



# YMH311-Yazılım Tasarım Ve Mimarisi

## Yazılım Tasarımı

**Prof. Dr. Resul DAŞ**

Fırat Üniversitesi Yazılım Mühendisliği Bölümü

## Bu Haftaki Konular

Yazılım Tasarımının Önemi.....	4
Tasarım Kavramları.....	8
Yapısal Tasarım.....	16
Tasarlanması Gereken Ortak Alt Sistemler.....	29
Kullanıcı Arayüz Tasarımı.....	36
Tasarım Kalite Ölçütleri.....	43
Yapışıklık.....	50

# Amaçlar

---

- Tasarımın ne olduğunu ve çeşitli tasarım türlerinin ürünün farklı yönleriyle nasıl ilgilendiğini açıklamak
- Tasarımı bir problem çözme etkinliği olarak sunmak, soyutlama ve modellemenin tasarımdaki rolünü ortaya koymak
- Yazılım yaşam döngüsünde tasarımın yerini belirlemek
- Yazılım mühendisliğinde tasarım metotlarını incelemek



# Yazılım Tasarımının Önemi

---

- Tasarlanmış (designed) bir dünyada yaşıyoruz.
- Tasarım ekonomik olarak öneme sahiptir ve yaşam kalitemizi doğrudan etkiler.
- Yazılım son derece yaygın hale gelmektedir.
- Yazılım tasarımının kalitesinin önemli sonuçları olmaktadır ve yazılım tasarımcıları bunların farkında olmalı, bunları ciddiye almalıdır.

# Giriş

---

- Tasarım, Sistem Analizi çalışması sonucunda üretilen **Mantıksal Modelin Fiziksel Modele dönüştürülme çalışmasıdır.**
- Fiziksel Model geliştirilecek yazılımın;
  - hangi parçalardan oluşacağını,
  - bu parçalar arasındaki ilişkilerin neler olacağını,
  - parçaların iç yapısının ayrıntılarını,
  - gerekecek veri yapısının fiziksel biçimini (dosya, veri tabanı, hash tablosu, vektör, vs.)tasarımını içerir.

# Yazılım Ürünleri

---

Bir yazılım ürünü, müşterinin gereksinim ve isteklerini karşılayan bir veya daha fazla programdan, verilerden, ve destekleyici materyal ve hizmetlerden oluşan bir varlıktır. Bu ürün, tek başına bir ürün olabileceği gibi başka bir ürünün temel bileşeni de olabilir.

# Yazılım Tasarımı Nedir?

---

- Yazılım tasarımcıları da temelde diğer disiplinlerdeki tasarımcıların yaptığı işi yapar.
- Tasarlanan şey bir yazılım ürünüdür.

**Yazılım tasarımı**, müşterinin gereksinim ve isteklerini karşılayan yazılım ürününün doğasını ve bileşimini belirleme etkinliğidir.

# Tasarım Kavramları

## Soyutlama (abstraction):

- **Soyutlama (abstraction):** Detayları gizleyerek yukarıdan bakabilme imkanı sağlar.



Soyutlama kavramı veri, işlev ve yapısal açılar için geçerlidir. Örneğin bir kapı nesne olarak ele alındığında onun kulpu, rengi menteşeleri, malzemesi gibi detayları düşünmeden kapıyı bir ev mimarisi içinde değerlendirebiliriz. Aksi taktirde diğer detaylara yoğunlaşan bir tasarımcı 'oda' düzeyinde görsel canlandırmalara hakim olamaz.



# Tasarım Kavramları

## İyileştirme (enhancement):

- **İyileştirme (enhancement):** Soyutlama düzeyinde irdeleme bittikten sonra, daha alt seviyelere inilerek tanımlamalarda ayrıntı, bazen de düzeltme yapılarak **tasarımın daha kesinlik kazanması sağlanır.**

Kulp	Menteşe
Renk	Malzeme

Soyutlama kavramı veri, işlev ve yapısal açılar için geçerlidir. Örneğin bir kapı nesne olarak ele alındığında onun kulpu, rengi menteşeleri, malzemesi gibi detayları düşünmeden kapıyı bir ev mimarisi içinde değerlendirebiliriz. Aksi taktirde diğer detaylara yoğunlaşan bir tasarımcı 'oda' düzeyinde görsel canlandırmalara hakim olamaz.

# Tasarım Kavramları

## Modülerlik (modularity):

- **Modülerlik (modularity):** Sistemi istenen kalite faktörleri ışığında parçalara ayırıştırma sonucu elde edilir. Bir işlev için sistemin tümü değil, ayrılmış bir kısmı üzerinde çalışma yapabilme olanağı sağlar.

Yuvarlak Köşeli Uzun Kısa ...	Yay Dil Vidalar ...
Ahşap Metal Cam ...	Beyaz Metalik Kahverengi ...

Kapı ve pencerenin de kendi ayrıntılarını, birer kelime ile soyutladığımız isimleri içerisinde saklamaları, onları birer neşene olarak bir oda içerisinde 'modüler' bir yapı düzeninde olabilmelerini sağlar. Bir pencerenin yerini değiştirmek, tasarım esnasında onun camı, menteşesi ve malzemesi gibi detayından bağımsız, aynı zamanda da odadaki diğer 'modüllerden' hemen hemen bağımsız olarak ele alınabilir.

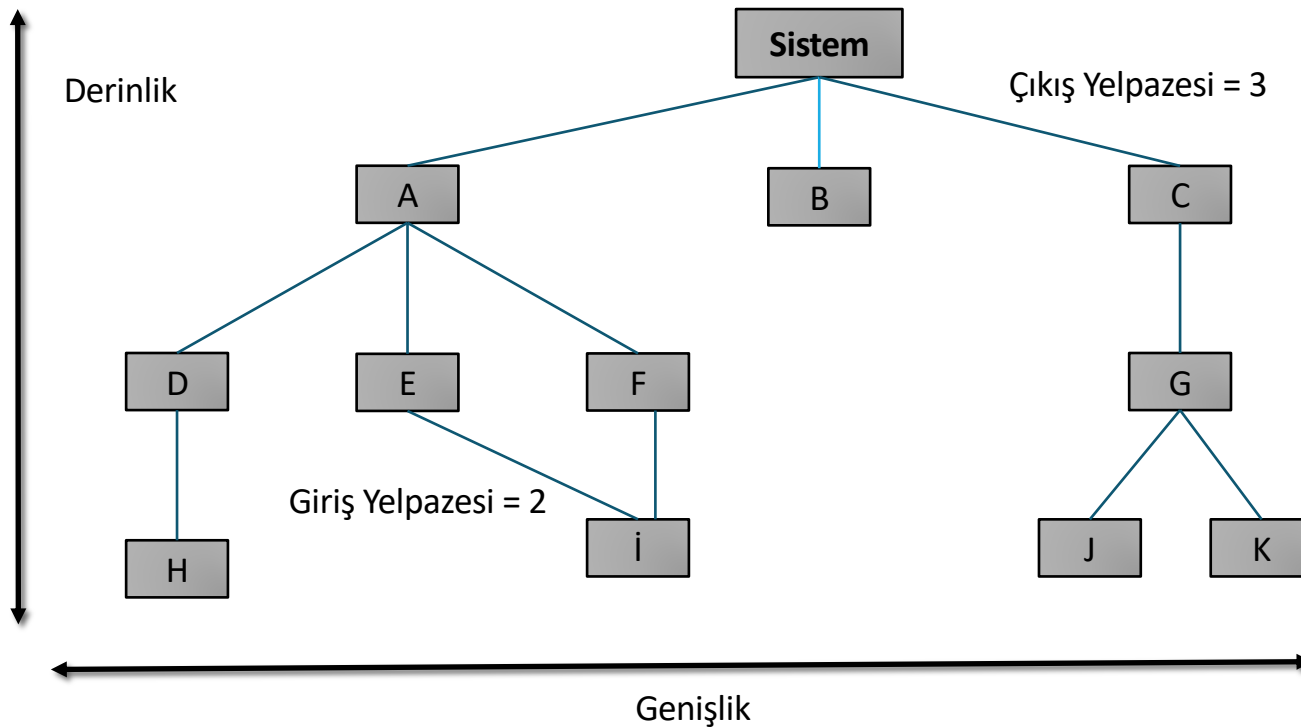
# Modülerlik

---

- Bütün karmaşıklığın tek bir modülde toplanması yerine, anlaşılabilir ve dolayısıyla projenin zihinsel kontrol altında tutulması için sistem bir çok modüle ayrılır.
- Modüller, isimleri olan tanımlanmış işlevleri bulunan ve hedef sistemi gerçekleştirmek üzere tümleştirilen birimlerdir.



# Sistem ve Modülleri



# İşlevsel Bağımsızlık

---

- Modüllere parametre ile veri gönderilir ve sonuç değer alınır. Bu modülü çağıran program parçası sadece bu sonucu kullanabilir. Çağrılan modülün işlevsel olarak yaptıkları ile ilgili değildir.



# Veri Tasarımı

---

- Yapı Tasarımı, arayüz tasarımı ve süreç tasarımından önce yapılması gereken ilk tasarım veri tasarımıdır.
- Bilgi saklama ve soyutlama bu işlem için önemli kavramlardır.



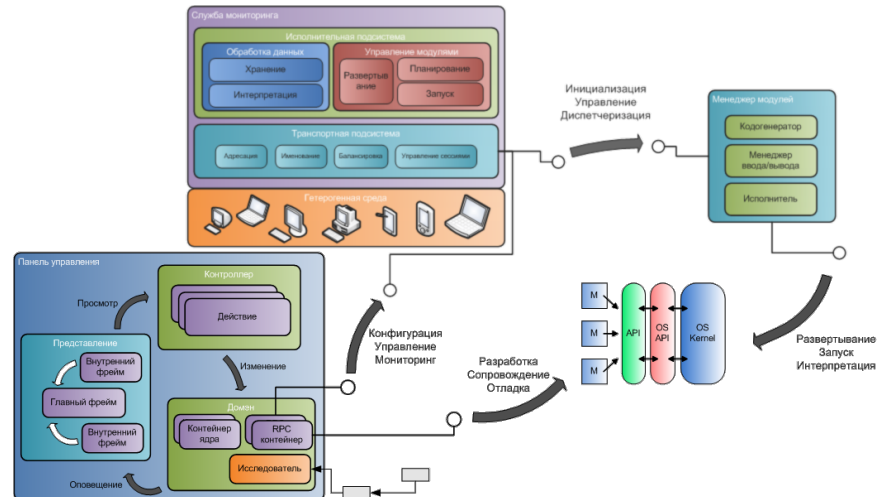
# Veri Tasarımında Dikkat Edilecek Konular

---

- Değişik veri yapıları değerlendirilmelidir.
- Bütün veri yapıları ve bunlar üzerinde yapılacak işlemler tanımlanmalıdır.
- Alt düzeyde tasarım kararları tasarım süreci içerisinde geciktirilmelidir.
- Bazı çok kullanılan veri yapıları için bir kütüphane oluşturulmalıdır.
- Kullanılacak programlama dili soyut veri tiplerini desteklemelidir.

# Yapısal Tasarım

- Yapısal Tasarımın ana hedefi modüler bir yapı geliştirip modüller arasındaki kontrol ilişkilerini temsil etmektir.
- Ayrıca yapısal tasarım bazen de veri akışlarını gösteren biçime dönüştürülebilir.
- Veri Akışları Üç parçada incelenebilir
  - Girdi Akışı
  - Çıktı Akışı
  - İşlem Akışı





# Ayrıntı Tasarım- Süreç Tasarımı

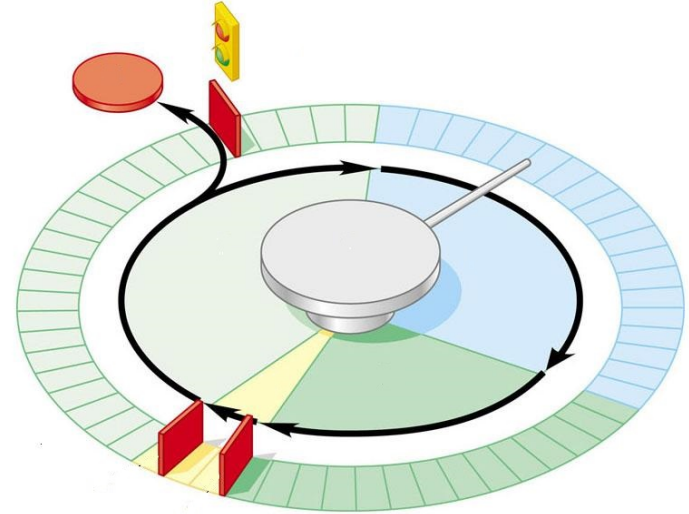
---

- Süreç tasarımı; veri, yapı ve ara yüz tasarımından sonra yapılır.
- İdeal şartlarda bütün algoritmik detayın belirtilmesi amaçlanır.
- Ayrıca süreç belirtiminin tek anlamı olması gerekir, değişik şahıslar tarafından farklı yorumlanmamalıdır.
- Doğal diller kullanılabilir (açıklamalarda, çünkü doğal dil tek anlamlı değildir)
- Süreç Tanımlama Dili (PDL)



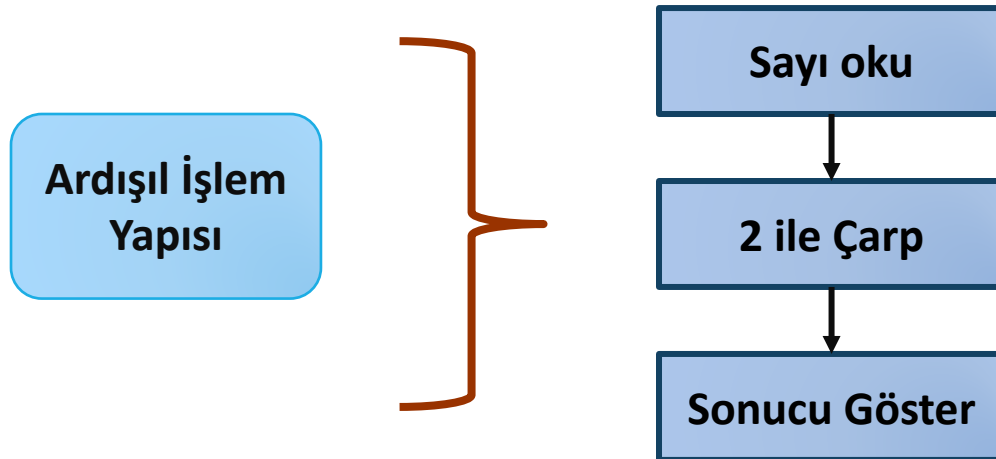
# Yapısal Program Yapıları

- Yapısal programlamanın temel amacı;
  - program karmaşıklığını en aza indirmek,
  - program anlaşılabilirliğini artırmaktır.
- Bu amaçla şu yapıları kullanılır;
  - Ardışıl İşlem yapısı
  - Koşullu işlem yapısı
  - Döngü yapısı
- GOTO kullanımı uygun değildir.



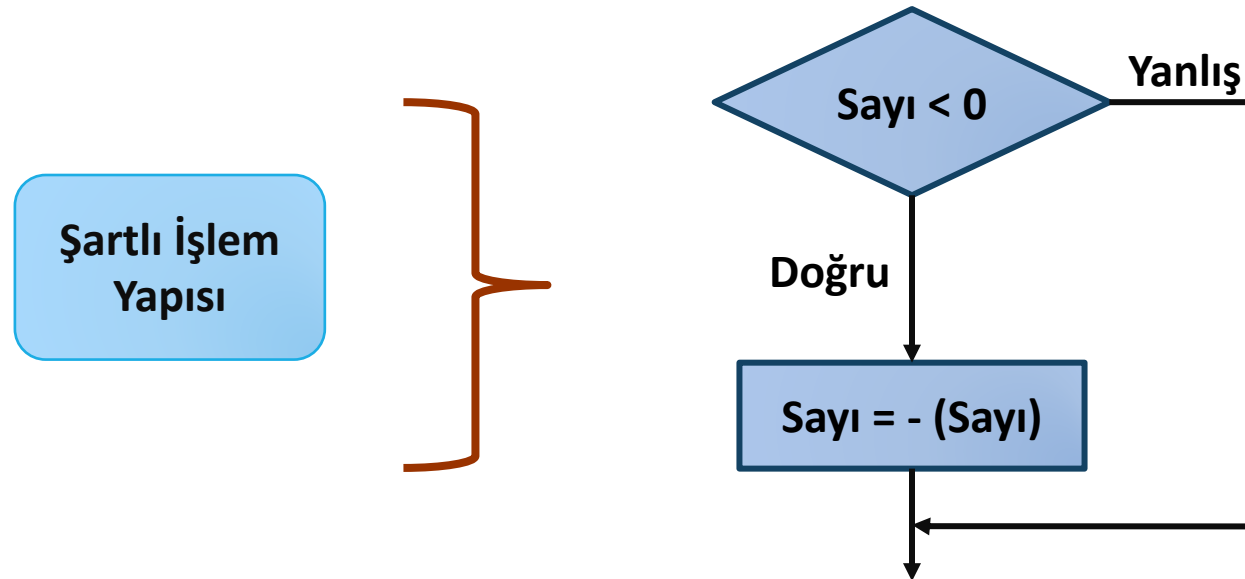
# Program Akış Diyagramı Yapıları

---

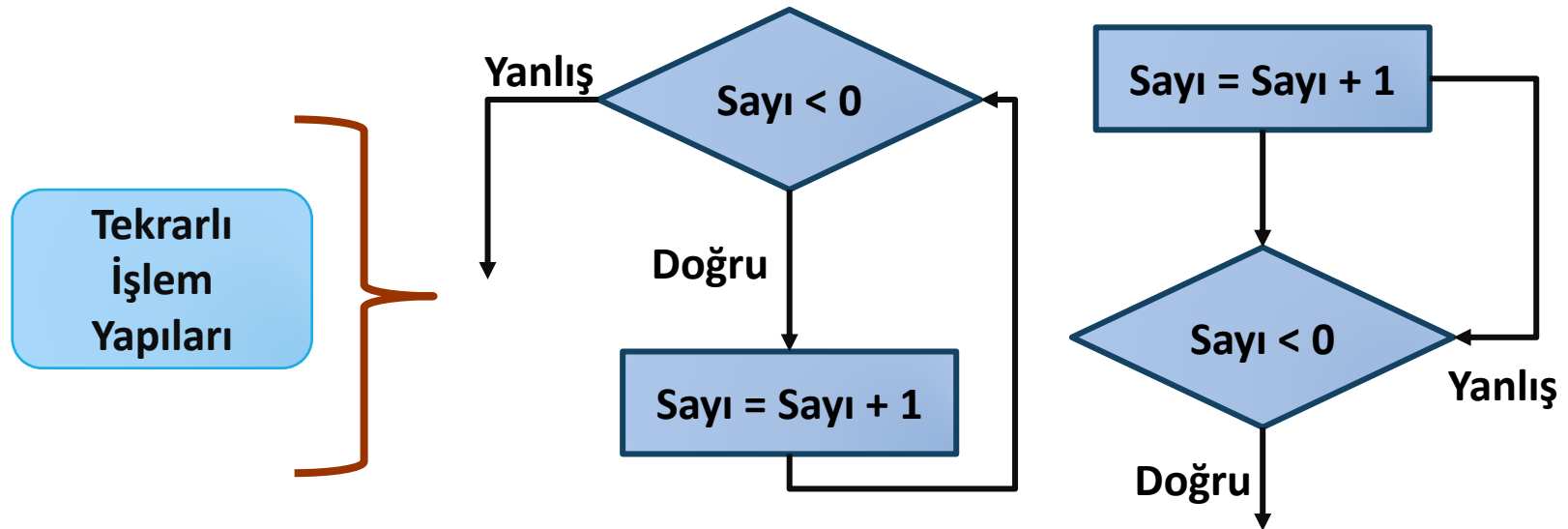


# Program Akış Diyagramı Yapıları

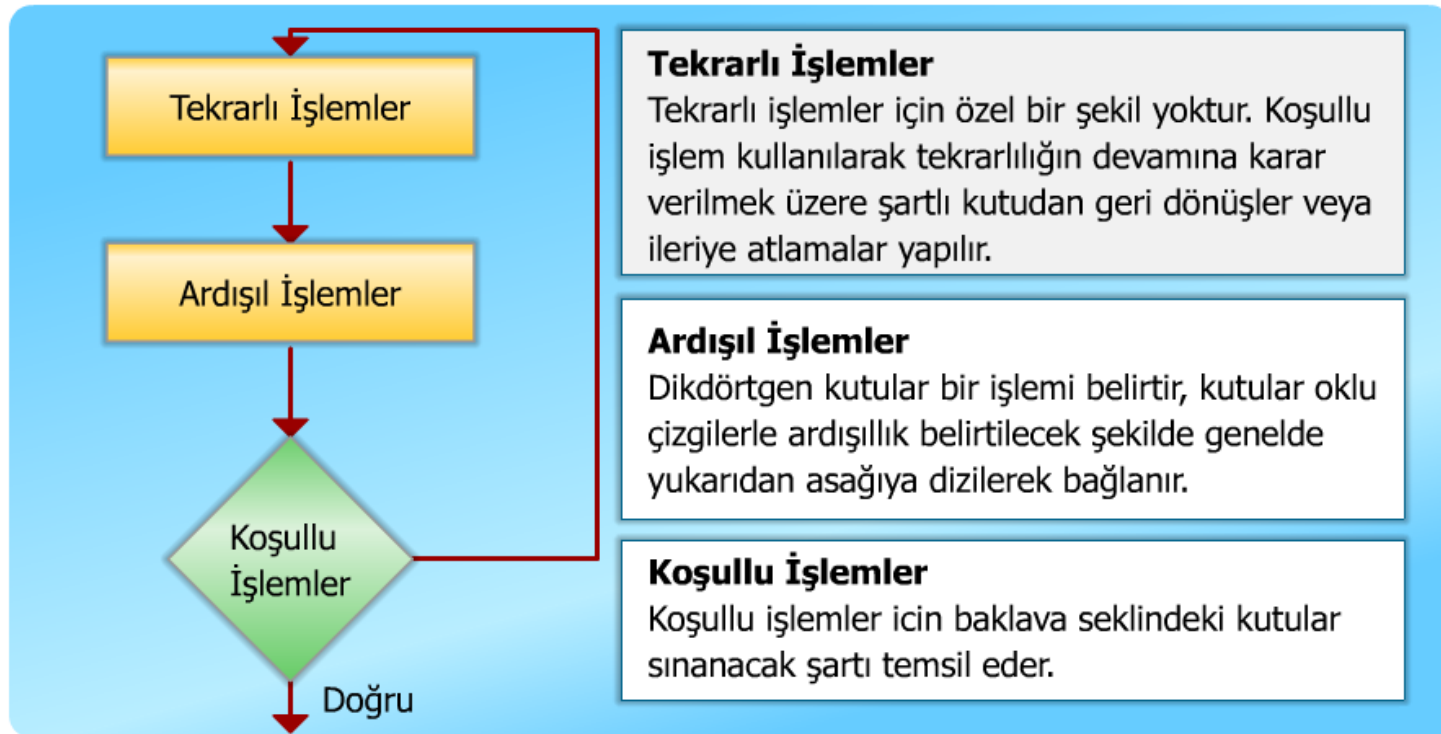
---



# Program Akış Diyagramı Yapıları

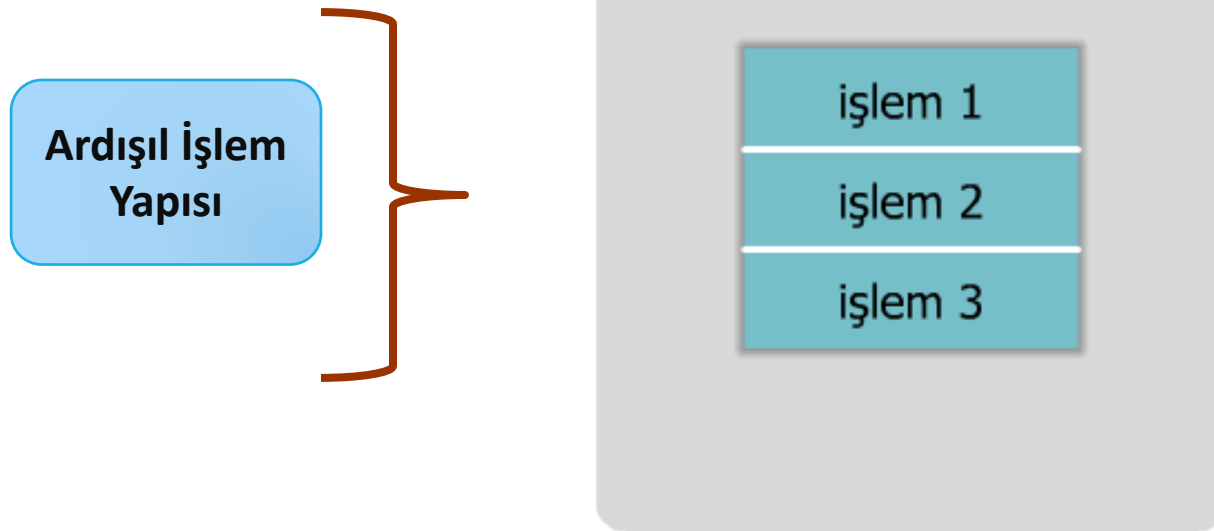


# Program Akış Diyagramları



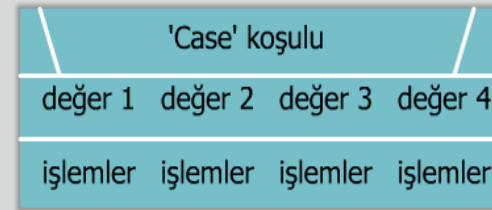
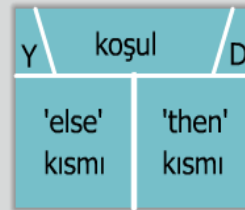
# Kutu Diyagramları

---



# Kutu Diyagramları

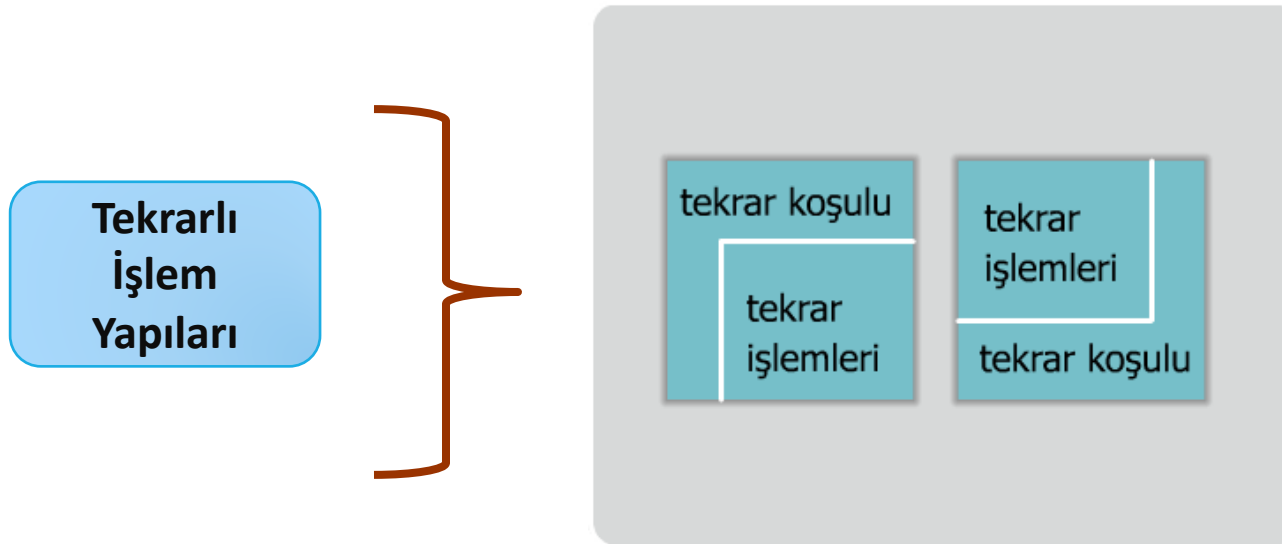
## Şartlı İşlem Yapısı





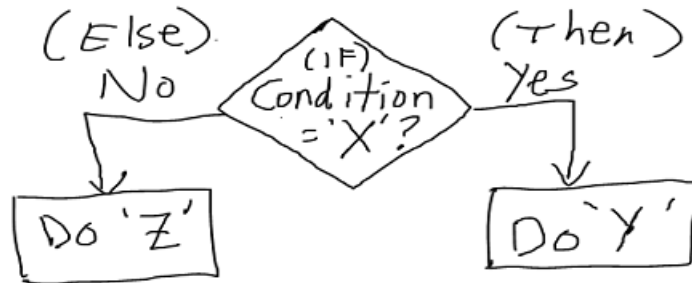
# Kutu Diyagramları

---



# Karar Tabloları

- Bazen karmaşık koşul değerlendirmeleri yapmak gerekir. Bunların düzenli bir gösterilimi karar tablolarında yapılabilir.
- Öncelikle, bütün işlemler saptanmalı, sonra ön koşullar belirlenmelidir.
- Belirli işlemler ile belirli koşulları birleştirerek tablo oluşturulur.
- Alt tarafta ise işlemler benzer satırlar olarak gösterilir.



# Karar Tabloları

---

Kurallar	1	2	3	4
Hesap Geçerli	✓	✓	✓	✓
Şifre Geçerli	✓		✓	
Yeterli Nakit Var	✓		✓	✓
Hesapta Yeterli Bakiye Var	✓			✓
Bakiye Bildirme İşlemi	✓			
Para Çekme İşlemi		✓		
Para Yatırma İşlemi			✓	
Para Gönderme İşlemi				

# Program Tasarım Dili

---

- Program Tasarım Dilleri süreç belirtiminde doğal dillerin programlama dili ile sentezlenmesi şeklinde ortaya çıkmıştır.
- Hangi programlama dilinin kullanılacağından bağımsız özellikler bulunmalıdır.

DO

Hesap Numarasını Oku

IF (hesap numarası geçerli değil) başlangıca dön  
işlem türünü iste

IF (para yatırma işlemi) { para\_yatir(); Başlangıca dön}

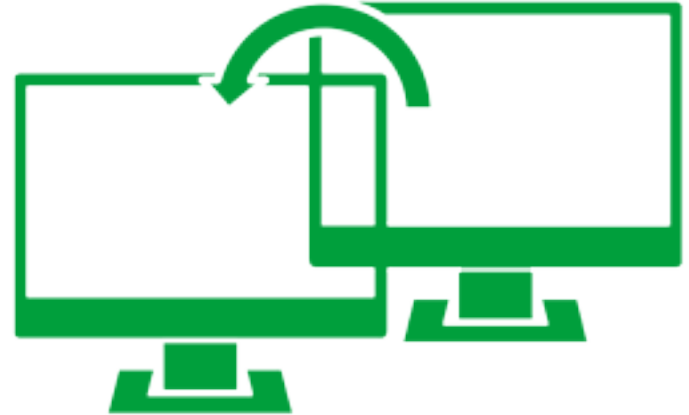
IF (yeterli bakiye yok) başlangıca dön

WHILE

# Tasarlanması Gereken Ortak Alt Sistemler

---

- Yetkilendirme altsistemi
- Güvenlik altsistemi
- Yedekleme altsistemi
- Veri transferi altsistemi
- Arşiv altsistemi
- Dönüştürme altsistemi



# Yetkilendirme Alt Sistemi

---

- Özellikle kurumsal uygulamalarda farklı kullanıcıların kullanabilecekleri ve kullanamayacakları özellikleri ifade eder.
  - İşlev bazında yetkilendirme
  - Ekran bazında yetkilendirme
  - Ekran alanları bazında yetkilendirme
- Oracle veri tabanına erişim konusunda yetkilendirme yapmaktadır.



# Güvenlik Alt Sistemi

---

- Güvenlik alt sistemi, bilgi sisteminde yapılan işlerin ve yapan kullanıcıların izlerinin saklanması ve gereken durumlarda sunulması ile ilgilidir.
- Bir çok yazılım geliştirme ortamı ve işletim sistemi, bu amaca yönelik olarak, "**sistem günlüğü**" olanakları sağlamaktadır.
- Sistem günlüğü ile sunulan olanaklar yeterli olmadığı durumlarda ek yazılımlar geliştirilmesi gerekmektedir.
- LOG files (Sistem günlüğü)

# Yedekleme Alt Sistemi

---

- Her bilgi sisteminin olağandışı durumlara hazırlıklı olmak amacıyla kullandıkları veri tabanı (sistem) yedekleme ve yedekten geri alma işlemlerinin olması gerekmektedir.
- Günümüzde tüm veri tabanı yönetim sistemi geliştirme platformları, oldukça zengin yedekleme ve yedekten geri alma olanakları sağlamaktadır.
- Bu konuda, tasarım bağlamında yapılması gereken, yedekleme işleminin düzenlenmesini tasarlamak olmalıdır.
- Yedeklemenin hangi sıklıkla yapılacağı, ne zaman, elle ya da otomatik olarak yapılıp yapılmayacağı gibi planlamalar, tasarım aşamasında yapılmalıdır.



# Veri İletişim Alt Sistemi

---

- Coğrafi olarak dağıtılmış hizmet birimlerinde çalışan makineler arasında veri akışının sağlanması işlemleri
- **Çevirim içi veri iletimi (real-time):** Verinin bir birimden diğerine anında iletilmesi olarak tanımlanır.
  - Bu tür veri iletişimi, gerçek zamanlı sistemler için oldukça önemlidir.
  - Bilgi sistemi uygulamalarında, - zamansal kritiklik, gerçek zamanlı uygulamalara oranla daha az olduğu için - bu tür iletişim çok yaygın değildir.
- **Çevirim dışı veri iletimi (disketler, teypler):** Bilgilerin iletişim hatları kanalıyla değil, çevrim dışı ortamlar (teyp, disket, cd vb.) aracılığı ile iletilmesi çevrim dışı iletişim olarak tanımlanmaktadır.
  - Kısacası, "kargo" olarak tanımlanan iletişimin, el ile yapılan türüdür.
  - Ağ iletişiminin sağlanamadığı durumlarda kullanılan bir yöntemdir.
  - Kullanımı giderek azalmaktadır.

# Arşiv Alt Sistemi

---

- Belirli bir süre sonrasında sık olarak kullanılmayacak olan bilgilerin ayrılması ve gerektiğinde bu bilgilere erişimi sağlayan alt sistemlerdir.
- Örneğin, insan kaynakları yönetimi bilgi sisteminde, emekli olan bir kişiye ilişkin bilgilerin, çevrim-içi olarak tutulan veri tabanından alınarak, çevrim dışı bir ortama alınması ve aradan örneğin beş yıl geçtikten sonra, pasaport işlemleri için gerek duyulabilecek kişi bilgilerine erişilmesini sağlayan işlemler arşiv alt sistemleri tarafından gerçekleştirilmektedir.
- İşlem türü olarak ortak bir çok özellik içeren arşiv alt sistemleri, uygulama bazında az da olsa farklılaşabilmektedir.
- Aktif veri tabanı.

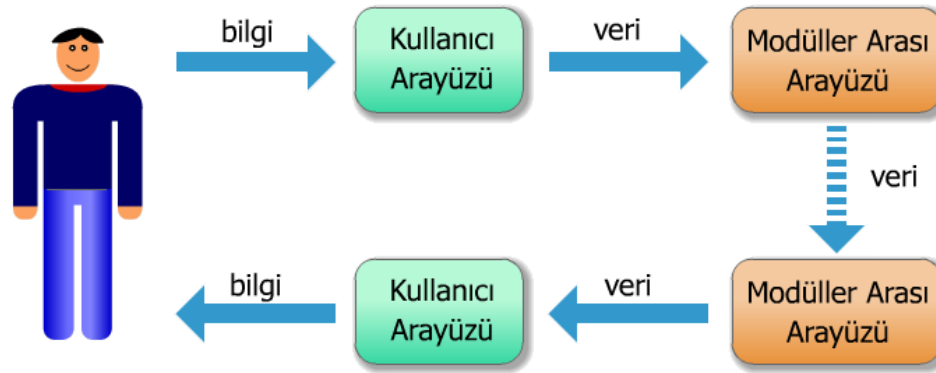
# Dönüştürme Alt Sistemi

---

- Geliştirilen bilgi sisteminin uygulamaya alınmadan önce veri dönüştürme (mevcut sistemdeki verilerin yeni bilgi sistemine aktarılması) işlemlerine ihtiyaç vardır.
- Mevcut uygulamalardaki bilgisayar ortamında saklanan bilgilerin ortam çeşitliliği, dönüştürme işlemlerini zorlaştırır.



# Kullanıcı Arayüz Tasarımı



- Kullanıcı ile ilişkisi olmayan arayüzler
  - Modüller arası arayüz
  - Sistem ile dış nesneler arası arayüz
- Kullanıcı arayüzleri
  - Kullanım kolaylığı ve öğrenim zamanı esastır.
  - Program = arayüz yaklaşımı vardır.

# Genel Prensipler

---

- Komut seçimi, veri giriş formlarının şekli gibi bir çok konuda tutarlı bir yapı izlenmelidir.
- Önemli silmelerde teyit alınmalıdır.
- Yapılan çoğu işlem kolayca geri alınabilmelidir
- İşlemler arasında ezbere tutacak bilgi miktarı azaltılmalıdır.
- Kullanıcı hareketleri, düşünme ve algılamasında verimlilik sağlanmalıdır.
- Hataların affedilmesi, yanlış giriş olduğunda program korunmalı ve düzeltme şansı verilmelidir.
- İşlemleri sınıflandırıp ekran geometrisi buna uygun olarak kullanılmalıdır.
- Komut isimleri kısa ve basit olmalıdır.
- Menülerin ve diğer etkileşimli araçların standart yapıda tasarlanmalıdır.

# Bilgi Gösterimi

---

- Yalnızca içinde bulunulan konu çerçevesi ile ilgili bilgi gösterilmeli
- Veri çokluğu ile kullanıcı bunaltılmamalı, grafik ve resimler kullanılmalı
- Tutarlı başlık, renkleme ve kısaltma kullanılmalı
- Hata mesajları açıklayıcı ve anlaşılır olmalı
- Değişik tür bilgiler kendi içinde sınıflandırılmalı
- Rakamsal ifadelerde analog görüntü verilmeli (%89 değil)



# Veri Girişi

---

- Kullanıcı hareketleri en aza indirilmelidir. Yazma yerine ekrandaki listelerden seçme, bir komuta en az sayıda fare tıklamasıyla erişme gibi.
- Gösterim ve girdi sahaları birbirinden ayırt edecek biçimler (renk, büyüklük, yerleşim vb.) tutarlı olarak kullanılmalıdır.
- Kullanıcı uyarlamasına izin verilmelidir: kullanıcı bazı özellikleri tanımlayabilir, bazı uyarı mesajlarını istemeyebilir.
- Kullanılan konu ile ilgili gereksiz komutlar geçici olarak etkisizleştirilmelidir.
- Bütün girdiler için yardım kolaylıkları olmalıdır.



# Kullanıcı Arayüz Prototipi

---

- Tasarım çalışması sonucunda, daha önceden gereksinim çalışması sırasında hazırlanmış olan kullanıcı arayüz prototipi, ekran ve rapor tasarımları biçimine dönüşür. Ekranlar son halini alır, raporlar kesinleşir. Kullanıcıya gösterilerek onay alınır.
- Tüm programın tek elden çıktığının ifade edilebilmesi açısından tüm ekranların aynı şablon üzerine oturtulması önerilmektedir.
  - Menü Çubuğu
  - Araç Çubuğu
  - Gövde (Değişebilir)
  - Durum Çubuğu



# Başlangıç Tasarım Gözden Geçirme

---

- Yapılan tasarım çalışmasının bir önceki geliştirme aşaması olan analiz aşamasında belirlenen gereksinimleri karşılayıp karşılamadığının belirlenmesidir.
  - Sistem gereksinimlerine yardımcı olan kullanıcılar
  - Sistem analizini yapan çözümleyiciler
  - Sistemin kullanıcıları
  - Tasarımcılar
  - Yönlendirici
  - Sekreter
  - Sistemi geliştirecek programcılardan oluşan bir grup tarafından yapılır.

# Ayrıntılı Tasarım Gözden Geçirme

- Başlangıç tasarımı gözden geçirme çalışmasının başarılı bir biçimde tamamlanmasından sonra, tasarımın teknik uygunluğunu belirlemek için **Ayrıntılı Tasarım Gözden Geçirme** çalışması yapılır. Bu çalışmada;
  - Çözümleyiciler
  - Sistem Tasarımcıları
  - Sistem Geliştiriciler
  - Sekreterden oluşan bir ekip kullanılır.



# Tasarım Kalite Ölçütleri

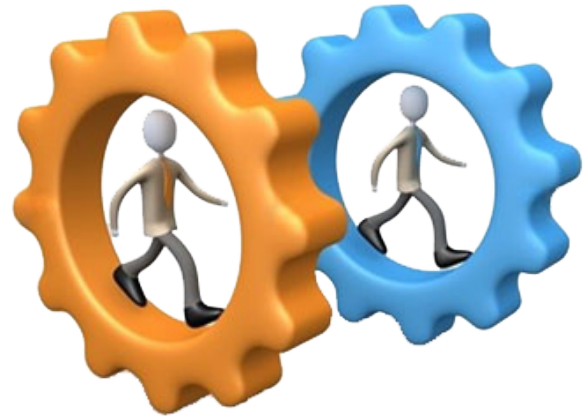
---

- **Bağlaşım (Coupling)**

Tasarımı oluşturan modüller arası ilişki ile ilgilidir.

- **Yapışıklık (Cohesion)**

Modüllerin iç yapısı ile ilgilidir.



# Bağlaşım

---

- Modüller arası bağıllığın ölçülmesi için kullanılan bir ölçüttür.
- Yüksek kaliteli bir tasarımda bağlaşım ölçümü az olmalıdır.
- Bağlaşımın düşük olması
  - Hatanın dalgasal yayılma özelliğinin azaltılması
  - Modüllerin bakım kolaylığı
  - Modüller arası ilişkilerde karmaşıklığın azaltılmasınedenleri ile istenmektedir

# Yalın Veri Bağlaşımı

---

- Herhangi iki modül arası iletişim yalın veriler (tamsayı, karakter, boolean, vs) aracılığı ile gerçekleştiriliyorsa bu iki modül yalın veri bağlaşımlıdır şeklinde tanımlanır.



# Karmaşık Veri Bağlaşımı

---

- Herhangi iki modül arasındaki iletişimde kullanılan parametrelerin karmaşık veri yapısı (kayıt, dizi, nesne, vs) olması durumunda modüller karmaşık veri paylaşımı olarak tanımlanır.



# Denetim Baęlaşımı

---

- İki Modül arasında iletişim parametresi olarak **denetim verisi** kullanılıyorsa bu iki modül denetim baęlaşımli olarak tanımlanır.



# Ortak Veri Bağlaşımı

---

- Eğer iki modül ortak bir alanda tanımlanmış verilere ulaşabiliyorsa bu iki modül **ortak veri bağlaşımlı** olarak tanımlanır.
- Verilerin ortak veri bağlaşımlı olmaları şu nedenlerden dolayı fazla istenmez;
  - Ortak veri alanını izlemek zordur.
  - Ortak veri kullanan modüllerde yapılan değişiklikler diğer modülleri etkiler.
  - Ortak veri üzerinde yapılacak değişikliklerde bu veriyi kullanacak bütün modüller göz önüne alınmalıdır.



# İçerik Bağlaşımı

---

- Modüllerin iç içe tasarlanması sonucu, bir modülün başka bir modül içerisinde tanımlanmış veri alanına erişebilmesi olanaklaşır ve bu durum **içerik bağlaşımına** yol açar.



# Yapışıklık

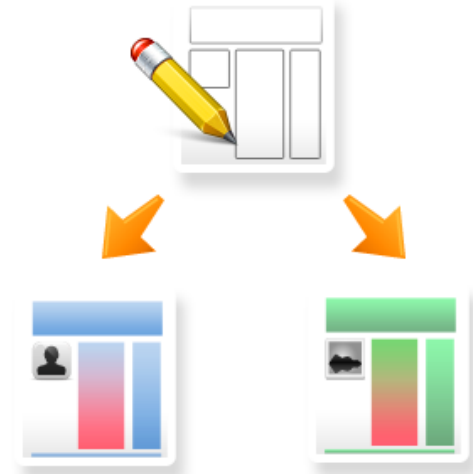
---

- Bir modülün kendi içindeki işlemler arasındaki ilişkilere ilişkin bir ölçüttür. **Modül gücü** olarak ta tanımlanır.
- Tasarımda **yapışıklık** özelliğinin yüksek olması tercih edilir.
- Yapışıklık ile Bağlaşım ters orantılıdır.

# İşlevsel Yapışıklık

---

- İşlevsel Yapışık bir modül, tek bir iş problemine ilişkin sorunu çözen modül olarak tanımlanır.
- Maas\_Hesapla, Alan\_Hesapla gibi



# Sırasal Yapışıklık

---

- Bir modülün içindeki işlemler incelendiğinde, bir işlemin çıktısı, diğer bir işlemin girdisi olarak kullanılıyorsa bu modül **sırasal yapışık** bir modül olarak adlandırılır.

Ham\_Veri\_Kaydını\_Düzeltil

Düzeltilmiş\_Ham\_Veri\_Kaydını\_Dogrula

Dogrulanmis\_Kaydi\_Gonder

# İletişimsel Yapışıklık

- Bir modülün içindeki farklı işlemler aynı girdi ya da çıktıyı kullanıyorlarsa bu modül **iletişimsel yapışık** bir modül olarak adlandırılır.

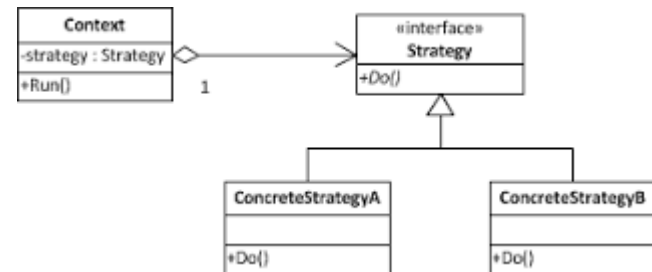
Sicil\_No\_yu\_Al  
Adres\_Bilgisini\_Bul  
Telefon\_Bilgisini\_Bul  
Maas\_Bilgisini\_Bul



# Yordamsal Yapışıklık

- Yordamsal Yapışık modüldeki işlemler arasında denetim ilişkisi bulunmaktadır.
- İşlemlerin birbirleri ile veri ilişkisi yoktur, ancak işlem sırası önemlidir.

Ekran\_Goruntusunu\_Yaz  
Giris\_Kaydini\_Oku



# Zamansal Yapışıklık

---

- Bir modül içindeki işlemlerin belirli bir zamanda uygulanması gerekiyor ve bu işlemlerin kendi aralarında herhangi bir ilişkisi yok, yani işlemlerin sırası önemli değil ise, **zamansal yapıışıklık** vardır.

Alarm\_Zilini\_Ac  
Kapiyi\_Ac  
Kamerayi\_Calistir

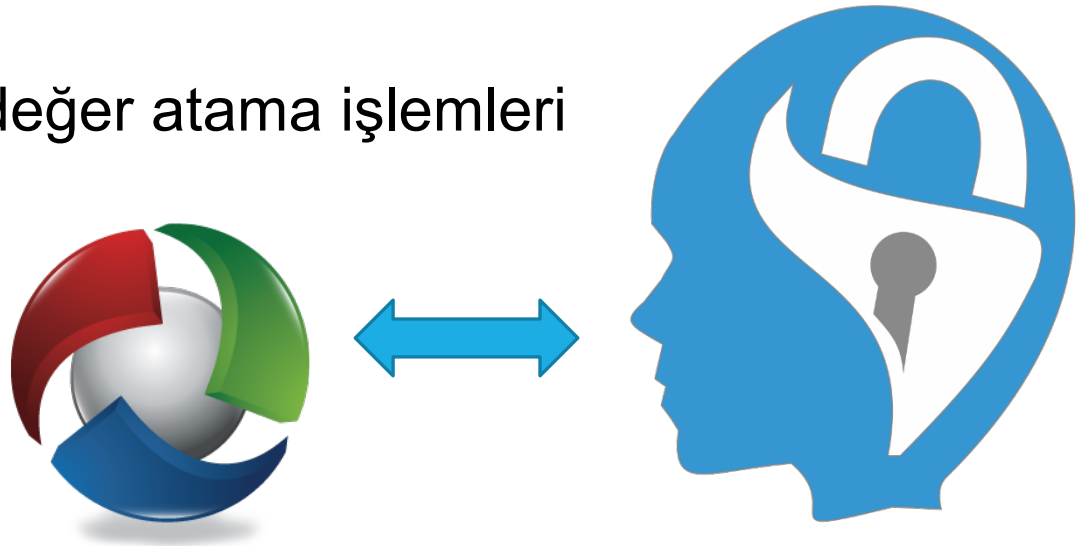


# Mantıksal Yapışıklık

---

- Mantıksal olarak aynı türdeki işlemlerin bir araya toplandığı modüller **mantıksal yapışık** olarak adlandırılır.

Dizilere değer atama işlemleri





# Gelişigüzel Yapışıklık

---

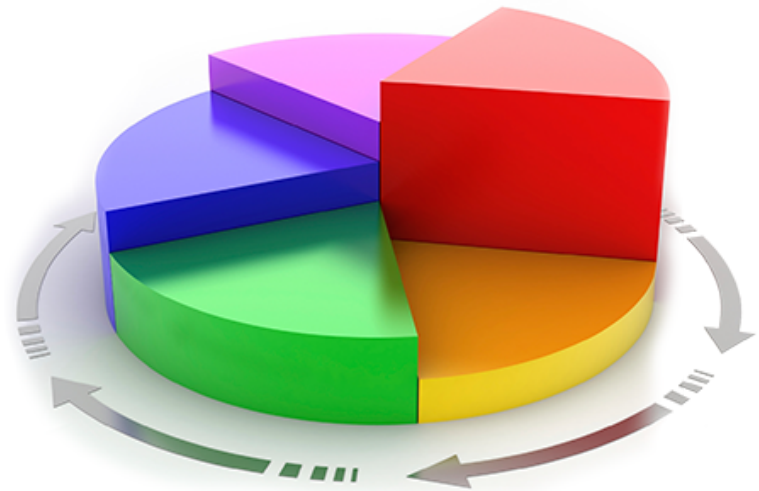
- İşlemler arasında herhangi bir ilişki bulunmaz.

Ara\_Kayit\_Oku

B\_dizisine\_baslangic\_de

Stok\_kutugu\_oku

Hata\_iletisi\_yaz



# Özet

---

- Tasarım, Sistem Analizi çalışması sonucunda üretilen Mantıksal Modelin Fiziksel Modele dönüştürülme çalışmasıdır.
- Tasarım kavramları: Soyutlama, İyileştirme ve Modülerlik olmak üzere 3 çeşittir.
- Yapı Tasarımı, arayüz tasarımı ve süreç tasarımından önce yapılması gereken ilk tasarım veri tasarımıdır.
- Veri Akışları Üç parçada incelenebilir: Girdi Akışı, Çıktı Akışı ve İşlem Akışı
- Süreç tasarımı; veri, yapı ve ara yüz tasarımından sonra yapılır.
- Program Akış Diyagramları: Tekrarlı, ardışıl ve koşullu şeklindedir.

Tasarlanması Gereken Ortak Alt Sistemler;

- Yetkilendirme altsistemi
- Güvenlik altsistemi
- Yedekleme altsistemi
- Veri transferi altsistemi
- Arşiv altsistemi
- Dönüştürme altsistemi      şeklindedir.

# Sorular

---

1. Yazılım tasarım sürecinin temel işlemlerini sayınız. E-R diyagramı çizerek ilişkilerini gösteriniz.
2. Geliştireceğiniz bir uygulama için ekran şablonunuzu belirleyiniz.
3. Bağlaşım ve yapışıklık kavramlarını açıklayınız. Program bakımı ile ilişkilerini belirtiniz.
4. Bir sistem tümüyle bağışımsız biçimde tasarlanabilir mi? Yani sistemin tüm modülleri arasında hiç bağlaşım olmadan tasarım yapılabilir mi?
5. Tümüyle işlevsel yapışık modüllerden oluşan bir sistem tasarlanabilir mi? Neden yapılabilir? Neden yapılamaz?
6. Bağlaşım ile Yazılım Taşınabilirliği arasındaki ilişkiyi belirtiniz.
7. Tasarım gözden geçirmenin önemi nedir? Yapılmaması ne tür sonuçlara yol açar?
8. Arşiv alt sistemi ile yedekleme alt sistemi arasındaki benzerlikleri ve farklılıkları belirtiniz.
9. Tasarım ile sınaama arasındaki ilişkiyi belirtiniz.
10. 13. Geliştirdiğiniz bir uygulama tasarımı için
  - (a) Bir birim sınaama belirtimi
  - (b) İki sistem sınaama belirtimi (senaryo)      hazırlayınız.

# Kaynaklar

---

“Software Engineering A Practitioner’s Approach” (7th. Ed.), Roger S. Pressman, 2013.

“Software Engineering” (8th. Ed.), Ian Sommerville, 2007.

“Guide to the Software Engineering Body of Knowledge”, 2004.

” Yazılım Mühendisliğine Giriş ”, TBİL-211, Dr. Ali Arifoğlu.

” Yazılım Mühendisliği ” (2. Basım), Dr. M. Erhan Sarıdoğan, 2008, İstanbul: Papatya Yayıncılık.

Kalıpsız, O., Buharalı, A., Biricik, G. (2005). Bilgisayar Bilimlerinde Sistem Analizi ve Tasarımı Nesneye Yönelik Modelleme. İstanbul: Papatya Yayıncılık.

Buzluca, F. (2010) Yazılım Modelleme ve Tasarımı ders notları (<http://www.buzluca.info/dersler.html>)

Hacettepe Üniversitesi BBS-651, A. Tarhan, 2010.

Yazılım Proje Yönetimi, Yrd. Doç. Dr. Hacer KARACAN

# Kaynaklar

---

“Software Engineering A Practitioner’s Approach” (7th. Ed.), Roger S. Pressman, 2013.

“Software Engineering” (8th. Ed.), Ian Sommerville, 2007.

“Guide to the Software Engineering Body of Knowledge”, 2004.

” Yazılım Mühendisliğine Giriş”, TBİL-211, Dr. Ali Arifoğlu.

”Yazılım Mühendisliği” (2. Basım), Dr. M. Erhan Sarıdoğan, 2008, İstanbul: Papatya Yayıncılık.

Kalıpsız, O., Buharalı, A., Biricik, G. (2005). Bilgisayar Bilimlerinde Sistem Analizi ve Tasarımı Nesneye Yönelik Modelleme. İstanbul: Papatya Yayıncılık.

Buzluca, F. (2010) Yazılım Modelleme ve Tasarımı ders notları (<http://www.buzluca.info/dersler.html>)

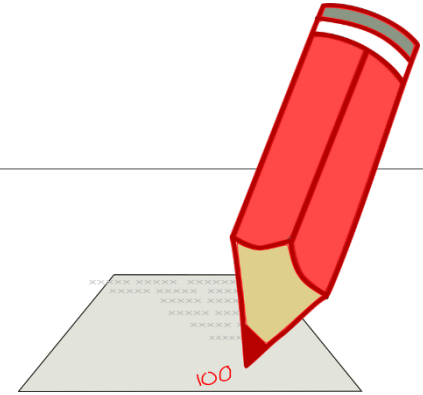
Hacettepe Üniversitesi BBS-651, A. Tarhan, 2010.

Yazılım Proje Yönetimi, Yrd. Doç. Dr. Hacer KARACAN

# Ödev

---

Yazılım Tasarım Kalıpları hakkında araştırma yapınız.



# Sorularınız

---

