

realiza una coleccion ,como una lista o matriz , =yield return para devolver cada elemento de uno a uno

**encapsulacion**

en una clase o una estructura pueden especificar hasta qué punto se puede acceder a sus miembros para codificar fuera de la clase o la estructura. No se prevé el uso de los métodos y las variables fuera de la clase, o el ensamblado puede ocultarse para limitar el potencial de los errores de codificación o de los ataques malintencionados.

**2.2 Instanciación de una clase.**

new operador

se usa para crear objetos e invocar constructores

new modificador

se usa para ocultar un miembro heredado de un miembro base

new restriccion

se usa para restringir tipos que pueden usarse como argumentos para un parametro de tipo de declaracion generica

**2.4 Métodos: declaración, mensajes, paso de parámetros, retorno de valores.**

**declaracion**

declarados para un método sin in, ref o out se pasan al método llamado por valor. Ese valor se puede cambiar en el método, pero el cambio se perderá cuando se devuelva el control al procedimiento que ha realizado la llamada. Si usa palabras clave de parámetros de método en la declaración del parámetro, puede modificar este comportamiento.

pasos

Mediante el uso de la palabra clave params, puede especificar un parámetro de método que toma un número variable de argumentos.  
Puede enviar una lista separada por comas de argumentos del tipo especificado en la declaración de parámetro o una matriz de argumentos del tipo especificado. También puede no enviar ningún argumento. Si no envía ningún argumento, la longitud de la lista params es cero.

retorno de valores

para retornar valores se utiliza el in, ref y out, esto parametros tiene sus propias caracteristicas

in especifica que este parámetro se pasa por referencia, pero solo se lee mediante el método llamado.  
ref especifica que este parámetro se pasa por referencia y puede ser leído o escrito por el método llamado.  
out especifica que este parámetro se pasa por referencia y se escribe mediante el método llamado

**2.5 Constructores y destructores: declaración, uso y aplicaciones.**

constructores

declaracion

public class Taxi  
{  
 public bool isInitialized;  
 public Taxi()  
 {  
 isInitialized = true;  
 }  
}  
  
class TestTaxi  
{  
 static void Main()  
 {  
 Taxi t = new Taxi();  
 Console.WriteLine(t.isInitialized);  
 }  
}

uso

son métodos de clase que se ejecutan cuando se crea un objeto de un tipo determinado, tienen el mismo nombre que la clase y, normalmente, inicializan los miembros de datos del nuevo objeto.

no toma ningún parámetro se denomina constructor predeterminado. Los constructores predeterminados se invocan cada vez que se crea una instancia de un objeto mediante el operador new y no se proporciona ningún argumento a new.

aplicacion

Los constructores se pueden marcar como public, private, protected, internal o protected internal. Estos modificadores de acceso definen cómo los usuarios de la clase pueden construir la clase.

destructores

uso

Los destructores se utilizan para destruir instancias de clases.  
  
declaracion

class Car  
{  
 ~Car() // destructor  
 {  
 // cleanup statements...  
 }  
}

aplicacion

cuando la aplicación encapsule recursos no administrados como ventanas, archivos y conexiones de red, debería utilizar destructores para liberar dichos recursos. Cuando el objeto cumple los requisitos para su destrucción, el recolector de elementos no utilizados ejecuta el método Finalize del objeto.

uso

es para eliminar objetos que ya no necesites o ya cumplieron su función y ya no tienen utilidad

2.7 sobrecarga de operaciones

La sobrecarga de operadores es uno de los mecanismos que nos permite ampliar las capacidades de los lenguajes de programación orientados a objetos. En C++, la declaración y definición de una sobrecarga de operador es muy similar a la declaración y definición de una función cualquiera. El ejemplo más sencillo de una sobrecarga de operadores nos lo da el lenguaje mismo, es decir, en una operación aritmética (por ejemplo, una suma ) el compilador determina el tipo de operación requerida de acuerdo con el tipo de datos involucrados. Vamos a suponer que se tienen las variables: int A, B; double X, Y; int R;

Sirve para realizar operaciones matematicas

Operador unarios

operadores unitarios, que son aquellos que sólo requieren un operando, como la asignación o el incremento.

Cuando se sobrecargan operadores unitarios en una clase el operando es el propio objeto de la clase donde se define el operador. Por lo tanto los operadores unitarios dentro de las clases no requieren parámetros

opetadores binarios

son aquellos que requieren dos operandos, como la suma o la resta

Cuando se sobrecargan operadores en el interior se asume que el primer operando es el propio objeto de la clase donde se define el operador. Debido a esto, sólo se necesita especificar un operando.

Sintaxis:

<tipo> operator<operador binario>(<tipo> <identificador>);

Normalmente el <tipo> es la clase para la que estamos sobrecargando el operador, tanto en el valor de retorno como en el parámetro.