

★ OOP 04 ★
Nueva edición - 2020



PROGRA_AMANDO

holamundo.co

Agenda

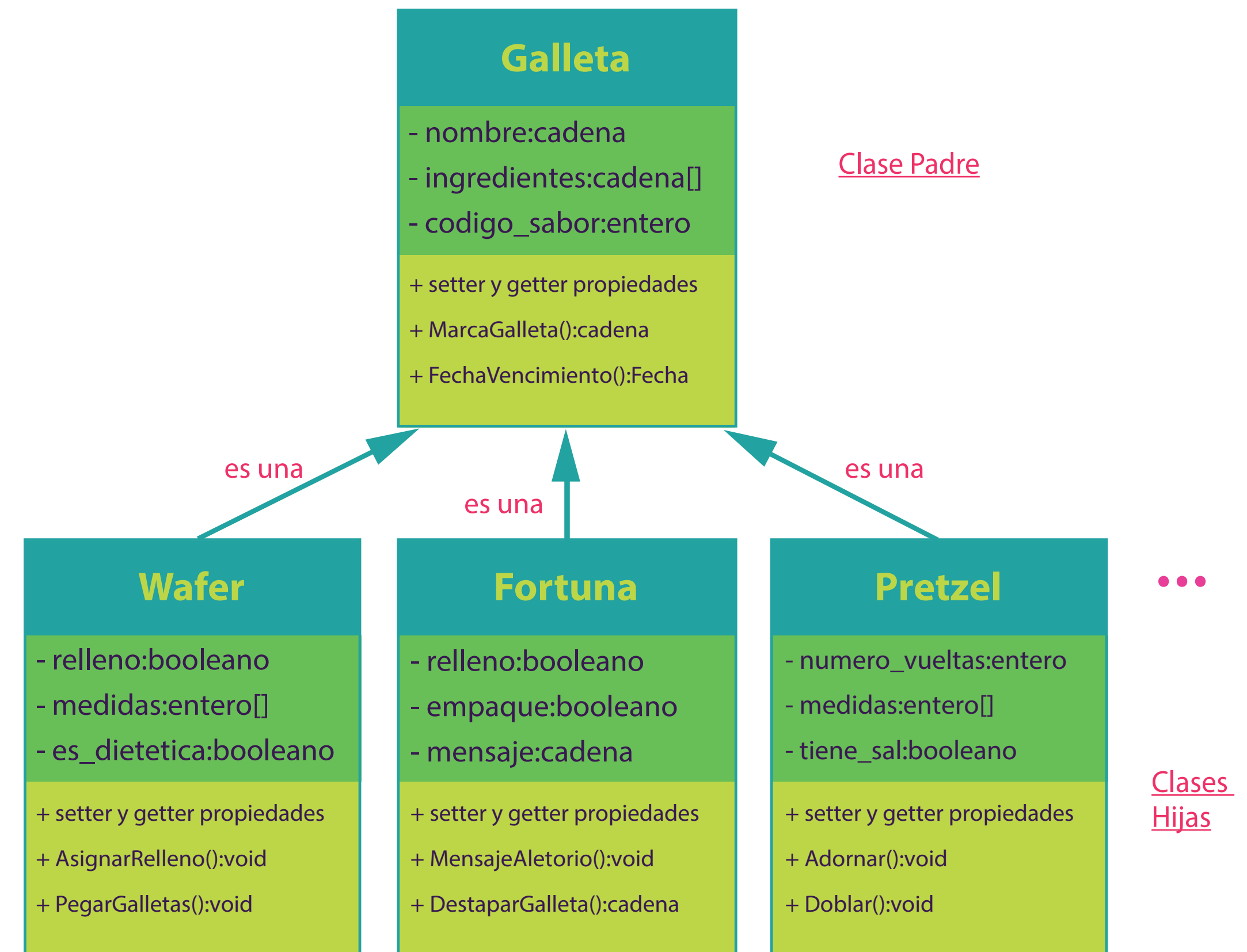
Herencia
Casting

Herencia

Con el fin de **especializar** las características de la **clase Galleta**, es posible crear una nueva clase que herede las propiedades y características de una clase superior llamada padre.

La clase hija **hereda** propiedades y métodos del padre y puede tener nuevas propiedades y métodos.

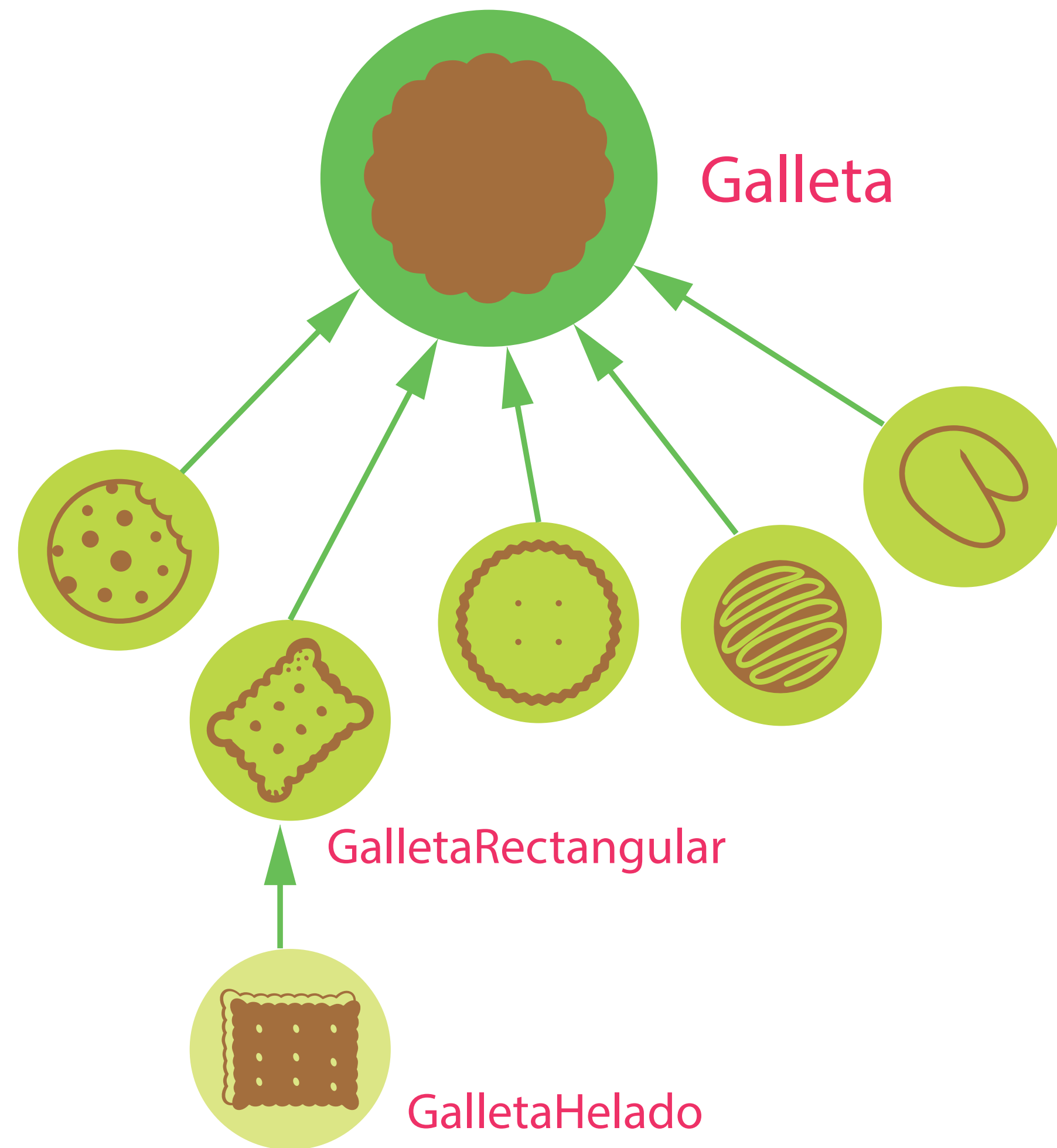
Se dice que la clase hija posee una relación de tipo “**es-un**” con la clase padre.



* La relación se lee: **Wafer es una Galleta**

Otras galletas, clases hijas: Helado, para perro, integrales, saladas, mini-galletas, vegana, artificial, etc.

Jerarquía de clases



Solo se puede heredar de **una sola clase** y múltiples clases pueden heredar de la misma clase.

Cada vez que se hereda de una clase, se busca **especializar** las propiedades y funcionalidades de la clase.

Se puede decir que:

La clase **GalletaHelado**, hereda de la clase **Galleta** y **GalletaRectangular**

Creación de la clase Padre

La clase padre no sufre ningún cambio en su implementación, es importante tener en cuenta identificar propiedades y acciones que deban estar en ella, para luego ser heredadas por las hijas.

```
class Galleta
{
    private string nombre;
    // ... resto de miembros dato


    public void FechaVencimiento()
    {
        Console.WriteLine("23/03/2021");
    }
    // ... resto de métodos

    public string Nombre { get => nombre; set => nombre = value; }
    // ... resto de setter y getters
}
```

Creación de la clase hija

La clase hija, se dice que extiende de la clase padre, para definir esta relación se usan **:(dos puntos)** como se muestra en el ejemplo.

```
class GalletaCuadrada:Galleta
{
    public void Congelar()
    {
        Console.WriteLine("Galleta congelada");
    }
}
```



Nuevo método
clase hija

Uso de propiedades

Al crear una clase hija, se heredan las propiedades y métodos de la clase padre, es posible acceder a esta información desde las clases hijas.

```
GalletaRectangular gr = new GalletaRectangular();
```

```
gr.Nombre = "Biscuit";
```

Se asigna el nombre, que es una propiedad heredada

```
gr.BordeOvalado(true);
```

El método BordeOvalado es una nueva acción, disponible solo en la clase hija

Uso de métodos

Las clases hijas definen acciones especializadas adicionales que no tienen la clase padre, de esta forma las hijas, seguirán haciendo una sola cosa bien, la clase padre es generalista y las hijas especialistas.

```
GalletaRectangular gr = new GalletaRectangular();
```

```
gr.Congelar();
```

Congelar es una acción nueva definida solo en la clase hija

```
gr.FechaDeVencimiento();
```

Acción heredada de la clase padre, que comparten las clases hijas. No se redefine

Manejo de referencias

Teniendo en cuenta la relación **“es un”** presente entre la clase padre y la clase hija, se pueden usar la clase padre como variable contenedora de sus hijas, la clase padre actua como una envoltura (wrapper)

```
Galleta galleta = new GalletaRectangular();
```

```
galleta.FechaVencimiento();
```

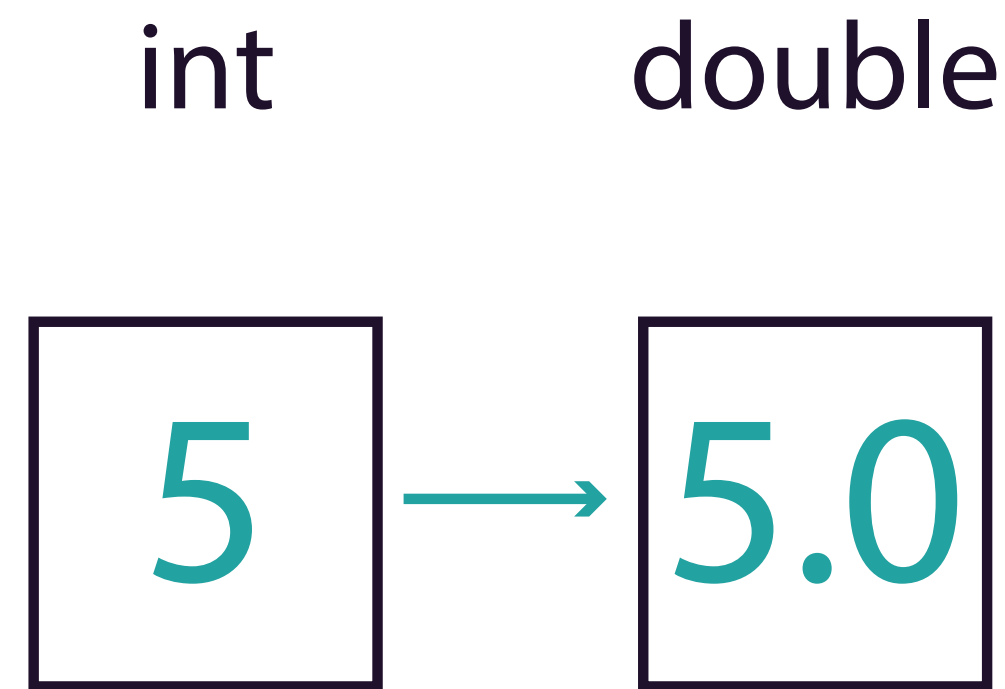
Se invoca una acción de la clase padre, que ha sido heredada

```
Console.WriteLine(galleta.Nombre);
```

Otra forma de acceder a los getters heredados

Casting

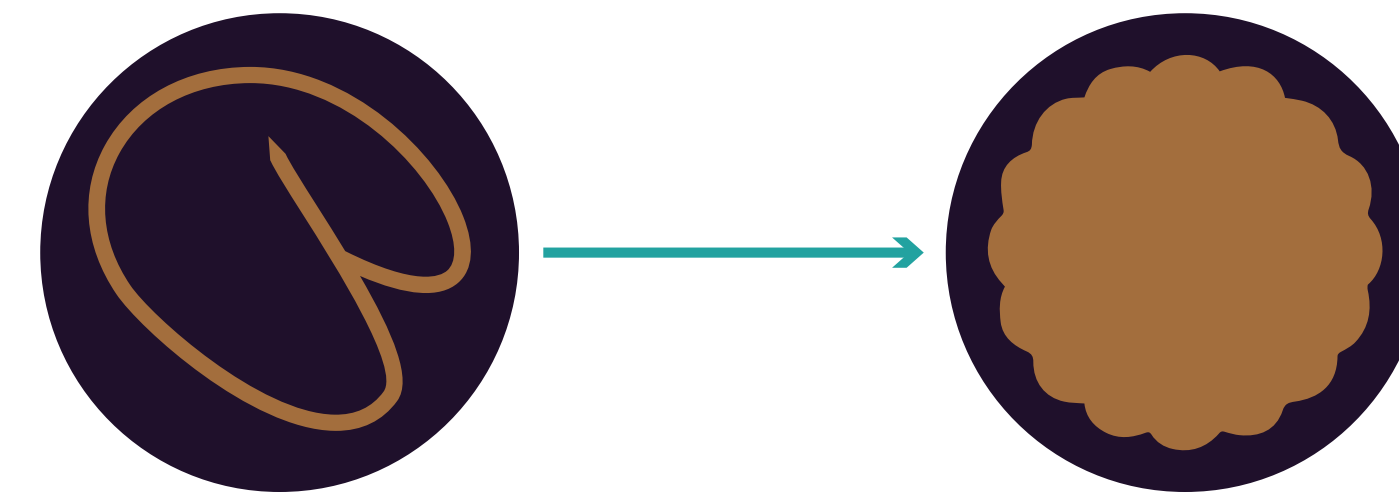
Se refiere a una operación de conversión entre dos tipos de datos diferentes pero que pueden llegar a ser compatibles porque comparten propiedades, acciones o pertenecen al mismo conjunto.



Si hablamos de **números**, el casting es el proceso de convertir un número en otro compatible

GalletaFortuna

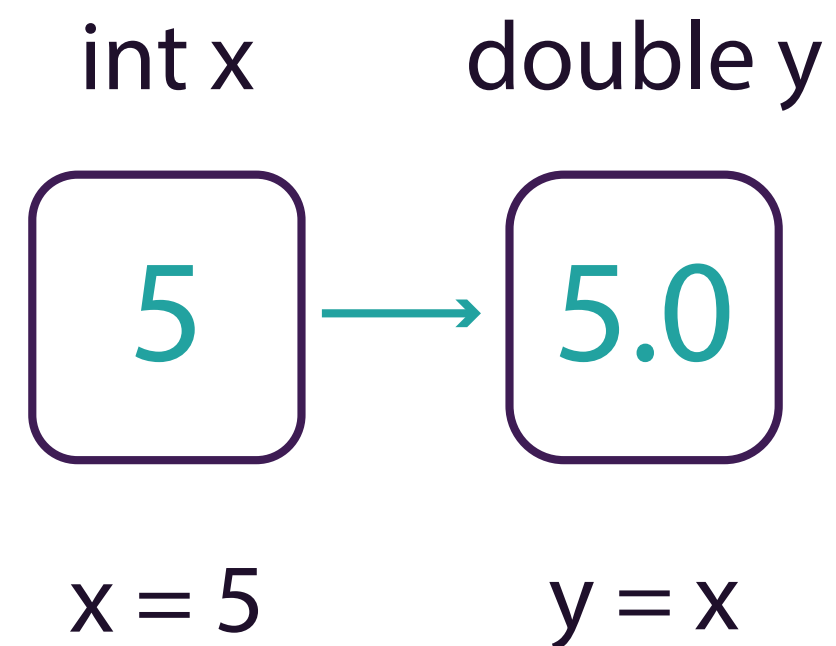
Galleta



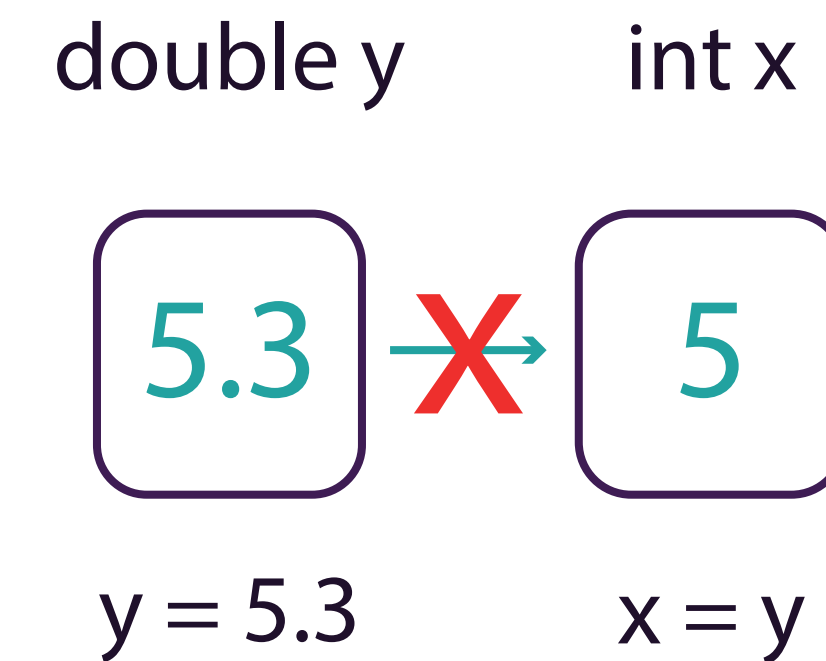
Si hablamos de **clases**, el casting es el proceso de convertir una clase a otra compatible. Si hay una relación de herencia, es posible realizar casting.

Casting implícito

Se realiza un casting implícito cuando la conversión entre datos se puede hacer de manera transparente, es decir sin que se afecten las operaciones que se vayan a realizar con los datos luego de la conversión



Este error, se detecta en tiempo de compilación



*Pérdida de datos
Error al realizar la conversión*

Casting explícito

Se presenta cuando la conversión se puede realizar porque hay dos tipos compatibles, pero no es obvia o puede traer problemas de precisión en las operaciones entre variables.

```
Galleta gr = new GalletaRectangular();
```

```
((GalletaCuadrada)gr).Congelar();
```

Se convierte el objeto a su tipo específico y luego se usa una acción.

```
int x = (int)Math.PI;
```

Pérdida de precisión, el casting autoriza la operación

Polimorfismo

Al comportamiento que asume la clase padre como wrapper contenedor, se le conoce como polimorfismo, ya que una referencia de un tipo padre, puede asumir el rol de cualquiera de sus hijas, esto hace que sea versátil y una forma de guardar una referencia que puede contener cualquier clase hija.

La referencia se convierte en la base que puede contener a sus hijas, las cuales son diferentes y especializadas, pero tienen similitudes, ya que extiende de un mismo padre.

```
Galleta[] hijas = {new GalletaRectangular(), new Oreo(), new Fortuna()};
```

El arreglo de tipo **Galleta**, puede contener galletas hijas de diferente tipo, las cuales tienen en común que son Galletas, por la relación de herencia.

Ejercicios

- Crear 5 clases diferentes que hereden de un concepto en común
- Crear una clase que tenga una herencia de varios niveles, por ejemplo GalletaHelado, hereda de GalletaCuadrada y Galleta cuadrada hereda de Galleta. GalletaHelado es nivel 3.
- ¿De cuantas clases puede heredar una clase hija?
- Crear un arreglo de una clase Padre, crear instancias de clases hijas y por medio de un ciclo, recorrer el arreglo e invocar un método en común.
- Crear el diagrama de clases de la clase Animal, Tiburon, Leon, Tortuga, Mico, Loro. Observe que hay animales terrestres, acuaticos y areos, evidence estas propiedades en el diagrama.
- En una jerarquía de 5 clases, cree una instancia de la última clase y la guarda en una referencia de la clase padre, luego por medio de casting explicito, llame métodos presentes en cada clase especifica.
- Cree tres clases hijas, con diferentes constructores que permitan inicializar miembros datos propios y heredados
- ¿Cual es el valor por defecto de los miembros datos de una clase hija? ¿Y el valor por defecto de miembros dato heredados?

