Fundamentos

Programación orienta a Objetos

- En un lenguaje de procedimientos, no hay ningún enlace entre los datos y los procedimientos que lo manejan. Por el contrario en un lenguaje orientada a objetos agrupamos los datos con el código que lo manejan. Las CLASES son la representación simbólica de los OBJETOS.
- Las CLASES describen campos, propiedades, métodos y eventos.
- Haciendo una analogía con el trabajo de un ingeniero, la CLASE corresponde a un plano de un motor y el OBJETO serian los diferentes motores que se construyen a partir de ese plano, Crear un OBJETO a partir de una clase es lo que se llama INSTANCIAR.

Propiedades de la Programación Orienta a Objetos

- Encapsulación: Un grupo de propiedades, métodos y otros miembros relacionados se tratan como una sola unidad u objeto.
- Herencia: Describe la posibilidad de crear nuevas clases basadas en una clase existente.
- Polimorfismo: Puede tener múltiples clases que se pueden usar de manera intercambiable, aunque cada clase implementa las mismas propiedades o los mismos métodos de maneras diferentes.
- Abstracto: Indica que se esta modificando tiene una implementación faltante o incompleta.

Clases y objetos

- Las clases describen el tipo de los objetos, mientras que los objetos son instancias de clases que se pueden usar
- La acción de crear un objeto se denomina creación de instancias.

```
class TestClass
{
    // Methods, properties, fields, events, delegates
    // and nested classes go here.
}
```

Método abstracto

- Se puede utilizar con clases, métodos, propiedades, indizadores y eventos.
- Se usa **abstract** en una declaración de clase para indicar que una clase solo pretende ser una clase base de otras clases.
- Los miembros marcados como abstractos o incluidos en una clase abstracta deben implementarse por clases que se derivan (heredan) de la clase abstracta.

Estructuras

- Tiene la finalidad de encapsular pequeños grupos de variables relacionadas, como la base y altura de un rectángulo o características de un elemento.

```
struct SampleStruct
{
}
```

Miembros de las clases

- Propiedades y campos.
- Métodos.
- Sobrecarga de métodos.
- Constructores.
- Destructores.
- Eventos
- Clases Anidadas

Propiedades y campos

- Representan información que contiene un objeto.
- Los campos se parecen a las variables ya que se pueden leer y escribir directamente.
- Las propiedades tienen procedimientos get y set, que proporcionan un mayor control sobre la forma que se establecen o devuelven los valores.

```
Class SampleClass
{
    public string sampleField;
}
```

```
class SampleClass
{
    public int SampleProperty { get; set; }
}
```

Métodos y sobrecarga de métodos

- Es una acción que un objeto puede realizar.
- Una clase puede tener varias implementaciones o sobrecargas del mismo método que se diferencien en el numero de parámetros o de tipos de parámetros.

Syntax:

```
[modifiers] return_type method_identifier ([arguments]) {
  method_code_block
}
```

Métodos

- Llamar a un método desde dentro de la misma clase.
- Llamar a un método que esta en una clase diferente.
- Usan llamadas anidadas.

Métodos

- Variables locales
  - Se definen dentro del método.
  - Son privadas para el método.
  - Se destruyen al terminar el método.
- Variables Compartidas
  - Se definen en la clase.
  - Permiten compartir datos entre los métodos de la clase.
- Conflictos de ámbito
  - El compilador no avisa si existen conflictos entre nombres de las variables locales y de clases.

Parametros

- Todos los parámetros se pasan por valor al menos que indiquen los contrario.
- Los parámetros se declaran entre los paréntesis después del nombre del método y se definen el tipo y nombre de cada parámetro.
- Paso de parámetros:
  - Por valor.
  - Por referencia.
  - De salida.

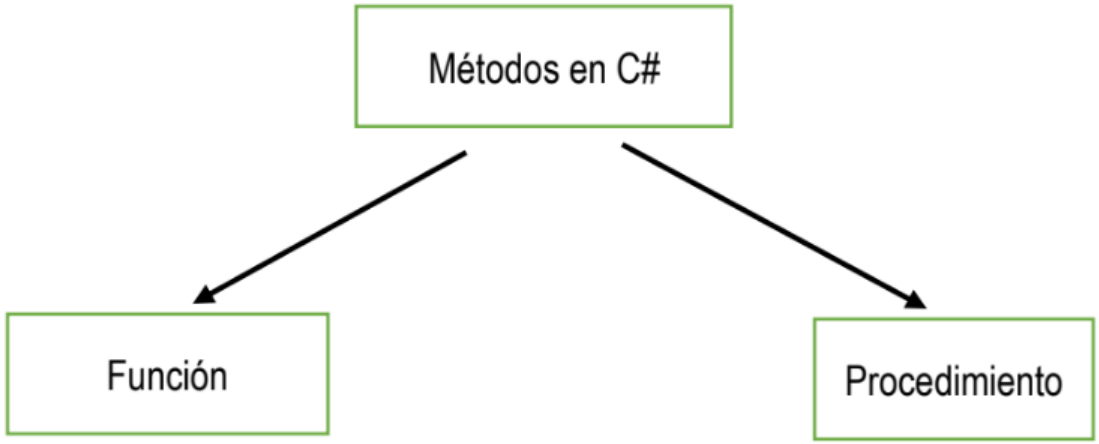
Paso de parámetros

- Por Valor:
  - Se copia el valor del parámetro.
  - Se puede cambiar el nombre de la variable dentro del método.
  - No afecta al valor fuera del método.
- Por Referencia:
  - Se pasa una referencia de la posición de memoria.
  - Se utiliza la palabra reservada **ref.**
  - Los tipos y valores de las variables deben de coincidir.
  - Los cambios hechos en el método afectan al llamador.
  - Se debe de asignar un valor al parámetro antes de la llamada del método (inicializarlo).

Paso de parámetros

- De Salida:
  - No sirven para pasar valores al método.
  - Solo devuelven resultados desde el método.
  - Se usa la palabra reservada **out.**
  - La variable no necesita estar inicializada.
  - Al parámetro de salida hay que asignarle siempre un valor.

Funciones y Operaciones



Retornan Valor

No Retornan Valor

Diferencias

- Una función sirve para describir una secuencia de paso u ordenes que hacen una tarea especifica de una aplicación mas grande.
- Un procedimiento realiza una tarea especifica y es relativamente independiente del resto del código.
- Los procedimientos suelen utilizarse para reducir la duplicación del código en los programas.

Constructores

- Son métodos de la clase que se ejecutan automáticamente cuando se crea un objeto de un tipo determinado.
- Se utilizan para inicializar los miembros de datos del nuevo objeto.
- Solo se ejecuta una vez cuando se crea la clase.
- Se pueden crear sobrecargas de la misma manera que un método

```
public class SampleClass
{
    public SampleClass()
    {
        // Add code here
    }
}
```

```
public class Person
{
   private string last;
   private string first;

   public Person(string lastName, string firstName)
   {
      last = lastName;
      first = firstName;
   }

   // Remaining implementation of Person class.
}
```

Destructores

- Se utilizan para destruir las instancias de las clases.
- No se pueden definir en estructuras. Solo se utilizan en clases.
- Una clase solo puede tener un destructor.
- No se pueden heredar ni sobrecargar.
- No se pueden llamar, solo se invocan automáticamente.
- No se permiten modificadores de acceso y no cuenta con parámetros.

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

Eventos

- Habilitan una clase u objetos para notificarlo a otras clases u objetos.
- La clase envía(o genera)el evento recibe el nombre del publicador y las clases que reciben (o controlador) del evento se denomina suscriptores.
- Para declarar un evento en una clase se utiliza la palabra reservada event.
- Para poder generar eventos se invocando los delegados de eventos.

Clases anidadas

- Una clase definida dentro de otra clase.
- La clase anidada por default es privada.
- Para crear una instancia de la clase anidada.

```
Container.Nested nestedInstance = new Container.Nested()
```

```
class Container
{
    class Nested
    {
        // Add code here.
    }
}
```