Nesne Yönelimli Programlama

1.Hafta

Konu: Genel Bakış

Dr. Öğr. Üyesi Güncel SARIMAN
E-posta: guncelsariman@mu.edu.tr

Dersi Amacı ve Hedefi

- Nesneye Yönelik Programlama (Object Oriented Programming) kavramlarını ve yöntemlerini tanımak.
- Kavramlar:
  - Soyutlama, Saklama, Kalıtım, Sarmalama, Çok biçimlilik, Nesne, Sınıf , Soyut Sınıf
- Bu yöntemlerin kaliteli yazılımlar üretmek için etkin biçimde nasıl kullanılacağını C# programlama dili ile göstermek.

Dersi Amacı ve Hedefi

- Bu dersi başarıyla tamamlayabilen öğrenciler şunları yapabilecektir;
  - Soyut düşünme farkındalığı,
  - Bir problemi nesneler, nesnelerin işlevleri ve nesneler arasındaki ilişkilere dönüştürebilme,
  - Nesneye yönelik programlama mantığı ile bir problemi modelleme,
  - Nesneye yönelik programlama tekniklerini kullanarak küçük ve orta seviye yazılım uygulamaları geliştirebilme.

Dersin Kaynakları

- Ders Kaynakçası olarak internetten erişebileceğiniz güvenilir Nesne Yönelimli Programlama ile ilgili yurt içi yurt dışı akademisyenlerin ders notlarından, online ders eğitim sitelerinden yararlanabilirsiniz.

- Ders Kitabı olarak ise
  - Tasarım Desenleri ve Mimarileri Yazar: Ali Kaya,Engin Bulut,Pusula Kitaplık 2016
  - HER YÖNÜYLE C# 5.0 , Volkan Aktaş, Kodlab Yayıncılık, İstanbul, 2013

Haftalık Ders İçeriği

	Hafta İçerikleri
Hafta 1	Ders Tanıtımı, Video Gösterimleri, Neden OOP?,UML Nedir?
Hafta 2	C# Programlama Dili Temel Bilgileri. Visual Studio IDE ile Form Yaratma Toolbox Kullanarak Buton, Label ve TextBoxt ekleme
Hafta 3	C# Programlama Dili Temelleri ile Nesneye Dayalı Programlamaya Geçiş Veri tipleri, Değişkenler ve Sabitler Tip Dönüşümleri, Operatörler Akış Kontrol Mekanizmaları ve Döngüler Diziler, Fonksiyonlar
Hafta 4	Nesneye Yönelik Programlama Tarihçesi Nesneye Yönelik Programlamanın Farklılıkları Nesne Kavramı, Sınıf Kavramı, Sınıf Oluşturulması
Hafta 5	Sınıflardan Nesne Yaratma, Metotlara Nesne Aktarımı, Properties (Özellikler) Yaratma, Auto-Implemented Properties
Hafta 6	this Referansı,Kurucu Metotlar,Nesne Başlatıcıları, Nesne Dizileri, Sort() ve BinarySearch() Metotları, Yıkıcı Metotlar

Haftalık Ders İçeriği

	Hafta İçerikleri
Hafta 7	Kalıtım Kavramı Protected Erişim Belirleyicisi Kullanımı,Temel Sınıfı Ezme (Overriding Base Class)
Hafta 8	Kalıtım Engelleme,Soyut Sınıflar (Abstract Classes)
Hafta 9	Vize Sınavı
Hafta 10	Kapsülleme (Encapsulation),Neden Kapsülleme Yapılır?,Çokbiçimlilik (Polymorphism)
Hafta 11	Sanal Metotlar (Virtual Methods),Fonksiyon Ezme (Override)
Hafta 12	Hata Yakalama (Exception Handling) Exception Nesnesi Try – Catch Yapısı Finally Deyimi Throw Deyim
Hafta 13	Projelerin Anlatılması
Hafta 14	Genel Tekrar

Değerlendirme

- Sınav ve Ödevler
  1. Vize
  2. Final
  3. Vize Sınavına Kadar Haftalık Ödevler
  4. Vize Sonrası Proje Ödevi (Proje Konusu Vizeye Kadar Belirlenmelidir.)

  Vize: %30
  Final: %40
  Ara Ödevler: %10
  Final Ödevi:%20

Dikkat Edilmesi Gereken Noktalar

- ✓ Derslerin başlama saatinden 10 dk sonrasında öğrenci derse kabul edilmeyecektir.
- ✓ Ders notları LABSİS platformundan paylaşılacaktır.(labsis.mu.edu.tr)
- ✓ Öğrenciler ödevleri sadece LABSİS'te açık olan Haftaya Gönderebilecektir. Mail den veya OBS den dosya kabul edilmeyecektir.
- ✓ Sınav sırasında kopya çekenlere ve verenlere o verilecektir.
- ✓ Uygulama ödevlerinde internetten alındığı tespit edilen veya arkadaşıyla ödevini paylaşan ve alan öğrencilere de o verilecektir.

1. Hafta

Neden OOP?

UML Nedir?

Neden OOP (Object Oriented Programming)

- ✓ Bir sonraki aşamaya, yani düzenli, temiz, dünyadaki standartlara uygun, tekrar kullanılabilir, kodu açtığımızda profesyonel birine göstermeye utanmayacağımız kodlar yazmak istiyoruz.
- ✓ Bizden sonra spagetti karman çorman kodumuz üzerinde çalışacak gariban yazılımcının edeceği küfürler yüzünden kulaklarımızın çınlamasını istemiyoruz. Kodumuz okunabilir oluyor. Karmaşa azalıyor.
- ✓ Minik minik, kendi projelerimize ait kütüphane oluşturup, yeni projelerde bunları modüler olarak kullanabilmek istiyoruz. Tıpkı Lego gibi. Kodumuz yeniden kullanılabilir oluyor.

Neden OOP (Object Oriented Programming)

- ✓ Zamanında spagetti kod yazarken, kendi kodumu okuyamadığım oluyordu, sonra kendime küfrediyordum, bunun da olmasını istemiyoruz.
- ✓ Nesne yönelimli programlamanın temel kavramlarını öğrendiğimiz zaman, diğer nesne yönelimli programlama dillerini kolayca kavrayabiliyoruz ve o dilleri kolayca öğrenebiliyoruz.
- ✓ İyi okunabilen kodlarda, karmaşa daha az olduğu için, hataları yakalamak daha kolay oluyor.
- ✓ Test edilebilen kodlar yazabiliyoruz. Bu sayede daha sonra ortaya çıkabilecek hataları önceden yakalayabiliyoruz.

Nesne-merkezli Olmak Ne Demektir?

- ✓ BT'nin en temel faaliyeti olan programlamaya tarih boyunca farklı yaklaşımlar geliştirilmiştir.
- ✓ Genel olarak 4 farklı isimle (yordamsal (emperative ya da procedural), fonksiyonel (functional), mantıksal (logical) ve nesne-merkezli (object-oriented)) ifade edilen bu programlama paradigmalarından tarihi olarak yordamsal olanı en önce ortaya çıkmış ve çok uzun süre genel amaçlı program geliştirmenin tek yöntemi olarak kalmıştır.
- ✓ Bir programlama yaklaşımı ya da paradigması olarak nesne-merkezli programlama, her şeyin nesne (object) üzerine kurulduğu bir programlama yaklaşımıdır. Bu yaklaşımın kurucusu sayılan Alan Kay'in sıraladığı, ileride detaylı olarak ele alacağımız, nesne-merkezli programlamanın prensiplerinden ilkinin "herşey bir nesnedir" (everything is an object) olması, bu durumu güzel bir şekilde ifade etmektedir.
- ✓ Nesne-merkezli dillerin en temel iki güçlü olduğu nokta, gerçekliği ifade etme ve değişimi yönetme becerisidir.

Nesne-merkezli Olmak Ne Demektir?

- ✓ Object-oriented terimini ilk defa kullanan ve Smalltalk dilinin mucidi Alan Kay, nesne-merkezli dillerin en temel 6 özelliğini şöyle ifade etmiştir:
- ✓ Herşey bir nesnedir (Everything is an object.):
- ✓ Nesneler birbirleriyle mesajlaşarak iletişimde bulunurlar (Objects communicate by sending and receiving messages):
- ✓ Her nesnenin bir belleği vardır (Objects have their own memory)
- ✓ Her nesnenin bir sınıfı vardır (Every object is an instance of a class):
- ✓ Sınıf, nesnelerinin ortak davranışlarını ifade eder (The class holds the shared behavior for its instances):

OOP Nedir?

- ✓ OP programlama tekniği, tamamen insanı taklit eder. Bu açıdan, yüzyılın en dinamik ve verimli programlama tekniği olarak düşünülebilir. Peki, bu taklit nasıl oluyor?

- ✓ Günümüz dünyasında, ihtiyaç duyduğumuz şeylerin hemen hepsini, üreticilerinden elde ediyoruz... Kazmayı küreği alıp evimizi yapmıyoruz ya da arka odadaki elektronik aletlerle kendimize bir LCD televizyon üretmiyoruz. Sorumluluğu vermiş olduğumuz üreticiden gidip satın alıyoruz. İşte teknolojinin bu kadar hızlı gelişmesinin temelinde de bu "branşlaşmanın" yatmakta olduğu çok net bir biçimde gözümüze çarpıyor.

- ✓ Bir mucit, bir ürün icat ediyor. Daha sonra bu ürünün seri üretimine başlıyor. Zaman geçtikçe, bu ürüne yeni özellikler ekliyor ve ayrıca, ürünün ilk hallerinde de bulunan işlevleri daha verimli hale getiriyor. Bu ürünün ilk halinde bulunan özelliklere "yeniden kullanılabilir" demek pek de yanlış olmaz. Ayrıca, ürünün yıllar boyunca geçirdiği değişim de, "geliştirilebilirlik" ilkesine özgüdür.

OOP Nedir?

- ✓ Binlerce satır koddan oluşan yazılım ürünleri... Bu ürünleri geliştiren programcılar; her kodu, her seferinde tek tek yazmak zorunda kaldılar. Muhtemelen bu sebepten dolayı, ürünün versiyonları arasında oldukça uzun bir zaman vardı. Çünkü tasarımdaki en ufak bir değişim (mesela yeni bir menü eklenmesi bile), kodun büyük ölçüde değişmesine neden oluyordu. Bu tam olarak zaman kaybıydı.

- ✓ Artan yazılım talebi, yeni bir kodlama tekniğini yani OOP tekniğini doğurdu. Tıpkı insanların branşlaştığı gibi, kodları da kendi aralarında görevlerine göre ayırarak bir nevi kod fabrikaları oluşturuldu.

OOP Nedir?

- ✓ İşte bu fabrikaların adı class (sınıf) olarak biliniyor. Haliyle biz, bu class'lardan nesne üretiyor ve o nesneyi, ihtiyacımız olan her yerde kullanabiliyoruz. OOP kodlama tekniği, tam olarak insanın şu anki yaşamını taklit eden bir yapı.

- ✓ Peki bir class'ı nasıl tasarlarız? Neler içereceğine nasıl karar veririz? Bu tamamen üreteceğimiz nesneyi ne amaçla kullanacağımıza bağlıdır. Bir fabrikayı nasıl kurarız? Fabrikayı kurma aşamasına geldiyseniz eğer, ne üreteceğinize çoktan karar vermişsiniz demektir değil mi? Öyleyse konu class olduğunda da, öncelikle istediğimiz nesnede hangi özelliklerin olması gerektiğini bilmek durumundayız.

OOP Nedir?

- ✓ Bir ayakkabı satıcısının kullanacağı bir program olsun. Kesinlikle bu projede bir ayakkabı nesnesine ihtiyacınız olacak öyle değil mi? Peki bu ayakkabı nesnesinin özellikleri neler olmalı?

Markası

Tipi (Bot, spor, çizme vs)

Malzemesi (Deri, süet vs)

Numarası

Rengi

Bağcıklı mı değil mi?

class' ımız (yani fabrikamız) üretilecek olan nesnemizin bu özelliklerini belirlemek durumunda.

UML Nedir?

- ✓ UML (Unified Modelling Language—Birleşik Modelleme Dili), iş sistemlerinin modellenmesi konusunda ortaya çıkmış bir dildir. Genellikle yazılım sektöründe kullanılmakla beraber, iş sistemlerini, bir süreci veya herhangi bir işi grafikler ile açıklamak isteyenlerce kullanılır.

- ✓ UML 1997 yılından bu yana geliştirilmekte olup 2015 yılında UML 2.5 sürümüyle birçok yönden geliştirilmiştir. Son sürümlerle beraber dildeki birçok eksik ve hatalar giderilmiştir.

UML Nedir?

- ✓ UML bir programlama dili değildir, aksine bir modelleme dilidir. Bir projenin her türlü bileşeninin ve bu bileşenlerin aralarındaki ilişkilerin görsel olarak ifade edilebilmesini sağlayan bir standarttır.
- ✓ Büyük çaplı projelerde karmaşık yapıları hem geliştirme takımına kolayca aktarmak hem de projenin mimarisini daha proje geliştirilmeye başlanmadan önce belirlemek için oluşturulmuştur.

UML Nedir?

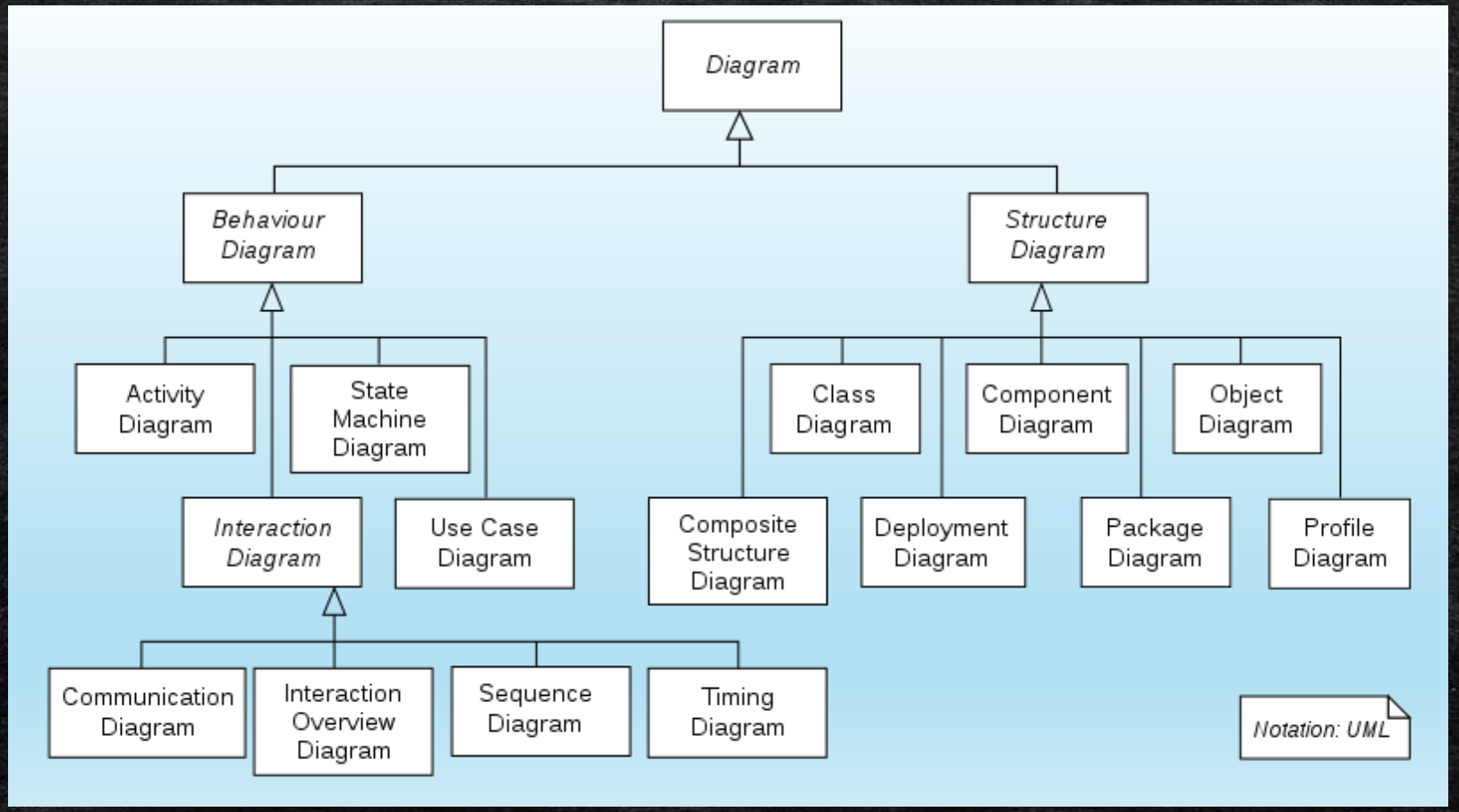


Diagram
Behaviour Diagram
Structure Diagram
Activity Diagram
State Machine Diagram
Interaction Diagram
Use Case Diagram
Class Diagram
Component Diagram
Object Diagram
Composite Structure Diagram
Deployment Diagram
Package Diagram
Profile Diagram
Communication Diagram
Interaction Overview Diagram
Sequence Diagram
Timing Diagram
Notation: UML

UML Çeşitleri

- ✓ Behaviour Diagram (Davranış Diyagramlar)
  - ✓ **Activity Diagram (Faaliyet Diyagramı):**
  - ✓ **State Machine Diagram (Durum Diyagramı):**
  - ✓ **Use Case Diagram (Kullanım Senaryosu Diyagramı):**
- ✓ Structure Diagram (Yapısal Diyagramlar)
  - ✓ **Class Diagram (Sınıf Diyagramı):**
  - ✓ **Component Diagram (Bileşen Diyagramı):**
  - ✓ **Diagram (Nesne Diyagramı):**
  - ✓ **Composite Structure Diagram (Birleşik Yapı Diyagramı):**
  - ✓ **Deployment Diagram (Dağılım Diyagramı):**

UML Faydaları Nedir?

- ✓ Takım çalışmasında yardımcı olur, UML standartlaşmış uluslararası bir dildir ve bu dili bilen herkes diyagramlardan aynı şeyleri anlar. Müşteri ve teknik sorumlular diyagramlar üzerinden rahatça iletişim kurabilirler.

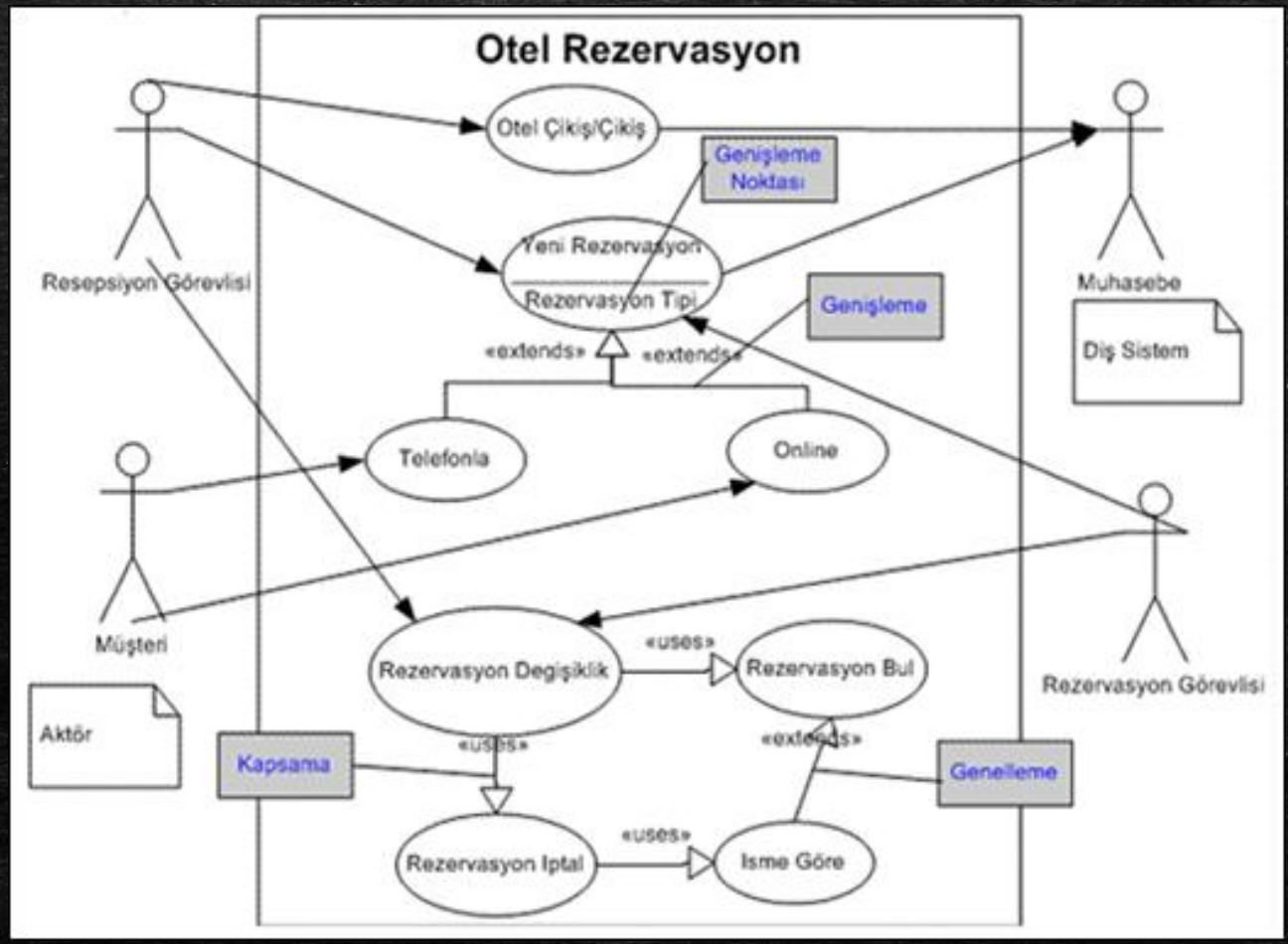
- ✓ Ekibinizde yer alan çalışma arkadaşlarınızla uyumlu bir şekilde çalışabilirsiniz ve ekibe yeni giren bir çalışan da projeye rahatlıkla dahil edilebilir.

- ✓ Kodlamayı kolaylaştırır, UML ile uygulamanızın tasarımı analiz aşamasında yapıldığı için, modellemeniz bittikten hemen sonra kod yazmaya başlayabilirsiniz.

UML Faydaları Nedir?

- ✓ Hataları en aza indirir, UML ile bütün sistem tasarlandığı için sistemde hata çıkma olasılığı azdır. Çıkan hataları düzeltmek ise çok daha kolaydır.
- ✓ Tekrar kullanılabilir bileşenleriniz artar, UML ile tüm sistem ve sistemin bileşenleri daha baştan belirlendiği için, o bölümler tekrar tekrar yazılmayacaktır.
- ✓ Program kararlılığı artar, UML ile ayrıntılı gereksinim analizleri yapıldıktan sonra senaryolar belirlenir. Senaryoların baştan belirli olması programınızı daha kararlı hale getirmenizde size yardımcı olur.

UML Diyagramları (Use Case Diyagram)



UML Diyagramları (Class Diyagram)