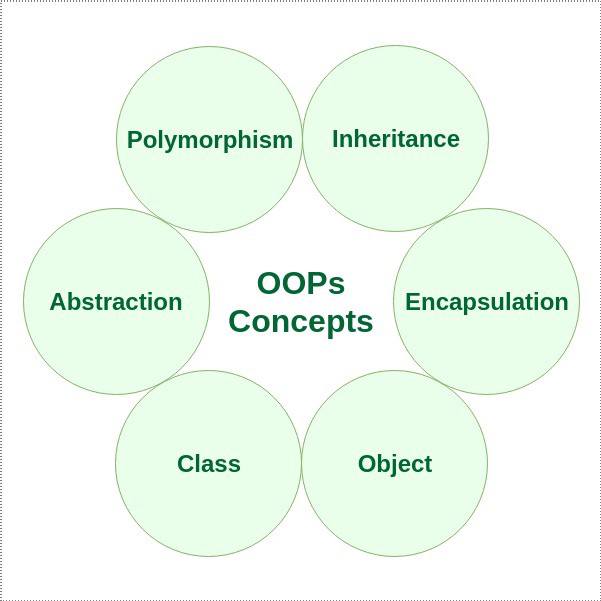
***OOP - OBJECT ORİENTED PROGRAMMİNG - NESNE YÖNELİMLİ PROGRAMLAMA***



* Gerçek dünyadaki nesnelerin tasarımı sınıf içinde yapılır.
* Sınıftan nesne üretilip değişiklik yapılmak istendiğinde tüm programda değişiklik yapmak gerekmez sadece oluşturulan nesnenin sınıf içinde değişiklik yapmak yeterlidir.
* Oluşturulan nesneler birbirinden bağımsız olduğu için bilgi gizleme olanağı artar.

**Örneğin:** A nesnesi B nesnesinin özelliklerini kullanamaz ve erişemez.

* Nesne oluşturma bir sınıf içerisinde gerçekleştirilir ve bu kodlar başka projelerde kullanılabilir.

**Örneğin:** Bir A nesnesi oluşturduysak bunu birçok yerde kullanabiliriz.

* Sınıflar oluşturarak daha az kod oluşturup daha fazla iş yapıp kod tekrarı önlenir.

**Örneğin:** İnsan sınıfında ad, soyad, yaş vb. gibi özellikleri bir defa oluşturup istediğimiz kadar kullanabiliriz.

**Class (Sınıf):**

Gerçek dünyadaki nesnelerin özellikleri ve davranışları sınıflara aktarılır.

Bu durumların sınıflara aktarılması metodlarla (fonksiyonlarla) olur. Sınıfta tanımlanan metot ve değişkenlere sınıfın üyeleri denir. Değişkenler ad, soyad, yaş gibi kullanacağımız bilgileri saklamaktadır.

Metotlar ise, kullanıcı kaydı, iki sayısının toplamı gibi bir görevi yerine getiren alt programlardır.

Sınıf soyut bir kavramdır doğrudan kullanılamaz nesne oluşturup kullanabiliriz.

**Nesne (Object):**

İçinde veri saklayan ve bu veriler üzerinde işlem yapacak olan metodlar bulunduran bileşenlerdir.

Nesneler her uygulamada tekrar kullanılabilir. Nesne oluşturduğumuzda hafızada yer kaplar.

**Nesne Yönelimli Programlama Özellikleri**

1. Abstraction - Soyutlama ==> Bir sınıfta davranış ve özelliklerin tanımlanmasına soyutlama diyoruz.

**Örneğin:** Araba sınıfında rengi, modeli, tekerlek sayısı, motor gücü, özellikleridir(property).

Hızlanması, fren yapması, durması davranışlarıdır ve metotlar (fonksiyonlar) ile tanımlanır.

2. Inheritance - Miras Alma / Kalıtım ==> Sınıflar birbirinden türeyebilir. Alt sınıf üst sınıfın özelliklerini alabilir.

**Örneğin:** Araba ve bisiklet sınıflarında ortak özellik olarak tekerlek sayısı, hızı gibi özelliklerini tekrar yazmak yerine bu özellikleri içeren bir sınıf oluşturup miras alabiliriz. Bir sınıftan birden fazla miras alınıyorsa buna çoklu kalıtım denir.

3. Polymorphism - Çok Biçimlilik ==> Alt sınıflar üst sınıfın gösterdiği davranışları göstermek zorunda değildir. Alt sınıfların farklı davranışları göstermesine Çok biçimlilik denilmektedir.

**Örneğin:** Gemi ve araba sınıflarını ele aldığımızda bunların hareket tipleri bulunmaktadır. Gemi su üzerinden giderken araba karada hareket etmektedir. Kısaca farklı nesnelerin (araba ve gemi gibi) aynı olaya (hareket tipine) farklı şekilde cevap vermesidir.

4. Encapsulation - Kapsülleme ==> Davranış ve özellikler sınıfta soyutlanarak kapsüllenir. Kapsülleme ile hangi özellik ve davranışın dışarıya sunulup sunulmayacağını belirleriz.  
**Örneğin:** İnsan sınıfında yemek alışkanlığı bizi ilgilendirmiyorsa bunu kapalı (private) yapıp gizleriz. Ancak ad soyad gibi bilgiler bizi ilgilendirdiği için bunlar açık bırakılır. Bu olaya bilgi saklama yani kapsülleme denilmektedir. Bilgi saklama erişim belirteçleri (public, private, protected, internal) ile gerçekleştirilir.

**Erişim Belirteçleri**

**public:** Herkesin kullanabileceği özellik ve davranışlardır.  
**private:** Sadece kendi sınıfta kullanılabilen özellikler ve davranışlardır.  
**protected:** Sınıf içinde ve miras alınan alt sınıflarda kullanılır.

**internal**: Aynı program içerisinden erişilebilir, fakat farklı bir program içerisinden erişilemez.