Basics:

(Unreal Styles sind im Style Dokument festgehalten)

Innerhalb von der Unreal Engine kann im C++ Ordner kein neuer Ordner erstellt werden. Dies geht auch außerhalb nicht, man kann nur neue Ordner erstellen, indem man eine neue C++ Klasse erstellt und beim auswählen des Ordners einen neuen Ordner erstellt. Damit erschafft man direkt auch neue Ordner innerhalb der Engine und man kann eine Ordnung erstellen.

Generell kann man von jeder Klasse ableiten und somit neue Abstraktionsebenen schaffen. Man sollte jedoch darauf achten, dass man von den richtigen Klassen ableitet, da sie Einfluss auf den Einsatz innerhalb der Engine haben kann.

Ein Beispiel hier:   
Wenn man eine extra Klasse kreieren möchte, die einem Actor Fähigkeiten hinzufügen kann, sollte man von einer UActorComponent Klasse ableiten. Wenn man die Komponente als eine UPROPERTY() bei einem Actor in der Headerfile hinzufügt, kann man alle UPROPERTY() Elemente von dem ActorComponent im Editor sehen und bearbeiten.

UPROPERTY() hat einige keywords, die vor allem für den Editor und für BP’s wichtig sind. Alle davon kann man [hier](https://wiki.unrealengine.com/UPROPERTY) nachlesen.

Interfaces sind den Interfaces von C# sehr ähnlich und haben wenig mit den abstrakten Klassen von C++ zu tun. Innerhalb von Ihnen können nur pure virtual functions erstellt werden.

Das hinzufügen des keywords innerhalb von UCLASS() (UCLASS(abstract)) verhindert, dass die Klasse direkt im Editor benutzt werden kann. Man kann jedoch innerhalb der Klasse alle Funktionen einer normalen Klasse benutzen.

Die UFUNCTION() keywords setzen vor allem, ob man die Funktion von Blueprints aufrufen kann. Einige Funktionen benötigen das Makro auch, damit man die Funktionen richtig benutzen kann. Ein Bsp.:

//.h file

AFoo();

UPROPERTY()

class UBoxComponent\* m\_boxTrigger;

UFUNCTION()  
void OnOnverlapBegin(class UPrimitiveComponent\* OverlappedComp, class AActor\* OtherActor, class UPrimitiveComponent\* OtherComp, int32 OtherBodyIndex, bool bFromSweep, const FHitResult& SweepResult); // Standard layout für ein OverlapBegin Event.

//.cpp file

Foo::Foo()

{

m\_boxTrigger = CreateDefaultSubobject<UBoxComponent>(TEXT(“BoxTrigger”));

m\_boxTrigger->OnComponentBeginOverlap.AddDynamic(this, &AFoo::OnOverlapBegin); /\* Hier kann nur eine Funktion mit dem richtigen Layout wie auch einem UFUNCTION() Makro über der Funktion benutzt werden \*/

}