**Polymorphism && Inheritance**

**Constructors**

When you create an object that derivates from a base class (inheritance), first the constructor of the base class is called and then the constructor of the derivate class.

1º Base Class Constructor

2º Derivate Class Constructor

**Destructors**

Here first the destructor of the derivate class is called and then the destructor of the base class.

1º Derivate Class Destructor

2º Base Class Destructor

**Polymorphism**

**Problem** with **Inheritance** is that if you create a pointer of the base class, you can only access to the base class normal member functions. The derived class can access all the member functions of the derivate class and all the protected and public functions of the base class.

Using base class pointers is **useful**:

* To iterate a container filled with pointers to objects that derivates from the base class (you can have more than one different object that derivates from the base class).
* To pass a base class pointer to a function as a parameter

The **solution** to this → **Virtual functions** and is called as **Polymorphism**.

* Depending on the type of the object the pointer refers to, a different member function is called.
* If the base class function is virtual, then the derived function is automatically virtual.
  + Even so, it is better to still add the virtual keyword to all the derived functions to add readability and help resolving errors.
  + Also add the keyword override at the end of the derived function, this way checks if the function is being overriding (changing what the function does).
* **Limitation**: You cannot call the function of a derived object if it is not presented in the base class.
  + If this virtual base function makes no sense in the base class then we add the pure specifier, so the function converts to pure virtual function which has no definition.
  + Also the base class converts to an abstract class: You cannot create objects (instantiate) of this class anymore.
  + In this case, derived classes must be virtual and override