**Virtual Table (vtable)**

* **What it is**: The vtable is a lookup table that holds pointers to virtual functions of a class. Each class with virtual functions has its own vtable.
* **How it works**: When a class defines a virtual function, a vtable is created for that class. This table stores the addresses of the virtual functions that the class defines or inherits. If a derived class overrides a virtual function, the vtable in the derived class will contain the address of the overriding function, rather than the base class function.

**When vtable is created and when not created**

**1)No Virtual Function in Child Class:** If the derived class does not have any virtual functions of its own and does not override any virtual functions from its base class, the derived class does not need to create its own vtable.Instead, the derived class will simply inherit the vtable from the base class.

**Shared vtable:** In this case, objects of the derived class will use the vtable of the base class. The vptr in the derived class's objects will point to the base class's vtable.

**2)When vtable is Created in Child Class:** A derived class will create its own vtable only if it either:

**a)Declares a new virtual function.**

**b)Overrides a virtual function from the base class.**

**Virtual Pointer (vptr)**

* **What it is:** The vptr is a hidden pointer within each object of a class with virtual functions. This pointer points to the vtable of the object's class.
* **How it works:** When an object of a class is created, the vptr in the object is automatically set to point to the class's vtable. During function calls, the vptr is used to access the vtable and determine the correct function to invoke.

**Role in Function Overriding**

* When a virtual function is called on an object, the compiler uses the vptr to find the appropriate vtable, and then looks up the function pointer in the vtable to invoke the correct function.
* If a derived class overrides a virtual function, the vtable in the derived class will store the address of the overriding function. When the function is called on an object of the derived class, the vptr will point to the derived class's vtable, ensuring that the overriding function is called.
* This mechanism allows for dynamic (runtime) polymorphism, where the correct function is chosen based on the actual object type, even if the function is called through a pointer or reference to the base class.

**NOTE:**

**The virtual pointer (vptr) is created for all objects of a class that has at least one virtual function, either directly in the class itself or inherited from a base class.**

**But if a derived (child) class does not declare any virtual functions and does not override any virtual functions from its base (parent) class, then it does not create its own vtable.**