SAKARYA ÜNİVERSİTESİ

Programlamaya Giriş
HAFTA 1
Ders Tanıtımı & Giriş

Prof. Dr. Cemil ÖZ

Doç. Dr. Cüneyt BAYILMIŞ

Dr. Öğretim Üyesi Gülüzar ÇİT

Konu & İçerik

- Dersin Tanıtımı & Amacı
- Ders İçeriği
- Ders Değerlendirme
- Önerilen Kaynaklar
- Temel Kavramlar
- Yazılım Geliştirme Yaşam Döngüsü
- Yazılımın Kalitesi
- Programlama Dilleri
- Yazılımların Çalıştırılması
- Programlama Teknikleri
  - Yapısal/Modüler Programlama
  - Nesneye Dayalı Programlama

Dersin Tanıtımı & Amacı

- Programlamaya Giriş
  - Bir programlama probleminin çözümü için gerekli ilke ve evreleri kavrayabilme.
  - Bir problemin çözümü için gerekli algoritma ve akış şemalarının oluşturulması.
  - Programlama dili kullanılarak, bir programlama dilinin yapısını anlayabilme ve kullanabilme.
  - Algoritma ve akış şemaları hazırlanan problemlerin bir programlama dili kullanarak kod yazımını yapabilme.
  - Yapısal programlama (Değişkenler, karar yapıları, döngüler, diziler, alt programlar, yapı gibi kavramları anlayabilme ve kullanabilme.)
  - Nesneye Dayalı Programlamaya giriş

Ders İçeriği

- Algoritma Kavramı
- Akış Diyagramı Kavramı
- Programlama Kavramı ve Programlama Dili (C/C++)
- Yapısal Programlama
  - Değişkenler, karar yapıları, döngüler, diziler, alt programlar, yapılar, vs.
- Nesneye Dayalı Programlama
- İşaretçiler
- Dosyalama

Ders Değerlendirme

- **Yıl İçi Başarı Oranı** **%55**
  - Vize %45
  - Ödev (4 adet) %40
  - Proje %15
- **Yıl Sonu Başarı Oranı** **%45**

Önerilen Kaynaklar

- C++ How to Program, Deitel, P., Deitel, H.M., Ninth Edition
- C++ ile Programlama, Çeviri Editörü: Cemil Öz, Dokuzuncu Baskından Çiviri, Palme Yayıncılık, 2016
- C++ ile Nesneye Yönelimli Programlamaya Giriş, Doç. Dr. Cemil ÖZ, Sakarya Yayıncılık, 2014
- Fahri Vatansever, İleri Programa Uygulamaları, Seçkin Yayıncılık, 2006, Ankara
- Fahri Vatansever, Algoritma Geliştirme ve Programlamaya Giriş, Seçkin Yayıncılık, 2002, Ankara.

Temel Kavramlar

- **Bilgisayar**
  - Genel olarak, girilen veriler üzerinde aritmetiksel, mantıksal ve ilişkisel işlemler yapabilen ve verileri depolama yeteneğine sahip olan elektromekanik cihazlara **bilgisayar** denir.
  - **Donanım**
    - Bilgisayarı oluşturan fiziksel parçalar
      - Mikroişlemci
      - Bellek
      - Giriş / Çıkış Birimi
  - **Yazılım**
    - Gerçek dünya problemlerini bilgisayar yardımıyla çözmek için **yazılım (program)** geliştirilir.
    - Probleme ilişkin çözümlerin, herhangi bir programlama dili kullanılarak, bilgisayarın anlayacağı komutlar dizisi şeklinde yazılmasına **program**, bilgisayarda kullanılan programların genel adına da **yazılım** denir.

Temel Kavramlar…

Programlama

- Günlük hayatta karşılan genel veya özel bir problemin makineler ile çözülmesi istendiğinde öncelikle problemin gerçek hayattan soyutlanıp makineye anlatılması gerekir.

- Bu problemlerin makinelere tanıtılması, öğretilmesi ve öğretilen çözüm yolları ile sonuca ulaştırılmasını sağlamama süreçlerimin tamamını kapsayan süreç **programlama** olarak adlandırılır.

Temel Kavramlar…

- **Programlama Dili**
  - Günlük hayatta karşılan genel veya özel bir problemin makineler ile çözülmesi istendiğinde problemin gerçek hayattan soyutlanıp makineye anlatılmasını sağlayan araç
  - Programlama dilleri, yazılımcının bilgisayara neyi nasıl yapacağını hangi koşullarda hangi işlemlerin yapılacağını tam olarak anlatmasını sağlar.
  - Her programlama dili diğerinden farklıdır. Her birinin farklı komutları ve kuralları mevcuttur. En iyi programlama dili budur diye bir şey söylemeyiz. Çünkü her programlama dilinin bir diğerine göre üstün ve zayıf tarafları vardır.
  - Günümüzde en yaygın kullanılan kullanılan programlama dilleri
    - Java ,C ,C++ ,C#, PHP, Python, JavaScript, ….

- **C/C++**
  - Neden?
    - Yapısal özellikleri güçlü
    - Her bilgisayar mühendisinin bilmesi gerekir
    - Sonraki yıllarda görülecek derslerde detaylı olarak verilen C++, C#, Java, vs. gibi programa dilleri ve diğer temel yazılım dersleri için temel teşkil etmekte

Temel Kavramlar…

- **Algoritma**
  - Bir problemin çözümüne yönelik işlem basamaklarının sıralı biçimde ve sonlu olarak ifade edilmesi
  - Bilgisayar dilinde " bir sorunun çözümü için öngörülen işlemlerin mantıksal ve sembolik anlatımı"
  - 9. yüzyılda yaşamış Türk-İslam matematikçi ve astronomu Harzemli Mehmet, toplama, çıkarma, ikiye bölme, bir sayının iki katını bulma, denklem çözümü, vb. gibi cebirsel işlemleri açıklayan bir çalışma yaptı. Batılılar, onun bu çalışmalarına Latince "algorismus" yani bugünkü adı ile **algoritma** dediler ve algoritma kavramı ilk defa burada kullanılmış oldu
  - Bir algoritma yazmak için mutlaka bir dile bağımlı kalma zorunluluğu yoktur. Önemli olan yazılan algoritmanın herhangi bir programlama diline uyarlanabilir olmasıdır.

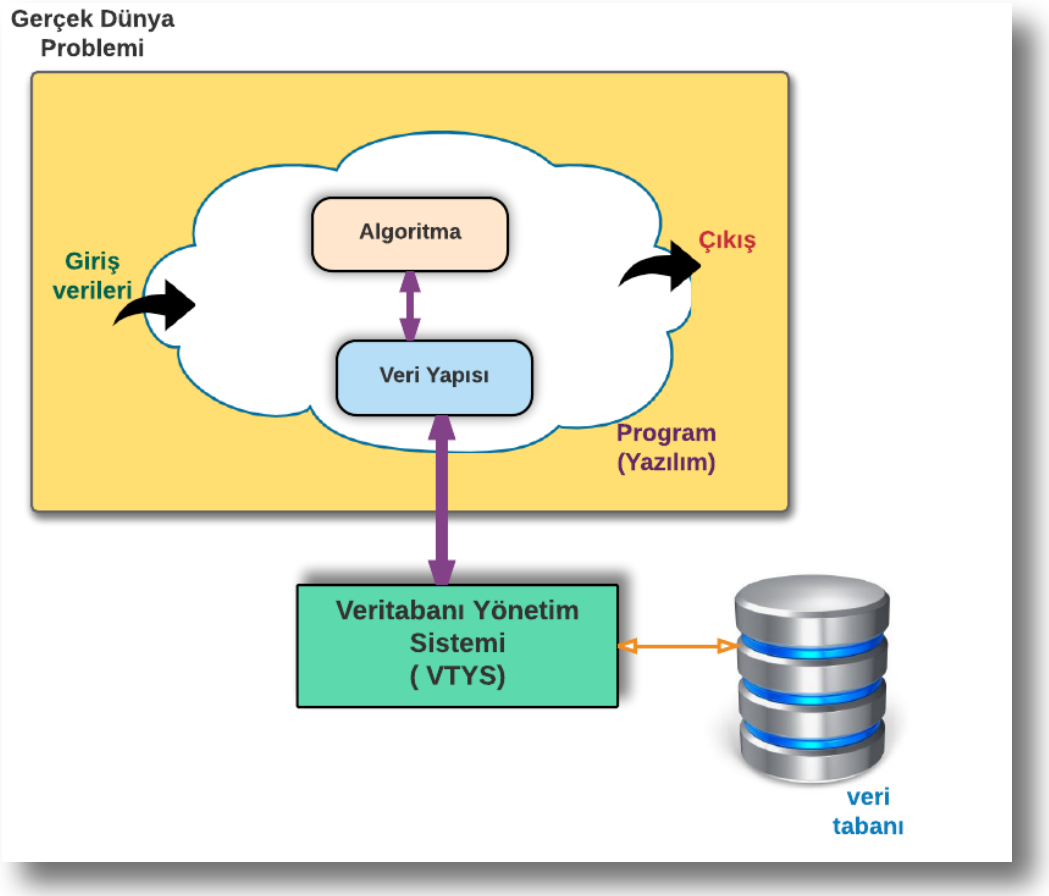
Temel Kavramlar…

- **Akış Diyagramı**
  - Algoritmaların özel geometrik şekiller ile gösterilmesidir

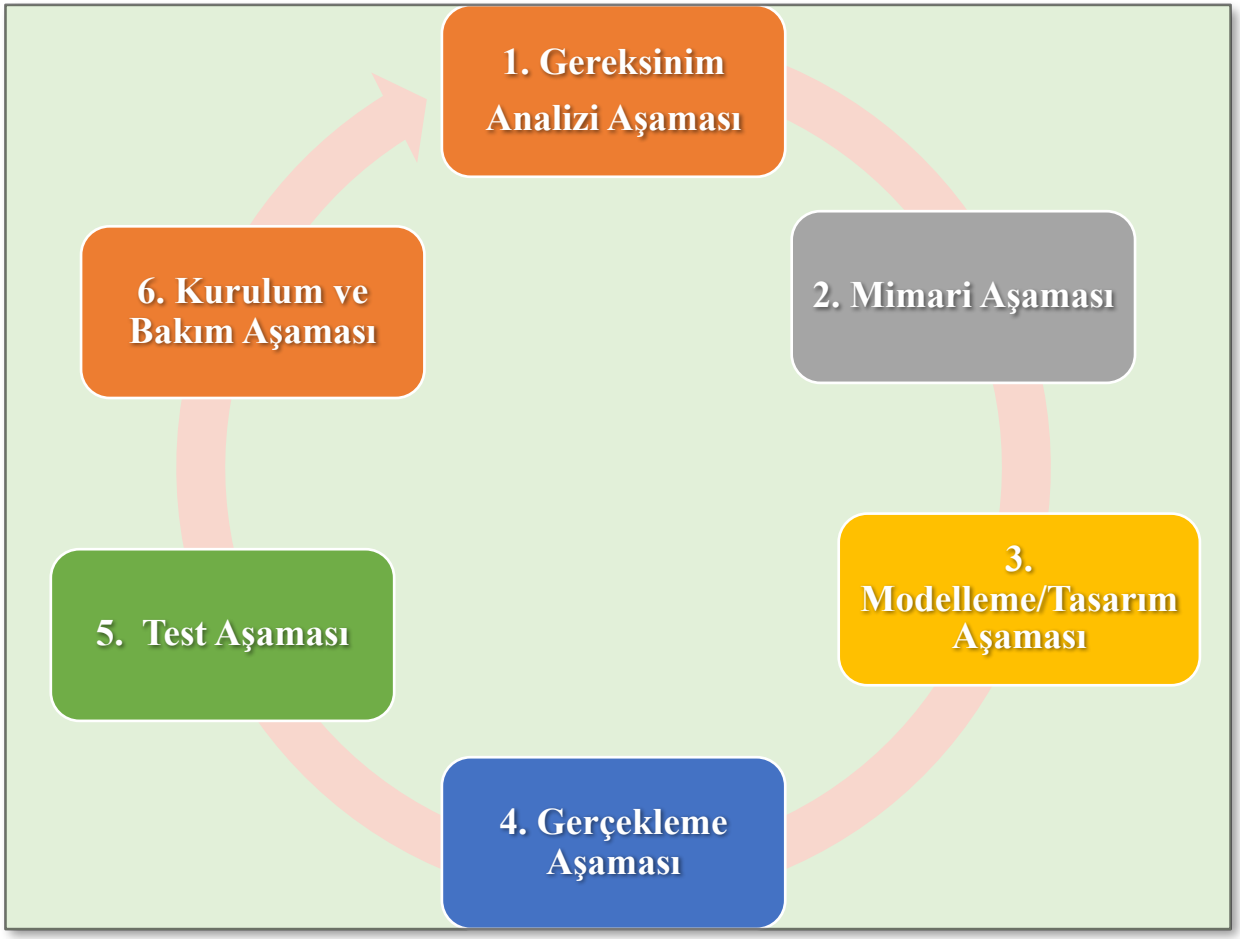
- **Sözde Kod**
  - Algoritmaların ifade edilmesinde akış diyagramlarının yanı sıra, konuşma dili ile programlama dili arasında, **sözde kod** (pseudo-code) adı verilen bir araç kullanılır.
  - Programlama dilinden bağımsız olarak yazılan kodlardır, yani programlar gibi derlenmez veya işlenmezler.
  - Her programcı kendi sözde kodunu geliştirebilir . Fakat kişisel sözde kodlar başkaları tarafından anlaşılabilir bir biçimde açık olmalıdır.

Büyük Resim – Algoritmik Problem

- Gerçek dünya problemlerini bilgisayar yardımıyla çözmek için **yazılım (program)** geliştirilir.

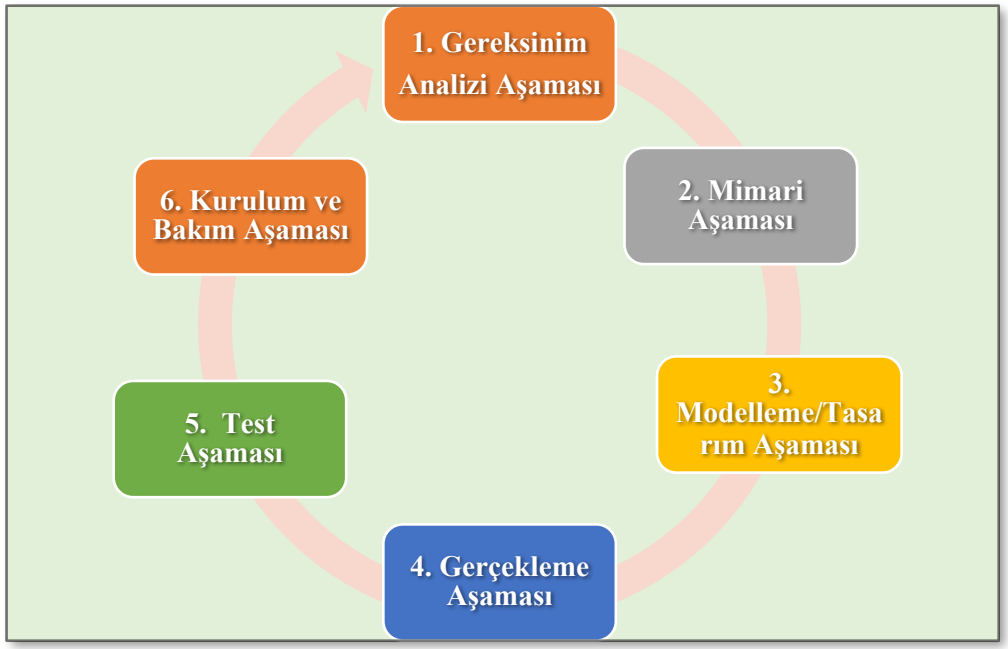


Yazılım Geliştirme Yaşam Döngüsü



Yazılım Geliştirme Yaşam Döngüsü…

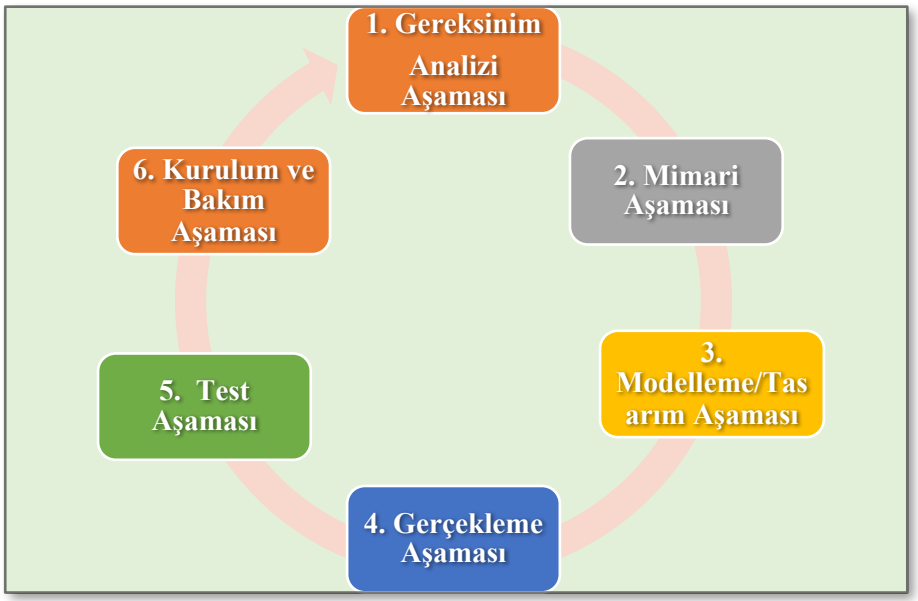
- **Çözümleme (Analiz) / Gereksinim Analizi**
  - Belirsizlik kalmayacak şekilde çözülmesi istenen problem anlaşılır ve yazılımdan yapması beklenenler ayrıntılı olarak listelenir.
  - İhtiyaç listesi oluşturulur.



Yazılım Geliştirme Yaşam Döngüsü…

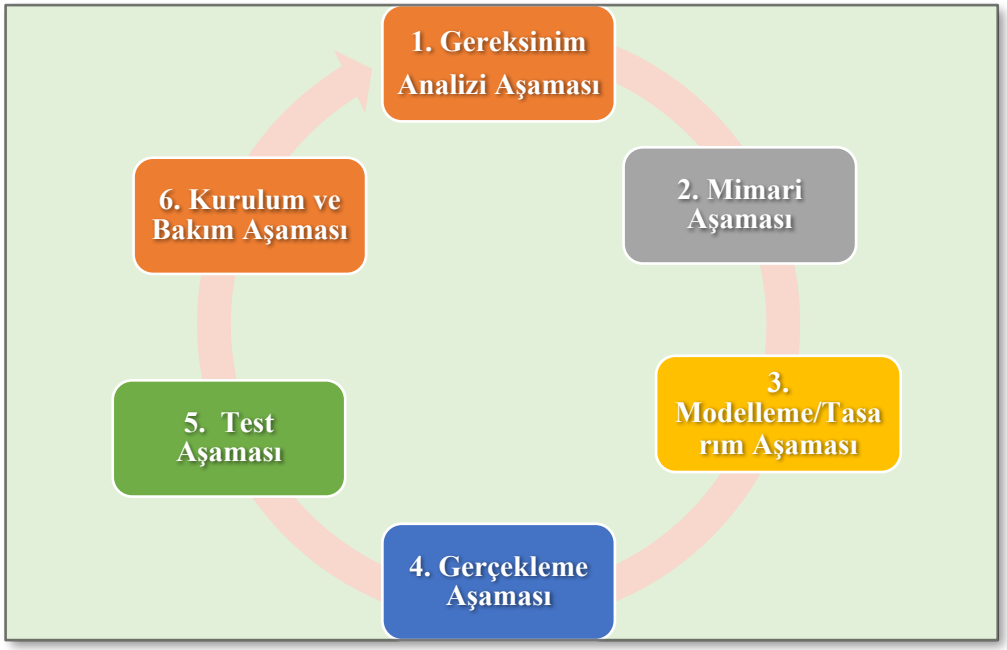
Modelleme / Tasarım Aşaması

- Bu aşamada amaç, gerçek dünyadaki problemin bilgisayarda temsil edilebilecek soyut bir modelinin oluşturulmasıdır. Çözümün hangi unsurlardan oluşacağı ve bu parçaların nasıl modelleneceği tamamıyla programlama yöntemine bağlıdır.
- Tasarım sonrası ortaya çıkacak olan yazılımın kalitesini doğruda etkilediğinden bu aşamada iyi ve doğru bir çözümün oluşturulması çok önemlidir. Bu nedenle kodlamaya geçilmeden önce kurulan modelin sağlamasının (verification) yapılması gerekir.
- UML (Unified Modelling Language), akış diyagramı, sözde kod, vs.



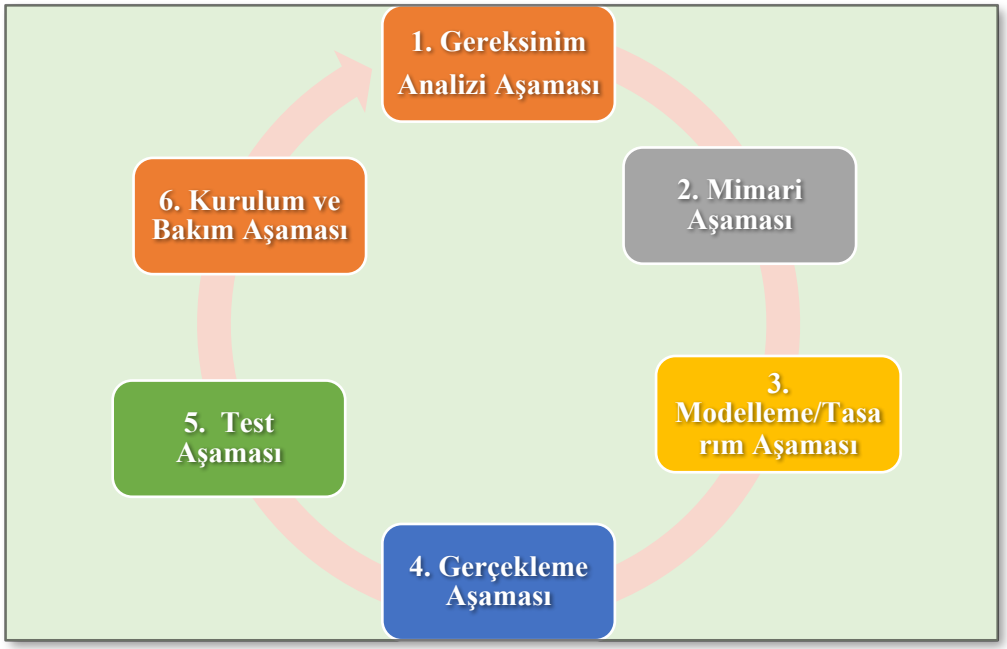
Yazılım Geliştirme Yaşam Döngüsü…

- **Gerçekleme Aşaması**
  - Tasarım aşamasında oluşturulan model, bir programlama dili ile bilgisayara bu aşamada aktarılır.



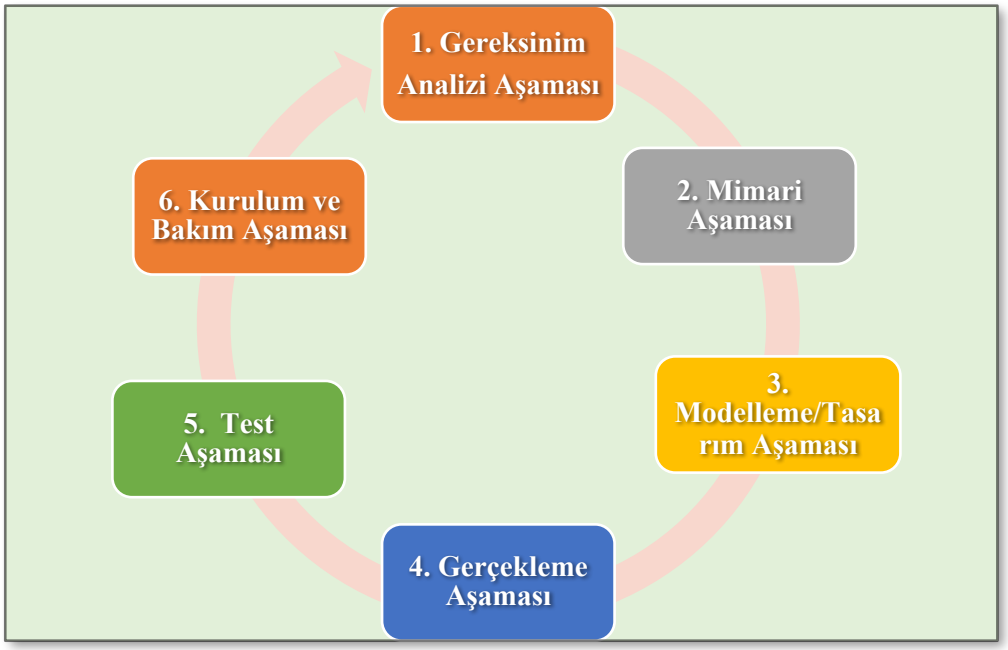
Yazılım Geliştirme Yaşam Döngüsü…

- **Test Aşaması**
  - Programın olası giriş değerleri için nasıl davrandığı incelenir.
  - Özellikle kritik değerler için programın çalışması test edilmelidir.
  - Hata Ayıklama
    - *Yazım Hatası, Mantıksal Hata, Çalışma Zamanı Hatası*



Yazılım Geliştirme Yaşam Döngüsü…

- **Kurulum ve Bakım Aşaması**
  - Dokümantasyon
    - *Yazılım projesinin her aşamasında yapılan işleri açıklayan dokümanlar hazırlamak gereklidir.*



Yazılım Kalitesi

- Etkinlik
  - Hız ve kaynak kullanımı
- Bakım kolaylığı
- Sağlamlık
  - Hatalardan etkilenme
- Anlaşılabilirlik
- Taşınabilirlik
- Geliştirilebilirlik
- Güvenlik

Programlama Dilleri…

- **Makine Dili (Düşük Seviye)**
  - Geliştirilen ilk programlama dilidir
  - Anlaşılması çok zordur
  - Komutlar, ikili (0 ve 1) sayılardan oluşur.
  - Biraz daha kolay okunabilmesi için komutlar 16'lık sayı sistemi (hex) ile yazılır fakat derlenme sonrası kod ikilik sisteme çevrilir.
  - İşlemci ikilik tabandaki kodu okur ve uygular.
  - Makine (yani, donanım) bağımlıdır.
    - +1300042774
    - +1400593419
    - +1200274027

Programlama Dilleri…

- **Assembly Dili (Orta Seviye)**
  - İkinci kuşak dillerdir
  - Anlaşılması nispeten daha kolaydır
  - İngilizce kısaltmalardan oluşan semboller kullanılır
  - Makine bağımlıdır
  - Assembler derleyecisi ile makine koduna dönüştürülür
    - LOAD sayi1
    - ADD sayi2
    - STORE toplam

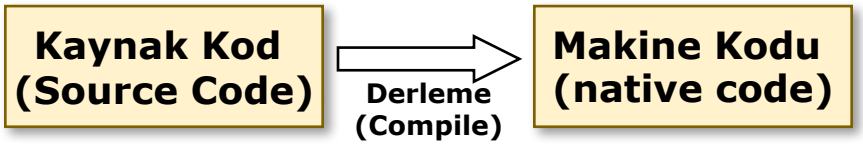
Programlama Dilleri…

- **Yüksek Seviyeli Diller**
  - Anlaşılması çok daha kolaydır
  - Yazılan programlar farklı makinelerdede kullanılabilirler.
  - Assembly dili veya makine dilindeki birçok satır tek bir komutla gösterilir.

    Böylece program daha kısa bir sürede yazılır.
  - Komutlar İngilizce kelimeler ve matematiksel ifadelerden oluşur
  - Pascal, Basic, Fortran, C/C++, Java, vs.
  - Derleyici tarafından makine koduna dönüştürülür
  - Toplam = sayi1 + sayi2

Yazılımların Çalıştırılması…

- **Derleme**
  - Derlenen Dil / Compiled Language
  - Bir programlama dilinde yazılan kodu başka bir kod haline dönüştüren programlar
  - Pascal, C, C++, …



- **Yorumlama**
  - Yorumlanan Dil / Interpreted Language
  - Komutlar teker teker çalıştırılır
  - Komut okuma ve çevirme işlemi çalışma zamanında yapılır (düşük hız)
  - Hata düzeltme daha kolaydır
  - Derleyiciden kaynaklanan sınırlamalar kalktığı için daha esnek bir çalışma ortamı sunar
  - Perl, TCL, Basic, MATLAB, …
- Basitçe, bir kaynak kodu hedef koda çevirdikten sonra çalıştıran ve dolayısıyla koddaki hataları yakalama işlemini ve kodun iyileştirilmesini daha kod çalıştırmadan yapan çeviricilere **derleyici**, kodu satır satır veya bloklar halinde çalıştırıp sırası gelmeyen satırları hiç çalıştırmayan ve bu satırlardaki hataları hiçbir zaman göremeyen ve kodun bütününe ait iyileştirmeleri yapamayan çeviricilere de **yorumlayıcı** denir.

Yazılımların Çalıştırılması…

- **Karma (Hybrid)**
  - Genelde bir ortam yazılan dilin çalıştırılmasına kadar geçen sürede ya bir derleyici yada bir yorumlayıcı kullanılmaktadır. Gelişmekte olan teknolojiyle iki programı birden kullanan diller de türemiştir.
  - Kodlar derlenerek byte koda dönüştürülür
  - Byte kod yorumlanarak çalıştırılır
  - Java, Pyhton
  - Java'da yazılan kod önce derlenerek byte code adı verilen ve sadece java sanal makinalarında (java virtual machine) çalıştırlabilen bir kod üretilmektedir. Bu üretilen ara kod daha sonra java sanal makinasında bir yorumlayıcı yapısına uygun olarak çalıştırılmaktadır.

Programlama Teknikleri

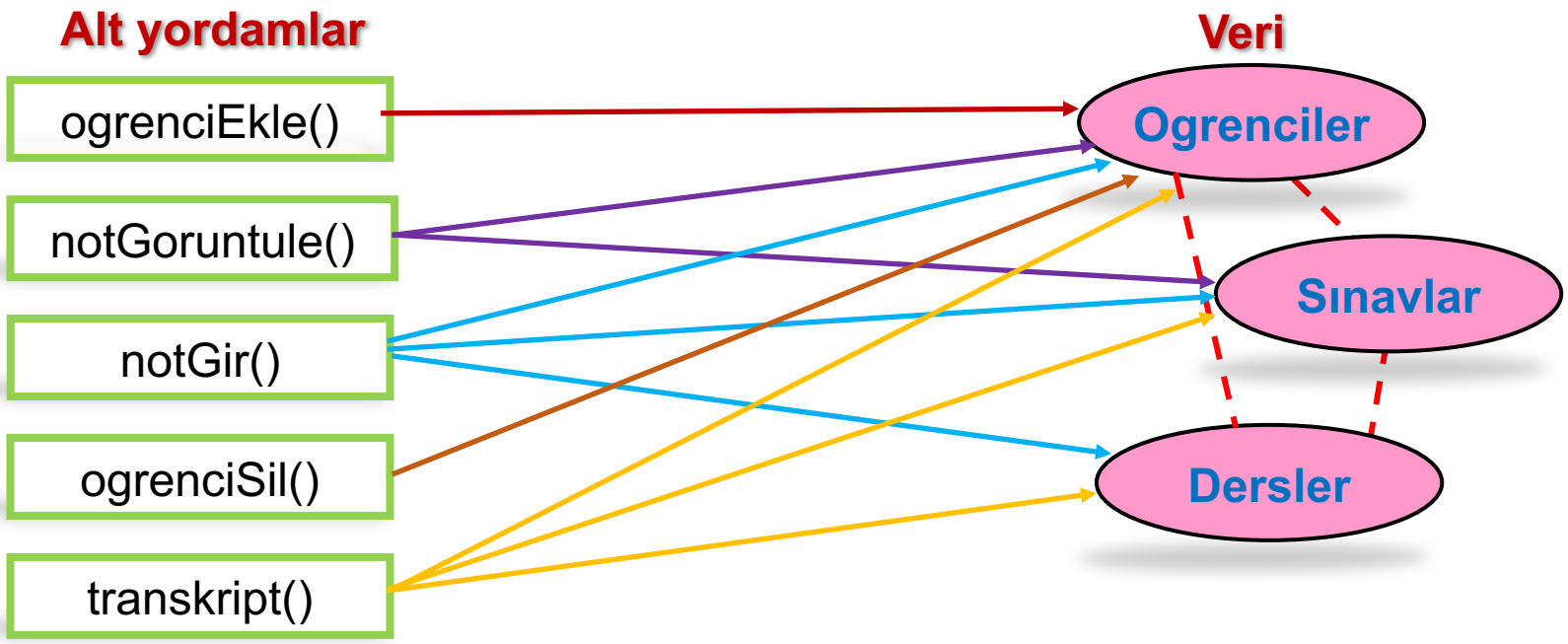
- Yapısal / modüler Programlama tekniği
  - Structured Programming
- Nesne Yönelimli Programlama tekniği
  - Object Oriented Programming

Yapısal/Modüler Programlama

- Programcı doğrudan probleme odaklanır ve problem çözümüne ilişkin fonksiyonları / yöntemleri geliştirir.
- Ana fonksiyon/yöntem (main) programı başlatır ve her biri özel görevleri yerine getiren yöntemler çağrılarak yapılması gerek işlem gerçekleştirilir
- Yöntemlerin ihtiyaç duyduğu veriler genellikle veritabanlarında ya da (her taraftan erişilebilen) değişkenlerde saklanır.

Yapısal/Modüler Programlama…

- Öğrenci, açılan dersler, not girişleri, vb. işlemleri gerçekleştiren bir bilgi sistemi düşünelim:

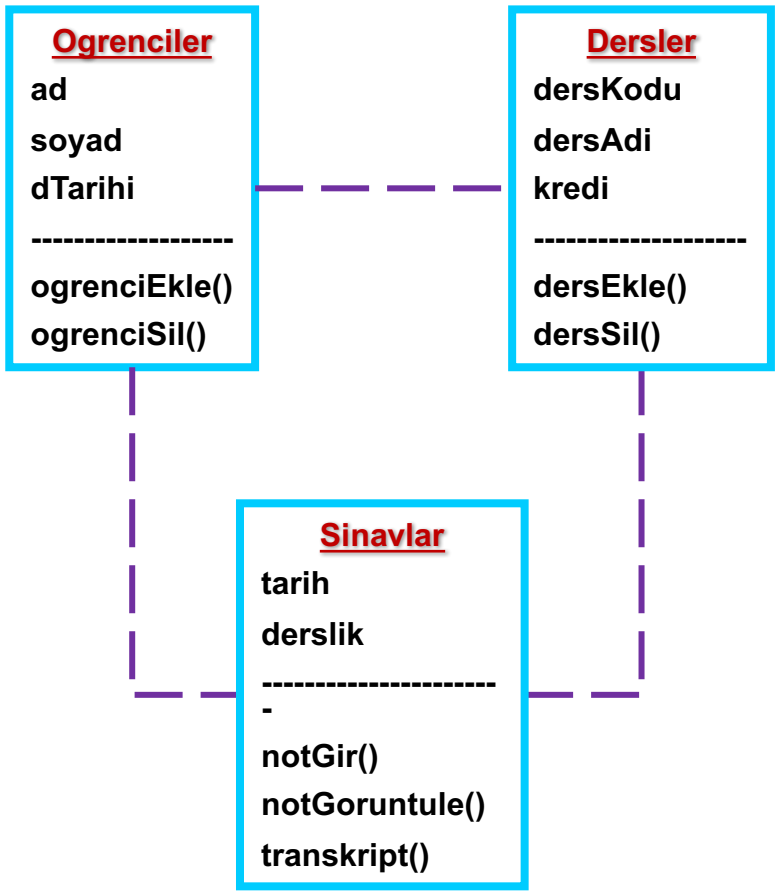


Yapısal/Modüler Programlama…

- Sistem büyüdükçe ilişkiler ve bağımlılık daha da karmaşıklaşır
- Hata bulma zorlaşır
- Program içerisinde değiştirme, ekleme, çıkarma, vs. yapmak zorlaşır ve beklenmeyen etkilere neden olabilir
  - Öğrenciler tablosunda öğrencinin doğum tarihi iki haneli olsun
  - Bu alanı dört haneli yapmak istediğimizi varsayalım
  - Öğrenciler tablosu sınavlar ve dersler tabloları ile ilişkili olduğundan beklenmeyen bir sorun çıkabilir
  - Tüm yöntemler öğrenciler tablosunu bir şekilde kullanıyor
  - O nedenle ogrenciEkle() yöntemi kesinlikle sorun çıkaracaktır

Nesneye Dayalı Programlama

- Yapısal teknikte programcı doğrudan probleme odaklanır ve problemin çözümüne ilişkin yöntemleri geliştirir. Nesneye yönelimli programlama tekniğinde ise temel bileşen nesnedir ve programlar nesnelerin birlikte çalışmasından meydana gelir. Nesne hem veriyi hem de bu veriyi işleyen fonksiyonları içerir. Programcılar dikkatlerini nesneleri oluşturan sınıfları geliştirmeye yoğunlaştırır.
- Yapısal teknikte bir fonksiyon herhangi bir görevi yerine getirmek için veriye ihtiyaç duyarsa, gerekli veri parametre olarak gönderilir. NDP de ise yerine getirilmesi gereken görev nesne tarafından icra edilir ve fonksiyonlar verilere parametre gönderimi yapılmaksızın erişebilirler.



Nesne Yönelimli Programlama…

- **Özellikleri / Avantajları**
  - Problemler daha kolay tanımlanıp çözülebilir. Gerçek dünya düşünülerek geliştirilmiştir. Geliştirme süreci daha kolay olur.
  - Bilgi Gizleme (Information Hiding, Data abstraction)
    - Nesne içerisindeki işlemler (nasıl) diğer nesnelerden soyutlanır-sadece ne yapacağını bilirler. Nesne içerisindeki değişiklik diğer nesneleri etkilemez. Dolayısıyla bakım aşaması daha kolay olur.
    - Gereksiz ayrıntılarla uğraşılmaz, probleme odaklanılır (arabanın gitmesi için gaza basmak yeterli). Daha hızlı geliştirme süreci sağlar.
  - Modüler Programlama (Moduler Programming)
    - Nesneler birbirinden tamamen bağımsız (veri + fonksiyon) (encapsulation, responsibility driven design)
    - Büyük ve karmaşık bir problem küçük parçalara ayrılarak daha kolay çözülebilir. Geliştirme ve bakım daha kolay olur.
    - Programların geliştirilmesi daha hızlı, geliştirilen bir nesne diğer programlarda da rahatlıkla kullanılabilir. Hata bulma-bakım daha kolaydır (Bisikleti başkasına verdiğimiz zaman da çalışır).
    - Geliştirme sürecinde grup çalışmalarına olanak sağlanır.

Nesne Yönelimli Programlama…

- **Özellikleri / Avantajları…**
  - Kodların Tekrar Kullanımı (Code Reuse)
    - Nesneler başka programlara kolaylıkla aktarılabilir. Bakım ve geliştirme zamanı/maliyeti düşer
    - Kalıtım, Çok şekillilik
  - Hata Bulma – Bakım/Onarım (Maintenance)
    - Bileşenler arasındaki ilişkiler açık olduğundan (veri+fonksiyon aynı yapı içerisinde) güncelleme, hata bulma ve bakım daha kolay
  - Tasarım Desenler (Design Patterns)
- Günümüzdeki en iyi yaklaşım. Gelecekte ?