Bölüm 1

Nesne Tabanlı Programlamaya Giriş

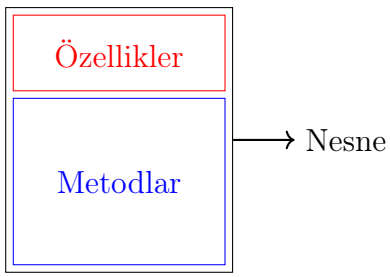
1.1 Giriş

Nesne tabanlı programlama (Object-Oriented Programming, OOP), modern yazılım geliştirmenin temel taşlarından biridir. OOP, yazılım sistemlerini gerçek dünyadaki nesneleri modelleyerek tasarlamayı ve geliştirmeyi sağlar. Bu bölümde, OOP'nin tanımı, tarihçesi, temel prensipleri ve modern yazılım mühendisliğindeki rolü ele alınacaktır. Amaç, okuyucuların OOP'nin neden önemli olduğunu ve nasıl kullanıldığını anlamasıdır.

1.1.1 Nesne Tabanlı Programlama Nedir?

Nesne tabanlı programlama, yazılım geliştirmede nesneler ve sınıflar etrafında organize olmuş bir paradigmadır.

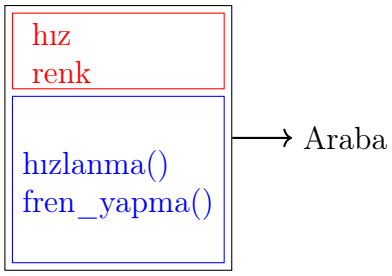
Bir nesne, veri (özellikler) ve bu veriler üzerinde işlem yapan metodların (fonksiyonların) birleşimidir.



Şekil 1.1 – Nesne (object) bileşenleri

Örneğin, bir *araba* nesnesi, hız, renk gibi özelliklere ve hızlanma, fren yapma gibi metodlara sahip olabilir.

OOP, modülerlik, yeniden kullanılabilirlik ve bakım kolaylığı gibi avantajlar sunar.



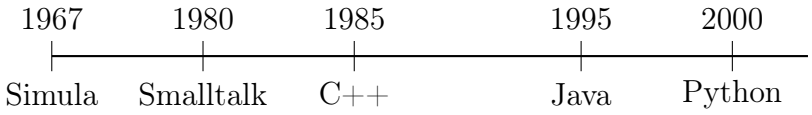
Şekil 1.2 – Araba nesnesinin bileşenleri

OOP'nin temel kavramları şunlardır:

— **Sınıf (Class)**: Nesnelerin şablonudur.
— **Nesne (Object)**: Sınıftan türetilen örneklerdir.
— **Kapsülleme (Encapsulation)**: Verilerin gizlenmesi.
— **Kalıtım (Inheritance)**: Sınıflar arası hiyerarşi.
— **Çok Biçimlilik (Polymorphism)**: Aynı metodun farklı davranışlar sergilemesi.
— **Soyutlama (Abstraction)**: Gereksiz detayların gizlenmesi.

1.2 Tarihsel Gelişim

Nesne tabanlı programlamanın kökeni 1960'lara dayanır. İlk olarak Norveç'te geliştirilen **Simula** dili, sınıflar ve nesneler kavramını tanıttı. 1980'lerde **Smalltalk**, OOP'nin modern formunu şekillendirdi. Günümüzde birçok dil OOP'yi destekler.



Şekil 1.3 – Nesne Tabanlı Programlamanın Tarihsel Gelişimi

1.2.1 OOP Destekli Programlama Dilleri

Nesne tabanlı programlama, birçok modern programlama dilinde desteklenmektedir. Aşağıdaki tablo, OOP'yi destekleyen başlıca dilleri, çıkış yıllarını ve temel özelliklerini özetlemektedir. Aşağıda, seçili dillerin (C++, Java, C#, Python) OOP desteği hakkında detaylı bilgiler verilmiştir:

C++: C++, sınıf tabanlı bir OOP dilidir ve dört temel prensibi tam olarak destekler. Kapsülleme, erişim belirleyiciler (public, private, protected) ile sağlanır. Kalıtım, tekli ve çoklu kalıtım (multiple inheritance) şeklinde desteklenir, ancak çoklu kalıtım diamond problem gibi sorunlar yaratabilir. Çok biçimlilik, fonksiyon aşırı yükleme (overloading), operatör aşırı yükleme ve sanal fonksiyonlar (virtual functions) ile gerçekleştirilir. Soyutlama, soyut sınıflar ve saf sanal fonksiyonlar ile uygulanır. Ek özellikler arasında şablonlar (templates) ile genel programlama (generics), arkadaş fonksiyonlar (friend functions) ve

Tablo 1.1 – OOP Destekli Programlama Dilleri

Dil	**Çıkış Yılı**	**Özellikler ve Önem**
Simula	1967	İlk OOP dili, sınıflar ve nesneler kavramını tanıttı.
Smalltalk	1980	Modern OOP'nin temeli, dinamik tip sistemi ve GUI odaklı.
C++	1985	Performans odaklı, sistem programlamada yaygın.
Java	1995	Platform bağımsızlığı, kurumsal uygulamalarda popüler.
Python	2000	Çok paradigmalı, okunabilirlik ve esneklik sunar.
C#	2000	Microsoft ekosisteminde güçlü, oyun geliştirmede yaygın.
Ruby	1995	Web geliştirmede (Ruby on Rails) popüler, esnek syntax.
JavaScript	1995	Prototip tabanlı OOP, web uygulamalarında baskın. Tam bir OOP dili olup olmadığı tartışmalıdır (aşağıya bakınız).
Swift	2014	Apple ekosisteminde modern, güvenli OOP.
Kotlin	2011	Android geliştirmede Java'nın modern alternatifi.

RAII (Resource Acquisition Is Initialization) yer alır. C++, performans odaklıdır ve sistem programlamada yaygın kullanılır, ancak manuel bellek yönetimi (pointers) karmaşıklık ekleyebilir.

Java: Java, sınıf tabanlı bir OOP dilidir ve platform bağımsızlığı ile bilinir. Kapsülleme, erişim belirleyiciler ve getter/setter metodları ile sağlanır. Kalıtım, tekli kalıtım (extends) ile sınırlıdır, ancak arayüzler (interfaces) çoklu kalıtım benzeri davranış sağlar. Çok biçimlilik, metod aşırı yükleme ve geçersiz kılma (overriding) ile desteklenir. Soyutlama, soyut sınıflar (abstract classes) ve arayüzler ile uygulanır. Ek özellikler arasında generics, istisna yönetimi (exceptions) ve paketler (packages) bulunur. Java, kurumsal uygulamalar için uygundur ve otomatik çöp toplama (garbage collection) ile bellek yönetimini kolaylaştırır, ancak performans açısından C++'dan daha yavaştır.

C#: C#, sınıf tabanlı bir OOP dilidir ve .NET ekosistemi ile entegredir. Kapsülleme, özellikler (properties) ve erişim belirleyiciler ile gerçekleştirilir. Kalıtım, tekli kalıtım ile sınırlıdır, ancak arayüzler çoklu kalıtım benzeri esneklik sağlar. Çok biçimlilik, sanal metodlar (virtual/override) ve aşırı yükleme ile desteklenir. Soyutlama, soyut sınıflar ve arayüzler ile uygulanır. Ek özellikler arasında olaylar (events), temsilciler (delegates), LINQ (Language Integrated Query) ve records (immutable classes) yer alır. C#, oyun geliştirme (Unity) ve Windows uygulamalarında güçlüdür, otomatik bellek yönetimi ve modern özelliklerle geliştirici dostudur.

Python: Python, sınıf tabanlı ve dinamik tipli bir OOP dilidir, çok paradigmalı yapısıyla esnektir. Kapsülleme, isim konvansiyonları (_private, __name_mangling) ile sağlanır, ancak katı bir erişim kontrolü yoktur. Kalıtım, çoklu kalıtım destekler ve MRO (Method Resolution Order) ile çözer. Çok biçimlilik, duck typing ve metod geçersiz kılma ile doğal olarak gerçekleşir. Soyutlama, abc modülü ile soyut sınıflar kullanılarak uygulanır. Ek özellikler arasında sihirli metodlar (magic methods: init, str), mixins ve dinamik özellik ekleme bulunur. Python, okunabilirlik ve hızlı geliştirme için idealdir, ancak dinamik tip sistemi hata yakalamayı zorlaştırabilir.

JavaScript: JavaScript'in nesne tabanlı programlama (OOP) dili olarak sınıflandırılması tartışmalıdır. JavaScript, prototip tabanlı miras kullanır; bu, geleneksel sınıf tabanlı dillerden (örneğin, Java, C++) farklıdır. ES6 ile gelen `class` sözdizimi, prototip tabanlı sistemi kolaylaştıran bir sözdizimsel kolaylık sağlar, ancak altta yatan mekanizma hala prototiplere dayanır. JavaScript, kapsülleme (kapanışlar ve özel alanlar ile), miras (prototip zinciri ile) ve çok biçimlilik (metod geçersiz kılma ile) gibi OOP ilkelerini destekler, ancak dinamik tip sistemi ve işlevlerin birinci sınıf üye olması, bazı geliştiricilerin bunu klasik bir OOP dili olarak görmemesine neden olur. Yine de, JavaScript'in çok paradigmalı yapısı, hem OOP hem de işlevsel programlamayı destekleyerek esnek ve güçlü bir dil olmasını sağlar.

1.3 Prosedürel Programlama ile Karşılaştırma

Prosedürel programlama, işlevleri ve veri yapılarını ayrı tutar. Örneğin, bir prosedürel programda bir araba modeli şöyle olabilir:

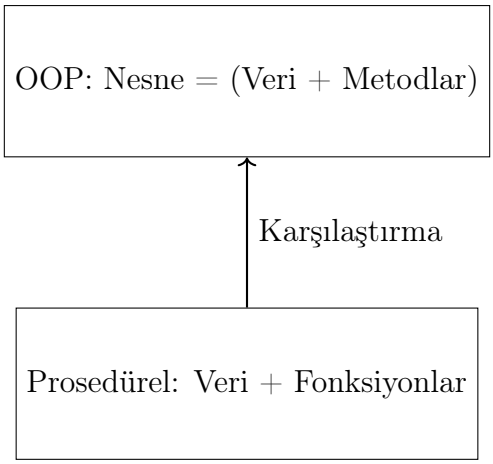
Sözde-kod 1.1 – Prosedürel Araba Modeli

```
yapı Araba {
 renk: metin
 hız: tamsayı
}

fonksiyon hızlan(Araba a) {
 a.hız = a.hız + 10
}
```

OOP'de ise aynı model şöyle olur:

Sözde-kod 1.2 – OOP Araba Modeli

```
sınıf Araba {
 özellik renk: metin
 özellik hız: tamsayı

 metod hızlan() {
  hız = hız + 10
 }
}
```



Şekil 1.4 – Prosedürel ve Nesne Tabanlı Programlama

1.4 Modern Bağlamda OOP

OOP; mikro hizmetler, bulut tabanlı sistemler ve büyük ölçekli yazılım projelerinde kritik bir rol oynar. Örneğin, bir e-ticaret platformunda kullanıcı, ürün ve sepet nesneleri modüler bir şekilde tasarlanabilir. Modern trendler, OOP'nin fonksiyonel programlama ile entegrasyonunu (örneğin, Python'da) ve test odaklı geliştirmeyi vurgular.

1.5 Alıştırmalar

1. OOP'nin temel kavramlarını kendi kelimelerinizle açıklayın.
2. Simula'nın OOP tarihindeki önemini tartışın.
3. Prosedürel ve nesne tabanlı programlama arasındaki farkları bir tablo ile karşılaştırın.
4. Gerçek dünyadan bir nesne seçin (örneğin, bir lamba) ve bu nesneyi OOP yaklaşımıyla nasıl modelleyeceğinizi tarif edin.
5. Tablo 1.1'deki dillerden birini seçin ve bu dilin OOP özelliklerini araştırarak kısa bir özet yazın. JavaScript'i seçerseniz, tam bir OOP dili olup olmadığı tartışmasını da ele alın.