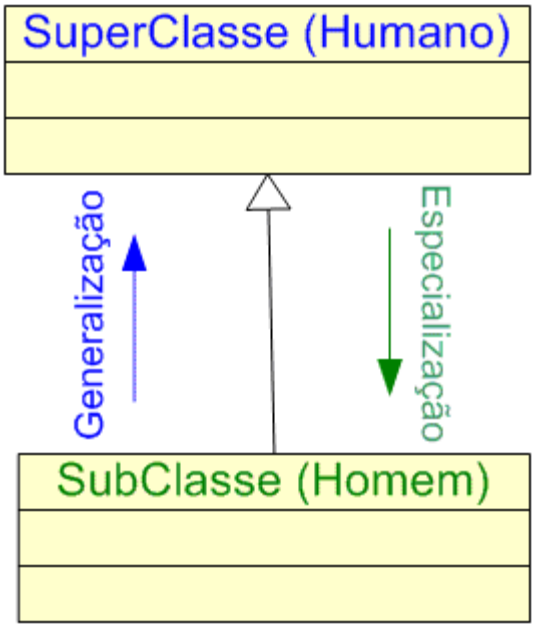
# POO

## Herança e Polimorfismo



## Herança e Polimorfismo

As classes são estruturadas hierarquicamente em superclasses, classes e subclasses por recurso a herança.

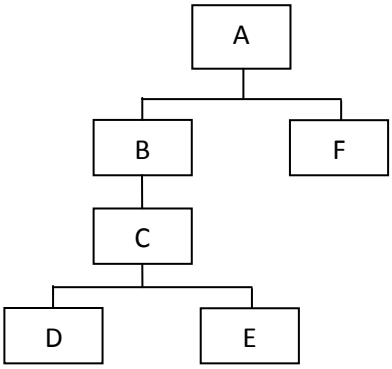
As classes

→ herdam as propriedades e os métodos das superclasses de que derivam,

→ acrescentam mais propriedades e métodos,

↓

passam tudo as subclasses que delas derivam.




O C# não permite que uma subclasse derive de mais do que uma superclasse.

A herança como veículo transmissor da especialização de subclasses é reforçada pelo polimorfismo. O polimorfismo é a transposição para a programação de polimorfia, que é a propriedade das substâncias que se apresentam em várias formas.

O polimorfismo permite ao programador definir métodos genéricos em classes, que podem ser implementadas de diversas formas nas diferentes subclasses. **O mesmo métodos pode, portanto, oferecer funcionalidades diferentes para cada classe que o implementa** (especialização).

### Derivação de uma subclasse

Uma subclasse deriva de uma classe, designada por classe base, herdando todas as propriedades e métodos desta.

```
public class Circulo: Centro
{
        //variáveis de instante
        //variáveis estáticas
        //construtores
        //acessores
        //métodos
}
```

**A subclasse Circulo deriva da classe base Centro.**

### Construtores de uma subclasse

Os construtores de uma subclasse constroem os objetos, colocando as suas propriedades – as especializadas definidas nessa subclasse e as genéricas que herdam da classe base – num estado inicial.

```
public class Circulo(int x, int y, double R): base(x,y)
{
        Raio = R;
}
```

Instanciação de um objeto Circulo com a variável de instante Raio, definida na subclasse Circulo e duas variaveis de instante herdadas da classe base.

### Variáveis de instante das classes e subclasses

As variáveis de instante e as variáveis estáticas da classe base podem ser acedidas por métodos das subclasses se forem definidas com o atributo ***protected***.

```
public class pai
{
    protected string NomePai;
        …
```

Variável de instante NomePai é definida como protected para poder ser manipulada pelas respetivas subclasses.

### Invocação de métodos da subclasse e da classe base

Os métodos de instante da classe base e das subclasses são invocados para os respetivos objetos.

```
//Moradia é uma subclasse de Propriedades
        Moradia M = new Moradia("J & P", 500900500, "Braga");
        Console.WriteLine("{0} Renda Mínima = {1}", M.ImpressaoProp(), M.RendaMinima());
```

Os métodos estáticos da classe base ou da subclasse são invocados para a classe que os contém.

```
//Moradia é uma subclasse de Propriedades
        Console.WriteLine(Moradias.MensagemM());
        Console.WriteLine(Propiedades.MensagemP());
```

### Destruidores

Os destruidores são métodos que removem os objectos. A destruição de um objeto tem de ser efetuada a partir do seu subobjecto com maior profundidade.

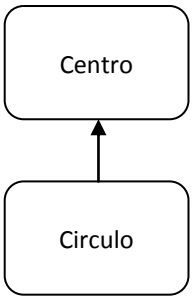
```
        ~Aluno()
        {
                Console.Writeline("Destrui aluno:{0}",Nome);
        }
```

A destruição de objectos pode ser seguida pela sua remoção física.

```
A=null;
System.GC.Collect();
```

Destruição do objeto A e invocação do Garbage Collection para forçar a sua remoção física.

Exemplo do código de programação da Classe Centro e a sua subclasse Circulo.




```
public class Centro
{
	private int x, y;

	public Centro()
	{
		x=0;
		y=0;
	}

	public Centro(int x, int y)
	{
		this.x=x;
		this.y=y;
	}

	public void CoodCentro()
	{
		Console.Writeline(“Centro = {0},{1}”, x,y);
	}
}

public class Circulo: Centro
{
	Private double Raio;

	public Circulo()
	{
	}

	public Circulo(int x, int y, double R):base(x,y)
	{
		Raio=R;
	}

	public double Area()
	{
		return Math.Round( Math.PI * Math.Pow(Raio,2) , 2);

	}
}
```