

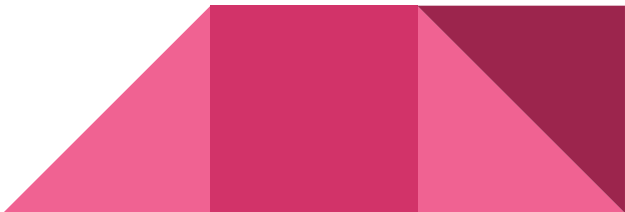
# Regularización: Programación

Amerike

Empezamos 3:05



# Temario 1

- Tipos de datos.
  - Métodos y funciones.
  - Clases y Objetos.
  - Propiedades.
  - Constructores y Destructores.
  - Polimorfismo.
  - Herencia.
  - Clases Abstractas.
  - Interfaces.
  - Encapsulamiento.
  - Modificadores de acceso.
  - Miembros estáticos.
  - Clases Genéricas.
  - Delegados.
- 

# Tipos de datos en C# (.NET vs Unity)

**int** (4 bytes): Almacena números enteros en el rango de -2,147,483,648 a 2,147,483,647.

**float** (4 bytes): Número decimal de precisión simple, hasta 7 dígitos de precisión.

**double** (8 bytes): Decimal con precisión doble, hasta 15 dígitos de precisión.

**bool** (1 byte): Almacena dos valores: true, false.

**char** (2 bytes): Almacena un carácter unicode.

**string** (4 bytes): Cadena de caracteres unicode.

**Vector2** (8 bytes): Par ordenado de valores float.

**Vector3** (12 bytes): Triada ordenada de valores float.

**Quaternion** (16 bytes): Representación de rotación en el espacio 3D.



# Métodos y funciones

Bloques de código para realizar una tarea en específico.

```
void  CalculaAreaCuadrado (float lado)
{
    var area = lado * lado;
}
```



# Métodos y funciones

Bloques de código para realizar una tarea en específico. Métodos no regresan un valor.

```
método { tipo acceso void parámetro CalculaAreaCuadrado (float lado)
        {
            var area = lado * lado;
        }
```

# Métodos y funciones

Bloques de código para realizar una tarea en específico. Función: regresa un tipo de dato al acabar la función.

```
función { tipo acceso float CalculaAreaCuadrado (parámetro (float lado)  
    {  
        return lado * lado;  
    }
```

# Tipos de acceso

**public:** Accesible desde fuera de la clase.

**private:** Accesible dentro de la clase.

**protected:** Accesible de la clase y sus clases derivadas.

**internal:** Accesible desde el mismo ensamblado.

**static:** Miembros de una clase pertenecientes a la misma clase.

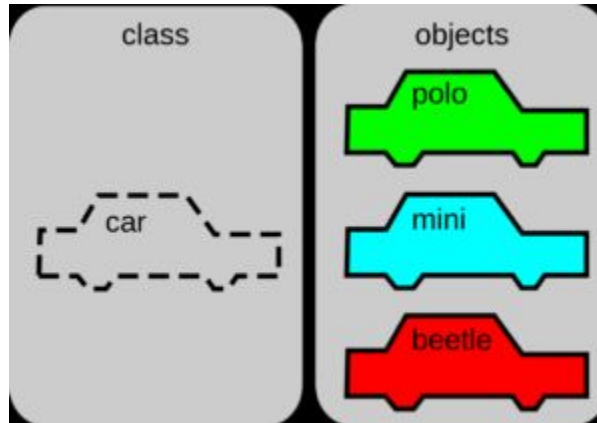
Y a todo esto, ¿qué es una clase?



# Clase

**Plantilla** que se usa para crear objetos. Cuenta con propiedades, métodos y funciones.

**Objeto:** Instancia de una clase





# Ejercicio:

Estamos realizando un juego aprovechando el hype de TLoZ TotK. En nuestro equipo nos han encargado la tarea de crear enemigos.

Empecemos creando una clase Enemy en donde definamos sus características.




# Constructor

Método que se utiliza para inicializar los campos de una clase al momento de su creación. Se llama automáticamente al crear un objeto.

```
public class Player
{
    public string Name;
    public int Health;

    public Player(string name, int health)
    {
        Name = name;
        Health = health;
    }
}
```



# Destructor

Método que se utiliza para liberar los recursos que ha adquirido una clase..

```
Player player1 = new Player("Jhon", 100);
```

```
player1 = null;
```



# Ejercicio:

Actualiza la clase Enemy para agregarle múltiples constructores.



# Polimorfismo

Capacidad de los objetos de tomar diferentes formas. Se puede realizar mediante herencia y sobrecarga de métodos.

```
public class Enemy
{
    ...
    public Enemy(string name)
    {
        Name = name;
    }


    public Enemy(string name, int health)
    {
        Name = name;
        Health = health;
    }
    ...
}
```

# Herencia

Creación de clases a partir de otras clases. Las clases creadas con herencia, se suelen llamar clase derivada o subclasses.

```
public class Moblin:Enemy  
{  
...  
}
```

Al heredar de una clase base, se pueden acceder a los campos y métodos públicos.

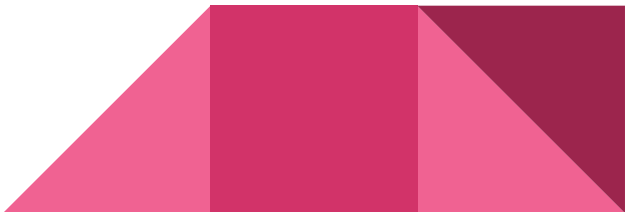


# Ejercicio

Crea enemigos de diferente categoría a partir de la clase base Enemy



# Temario 1 (Recapitulando)

- Tipos de datos.
  - Métodos y funciones.
  - Clases y Objetos.
  - Propiedades.
  - Constructores y Destructores.
  - Polimorfismo.
  - Herencia.
  - Clases Abstractas.
  - Interfaces.
  - Encapsulamiento.
  - Modificadores de acceso.
  - Miembros estáticos.
  - Clases Genéricas.
  - Delegados.
- 



# Futuro de las sesiones

1. Conceptos básicos (Tipos de datos - Herencia)
  - a. Ejercicios.
  - b. Tareas.
2. Estructuración de código (Resto Temario 1)
3. Temario 2
  - a. Ejercicios.
  - b. Tareas.



## Temario 2

- Patrones de diseño.
- Inyección de dependencias.
- Eventos.
- Scriptable Objects.
- Corrutinas.

## Temario 3

- Shaders.
- Uso de la GPU.
- Superficies y materiales.
- Shader Graph
- Comunicación con buffer.

