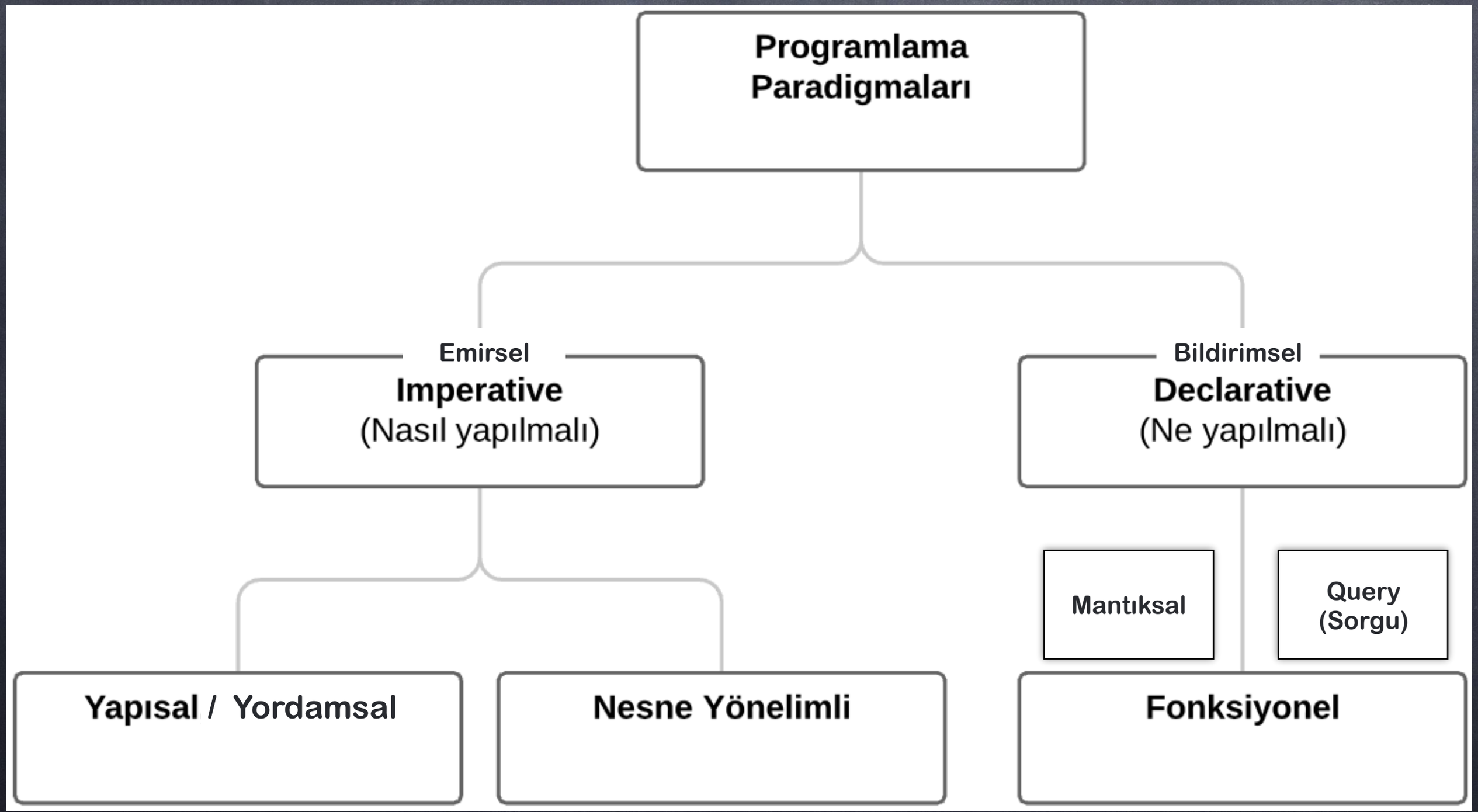
Nesne Yönelimli Analiz ve Tasarım

Programlama Paradigmaları

Programlama Paradigmaları

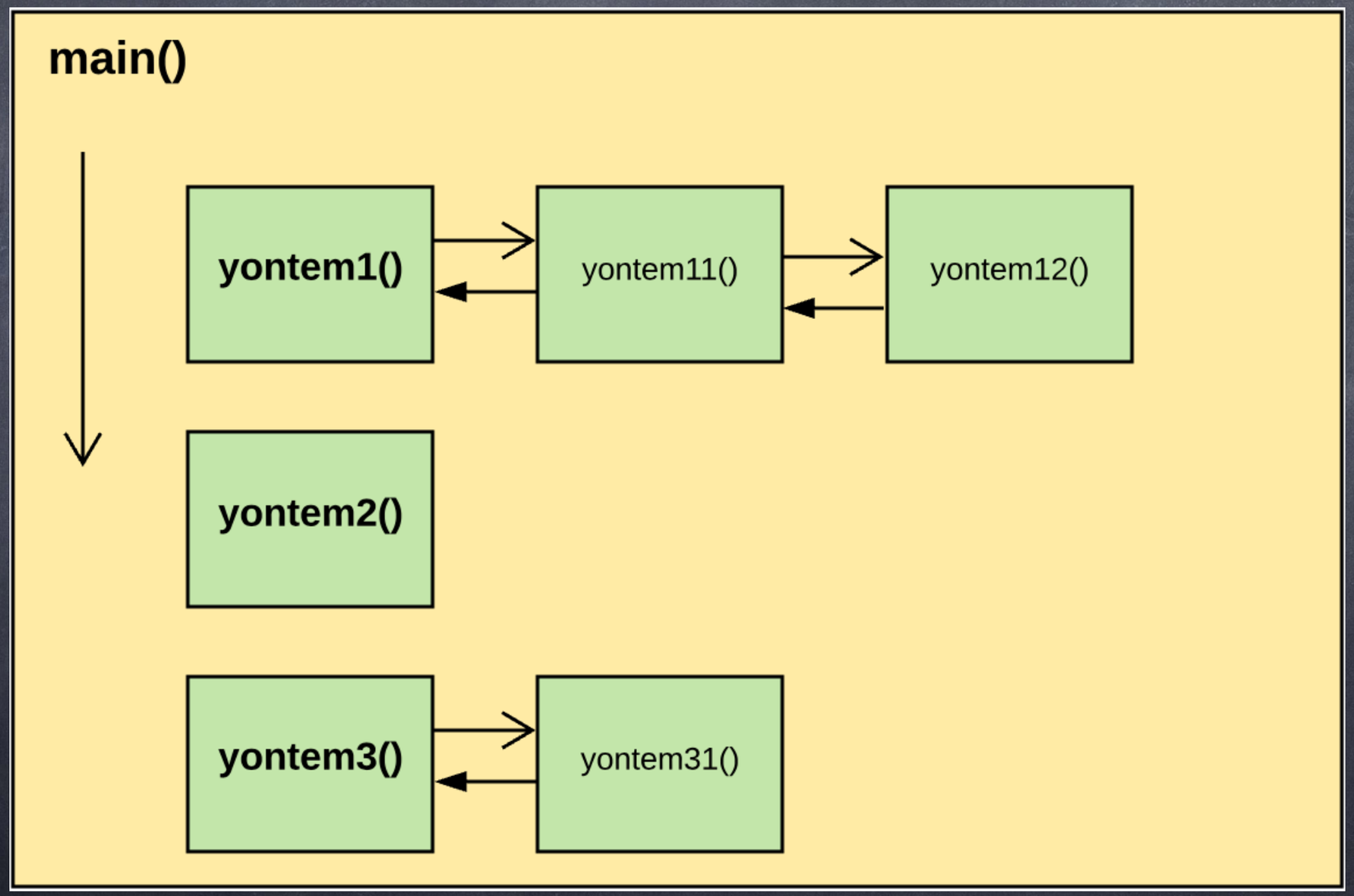


Yapısal (Yordamsal-Prosedürel) Programlama Paradigması

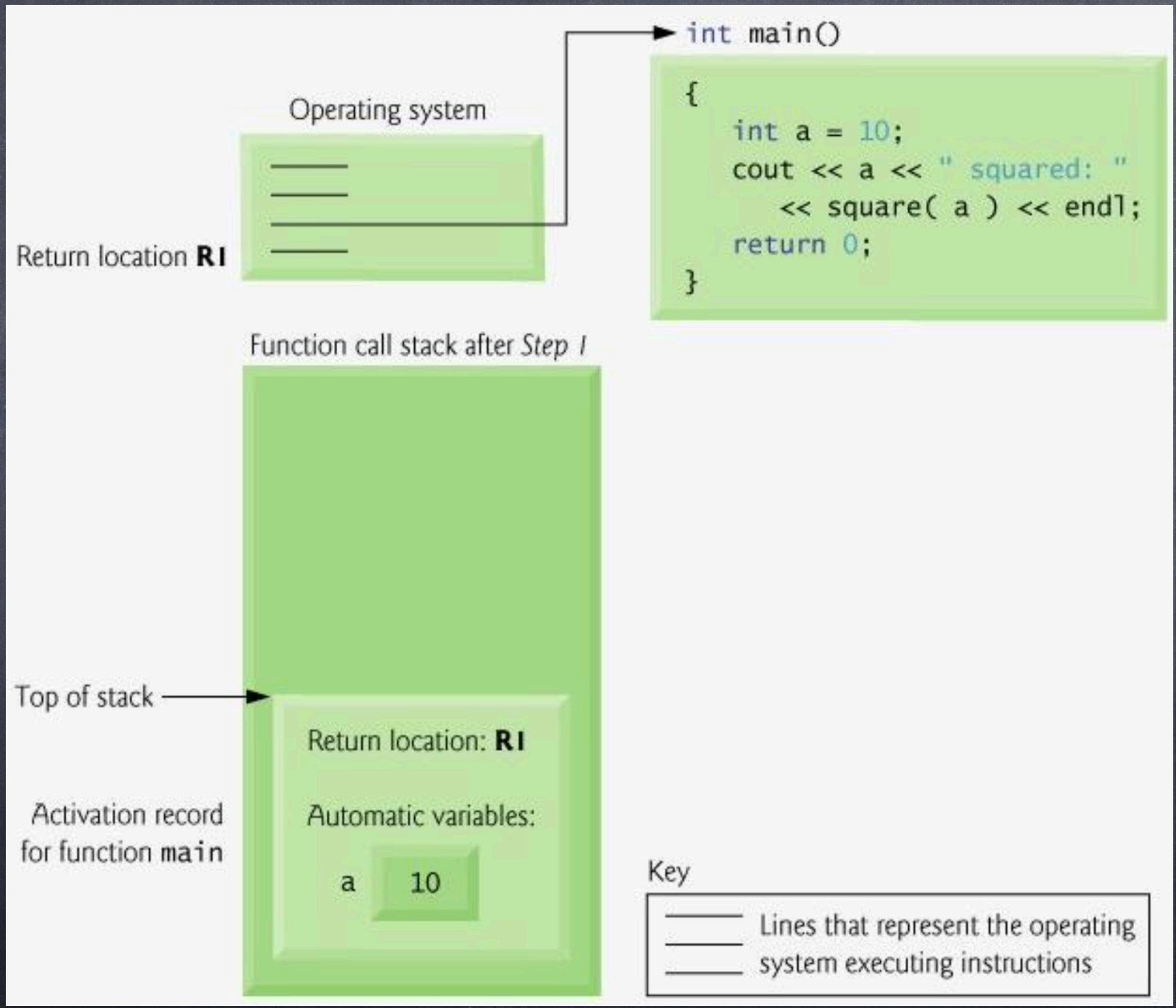
* Structured programming is a programming paradigm aimed at improving the clarity, quality, and development time of a computer program by making extensive use of subroutines, block structures and for and while loops in contrast to using simple tests and jumps such as the goto statement which could lead to "spaghetti code" which is difficult both to follow and to maintain.

https://en.wikipedia.org/wiki/Structured_programming

Yapısal (Yordamsal-Prosedürel) Programlama Paradigması

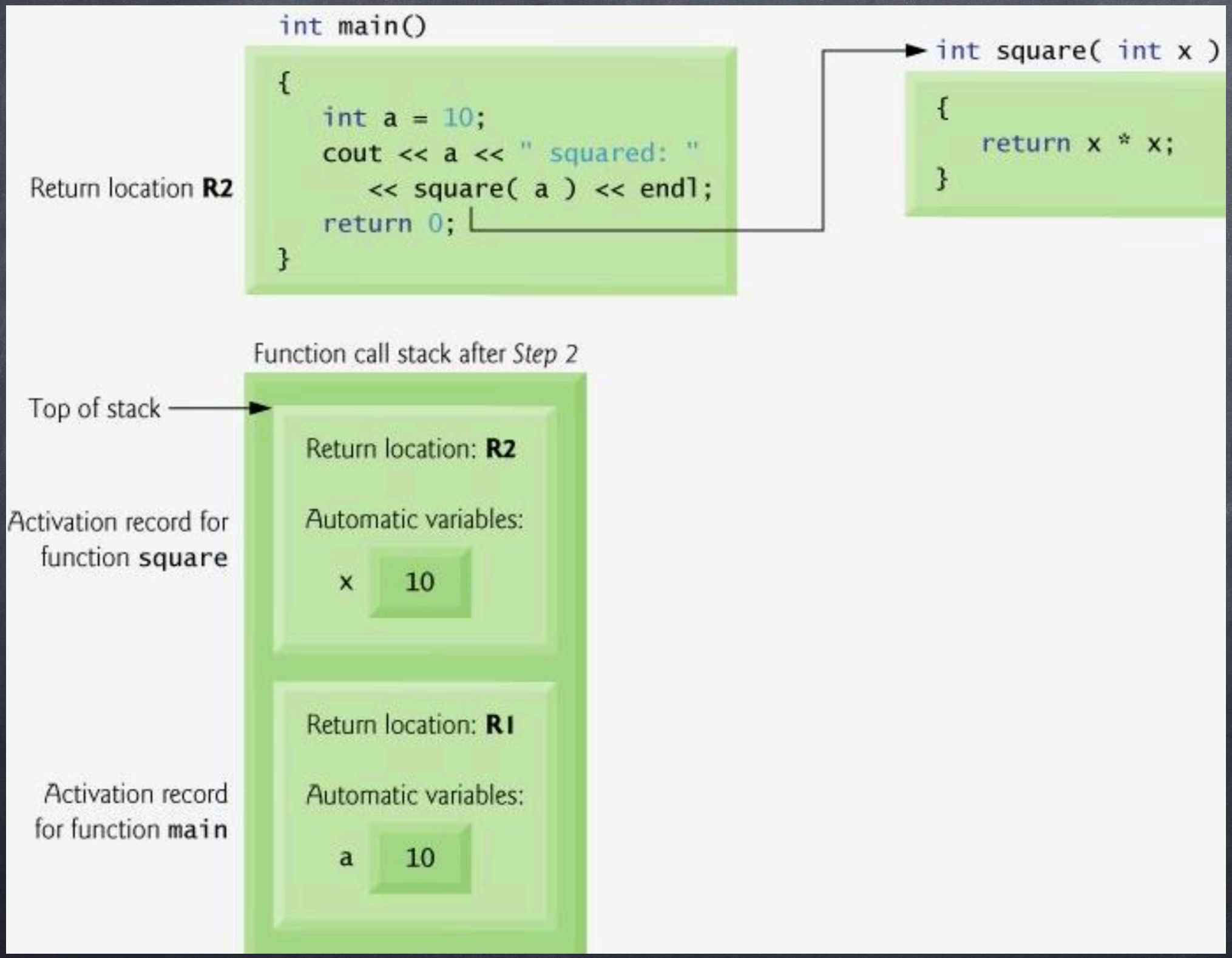


Yapısal (Yordamsal-Prosedürel) Programlama Paradigması



Deitel, C++ How To Program, Prentice Hall

Yapısal (Yordamsal-Prosedürel) Programlama Paradigması

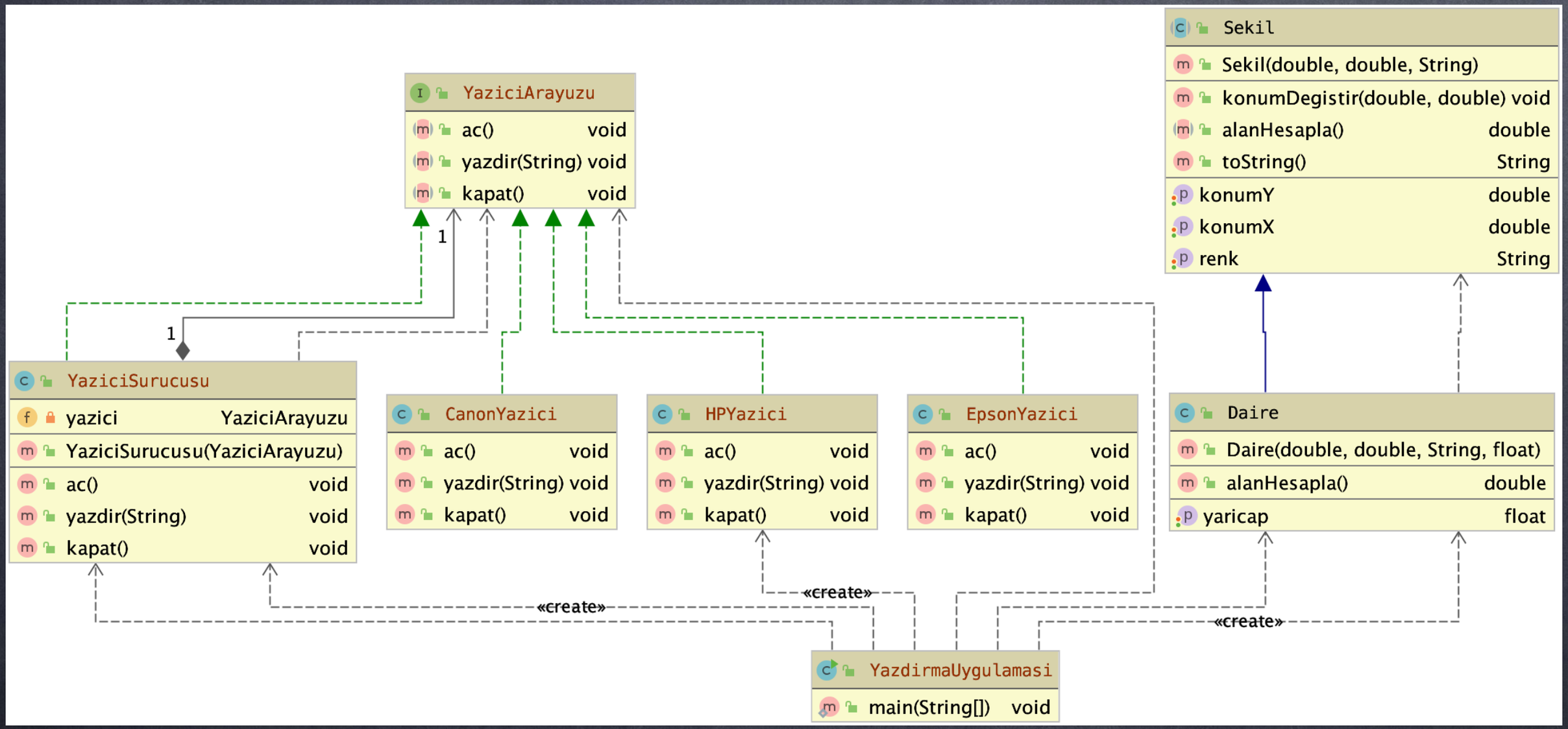


Deitel, C++ How To Program, Prentice Hall

Nesne Yönelimli Yazılım Geliştirme Paradigması

* Yapısal teknikte doğrudan probleme odaklanılır ve problemin çözümüne ilişkin yöntemler geliştirilir (control-centric).
* Nesne yönelimli programlama tekniğinde ise temel bileşen nesnedir (data-centric) ve programlar nesnelerin birlikte çalışmasından meydana gelir.
* Nesne hem veriyi hem de bu veriyi işleyen yöntemleri içerir. Yazılım geliştiriciler dikkatlerini nesneleri oluşturan sınıfları geliştirmeye yoğunlaştırır. Böylece program zekası veri merkezli olarak tüm modüllere dağıtılır.
* Yapısal teknikte bir fonksiyon herhangi bir görevi yerine getirmek için veriye ihtiyaç duyarsa, gerekli veri parametre olarak gönderilir. NYP de ise yerine getirilmesi gereken görev nesne tarafından icra edilir ve fonksiyonlar verilere parametre gönderimi yapılmaksızın erişebilirler.

Nesne Yönelimli Yazılım Geliştirme Paradigması



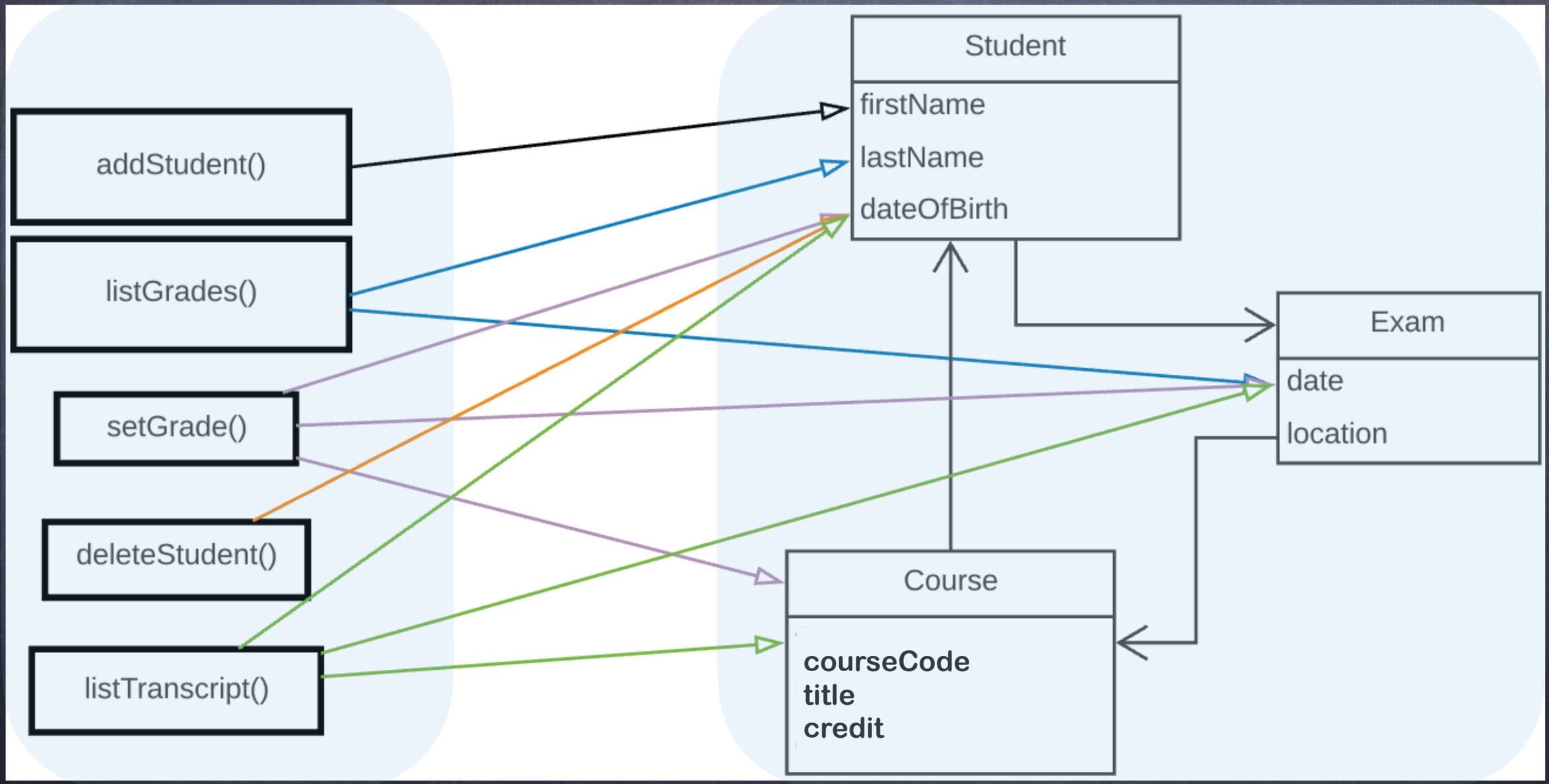
Nesne Yönelimli Yazılım Geliştirme Paradigması

* Sistem büyüdükçe ilişkiler/bağımlılık daha da karmaşıklaşır.
* Sonradan değişiklik zorlaşır.
* Program içerisinde değişiklik (ekleme, çıkarma, düzeltme) yapmak zorlaşır ve beklenmeyen etkilere neden olabilir...
* Örneğin(bir sonraki yansıda yer alan ÖBS uygulaması);
  * Ogrenciler tablosunda ogrencinin dogum tarihi iki haneli
  * Bu alanı 4 haneli yapmak istiyoruz...
  * Bu veriyi kullanan yöntemlerinde güncellenmesi gerekecektir.
  * Hangi yöntemler hangi veriye erişiyor ?

Nesne Yönelimli Yazılım Geliştirme Paradigması

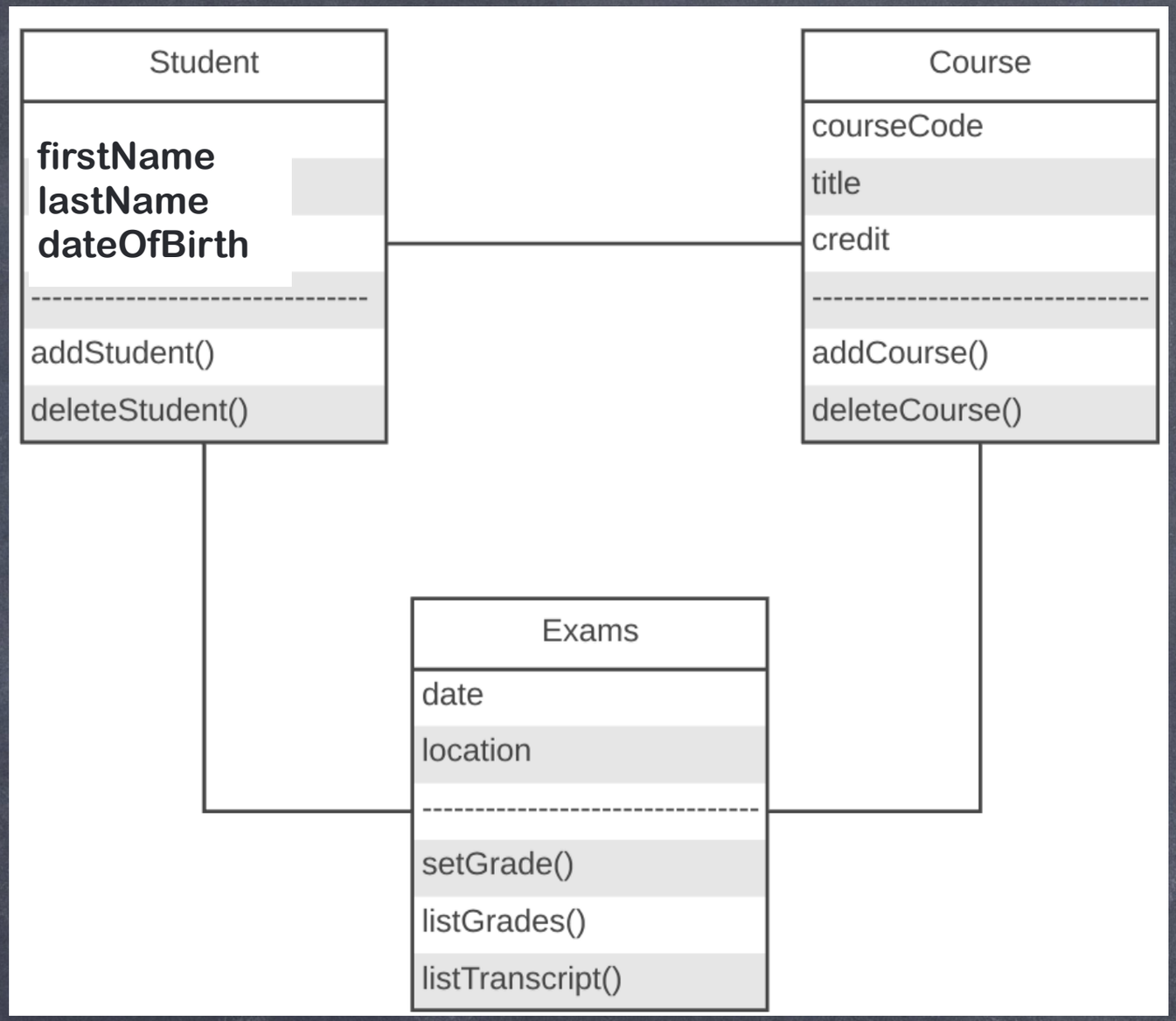
Yöntemler

Veri



Hangi yöntemler hangi veriye erişiyor ?

Nesne Yönelimli Yazılım Geliştirme Paradigması



High Coherence

Modüller tek ve özel bir işi, mükemmel bir şekilde yapmalı. Alt sistemler içerisindeki sınıflar benzer işi yapmalı ve ilgili olmalı. "Cohesion" Bunun ölçüsüdür .

Low Coupling

Modüller arasındaki bağlantı zayıf olmalıdır. Böylece yapılacak değişikliğin bağlantılı modülleri etkilemesi önlenir.

Nesne Yönelimli Yazılım Geliştirme Paradigmasının Temel Özellikleri

* Encapsulation (Information Hiding)
* Inheritance
* Polymorphism
* Abstraction
* Modular Programming
* Code Reuse
* Maintenance
* Design Principles (SOLID)
* Design Patterns

Uygulamalar
https://github.com/celalceken/NesneYonelimliAnalizVeTasarimDersiUygulamalari/tree/master/Ders2

Fonksiyonel Programlama

* (Pure) saf fonksiyonların birleşiminden oluşur
  * aynı giriş için aynı çıkış üretilir (no side effect)
  * paylaşılan/global değişken yoktur
  * parametre olarak gelen nesnenin durumları değiştirilemez.
* Tekrarlı yapılar (döngü) yoktur.
* Durum değişimine izin verilmediği için paralel programlar yazılabilir (akış işlemede kullanılır).

Uygulamalar
https://github.com/celalceken/NesneYonelimliAnalizVeTasarimDersiUygulamalari/tree/master/Ders4/VeriTopluluklariUzerindeIslemler

```
List<Kitap> kitaplarV = new Vector<Kitap>();
kitaplarV.add(new Kitap("Veritabanı Yönetim Sistemleri ",100.00));
kitaplarV.add(new Kitap("Nesne Yönelimli Programlamaya Giriş ",125.00));
kitaplarV.add(2,new Kitap("Bilgisayar Ağları",150.00));
kitaplarV.add(3,new Kitap("Veri Yapıları",160.00));

for (int i=0; i<kitaplarV.size();i++)
    System.out.println(kitaplarV.get(i).getAdi());

// ************Lambda İfadeler************
kitaplarV.forEach(kitap -> System.out.println(kitap.getAdi()));
```

```
double sum=0;
for (int i=0; i<kitaplarV.size();i++){
    sum+=kitaplarV.get(i).getBirimFiyati();
}
System.out.println(sum);

// ************Lambda İfadeler************
sum= kitaplarV.stream().mapToDouble(item->item.getBirimFiyati()).sum();
System.out.println(sum);

List<Kitap> filtrelenmisKitaplar = kitaplarV
        .stream()
        .filter(item -> item.getAdi().startsWith("V"))
        .collect(Collectors.toList());
System.out.println(filtrelenmisKitaplar);
```