Nesne Tabanlı Programlama
Bölüm 10
UML Class Diyagramları

Dr. Öğr. Üyesi Murat TAŞYÜREK (kayubmprogramlama1@gmail.com)

11 Aralık 2023

Kayseri Üniversitesi, Bilgisayar Mühendisliği Bölümü

Ders İçeriği

- UML
- Namespace, Enum kullanımı
- Visual Studio eklenti yükleme
- Örnekler

UML

- **Unified Modeling Language (UML)** bir sistemin tasarımını görselleştirmek için yazılım mühendisliği alanında genel amaçlı modelleme dilidir.
- Yazılı bir dil değildir.
- **UML**, sistem ve yazılım geliştiricilerin yazılım sistemlerinin eserlerini belirleme, görselleştirme, oluşturma ve belgelemenin yanı sıra iş modellemesi için geliştirilen bir yöntemdir.
- 1995 yılında, yazılımlarda bir standart yaklaşım oluşturmak için geliştirilmiştir.
- **UML**, büyük ve karmaşık sistemlerin modellenmesinde başarısı kanıtlanmış en iyi mühendislik uygulamalarının bir koleksiyonunu temsil eder.

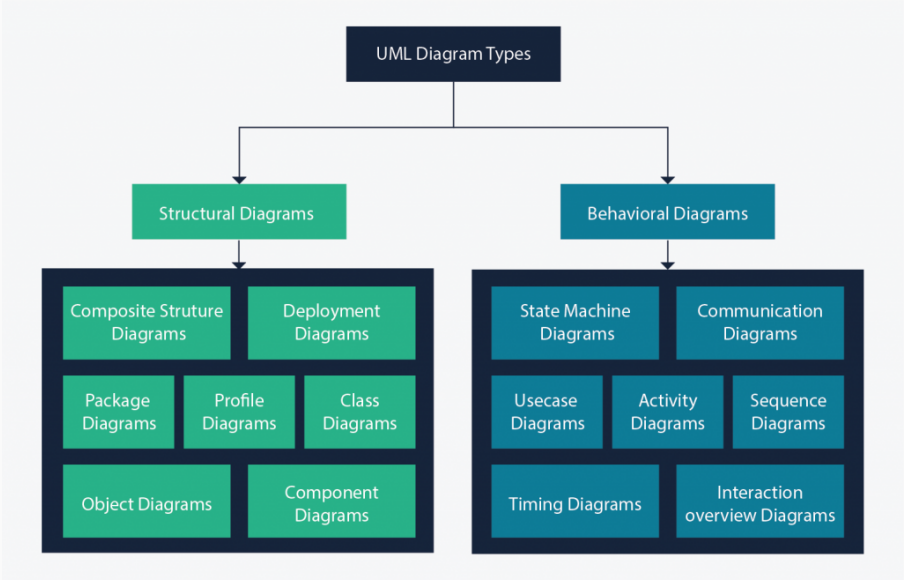
UML

- **UML**, nesne yönelimli yazılım geliştirmenin ve yazılım geliştirme sürecinin çok önemli bir parçasıdır.
- **UML**, yazılım projelerinin tasarımını ifade etmek için çoğunlukla grafik gösterimler kullanır.
- **UML**'yi kullanmak, proje ekiplerinin iletişim kurmasına, potansiyel tasarımları keşfetmesine ve yazılımın mimari tasarımını doğrulamasına yardımcı olur.
- **UML** diyagramları ile önceden modellediğiniz bir nesne tabanlı modelinizi diğer insanlarında anlayabileceği (başka programla dili kullanan kullanıcılar veya programlama bilmeyen kişiler) formata çeviriyorsunuz
- Bu sayede yazılım geliştiren kişiler arasında ortak bir dil oluşturuyor.

UML Çeşitleri

- Yaygın olarak bilinen 14 çeşit **UML** türü bulunmaktadır.
- Yapısal ve davranışsal olmak üzere iki gruba ayrılmaktadır.
- **Structure Diagrams(Yapısal Diyagramlar)**, modellenmekte olan sistemde bulunması gerekenler için kullanılır.
- Bu ders kapsanında inceleyeceğimiz Class Diagram, Structure Diagram altında bulunmaktadır.
- Tasarlanan OOP modelin herkes tarafından anlaşılabilecek yapısını göstermek için kullanılır.

UML Types


UML Diagram Types
Structural Diagrams
Behavioral Diagrams
Composite Struture Diagrams
Deployment Diagrams
Package Diagrams
Profile Diagrams
Class Diagrams
Object Diagrams
Component Diagrams
State Machine Diagrams
Communication Diagrams
Usecase Diagrams
Activity Diagrams
Sequence Diagrams
Timing Diagrams
Interaction overview Diagrams

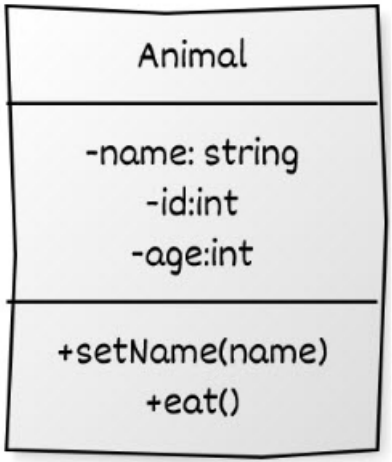
Örnek

- Animal türünde bir sınıfımızın olduğunu düşünelim.
- Bu sınıfımızın private olarak metin türünde name, integer türünde id ve age özellikleri olsun.
- Name özelliğini set etmek için setName adına public olarak erişilebilen bir metodu olsun.
- Eat ismininde dışardan erişebilen bir metodu olsun.
- İlgili sınıfın c# kodunu ve UML Class diagram'ını inceleyelim.

Source code

```
0 references
class Animal
{
    private String name;
    private int id;
    private int age;
    0 references
    public void setName(String name)
    {
        this.name = name;
    }
    0 references
    public void eat()
    {
        Console.WriteLine("Eating");
    }
}
```

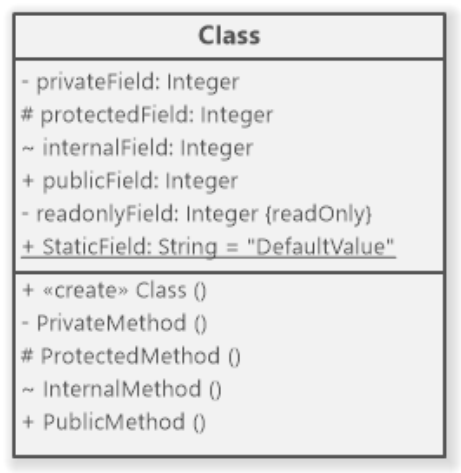
UML Class Diagram



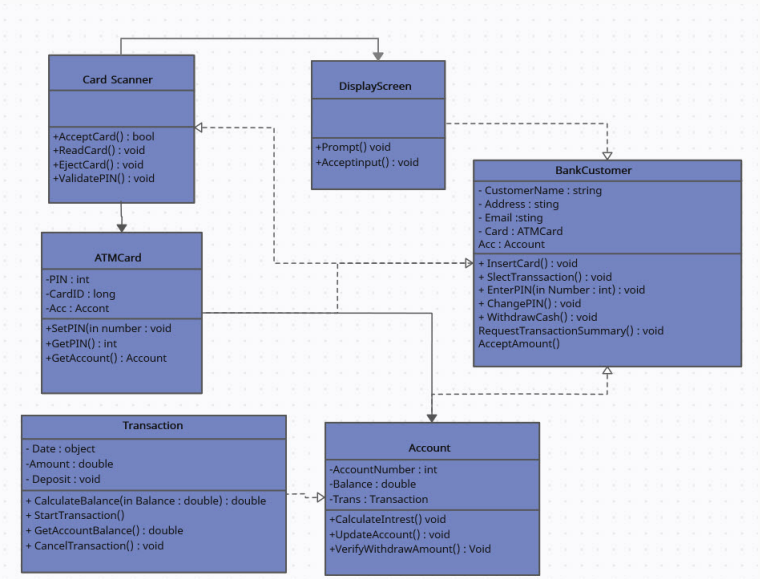
UML Class Diagram

- Class diyagramlarında sınıflar yukarıda görüldüğü gibi ifade edilir.
- En üst kısımda **Class** (sınıf) adı yazar.
- İkinci kısım **Attributes** (özellik) kısmıdır.
- Alt kısım ise **Methods** kısmını göstermektedir.
- - ifadesi **private** , + ifadesi **public** olarak erişildiğini ifade edilmektedir.
- Internal (sadece oluşturulan proje içerisinden erişilebilen modifier türü) $\sim$ ile ifade edilir. Protected # ile ifade edilir.
- **readOnly** {readOnly} olarak, **statik** ise altı çizi (under line) olarak ifade edilir.

Örnek

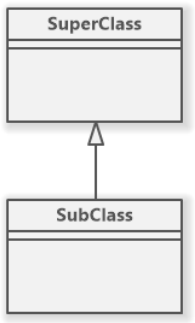

Class
- privateField: Integer
protectedField: Integer
~ internalField: Integer
+ publicField: Integer
- readonlyField: Integer {readOnly}
+ StaticField: String = "DefaultValue"
+ «create» Class ()
- PrivateMethod ()
ProtectedMethod ()
~ InternalMethod ()
+ PublicMethod ()

Gerçek hayatta karşınıza çıkacak UML Class Diagram



C# Inheritance in UML

- Sınıfların **kalıtımı (inheritance)** doğrudan UML kullanılarak modellenebilir.
- UML genellemesini kullanarak bir alt sınıfı bir üst sınıfa bağlayabilirsiniz. Genellemenin üçgen baş oku, bir alt sınıftan bir üst sınıfa işaret eder.

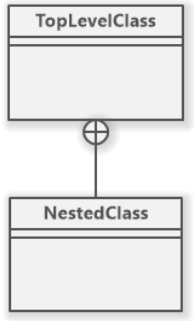


C# Nested Classes in UML

- Bağlı sınıflar (nested classes), UML içerme ilişkisi kullanılarak tanımlanabilir.
- Bağlayıcı, içinde artı işareti olan bir daire olarak gösterilir. Dairenin ucu kapsayıcı sınıfına, diğer ucu iç sınıfa bağlıdır.

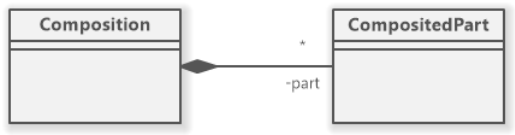
In C#, we can define a class within another class. It is known as a nested class. For example,

```
class OuterClass {
  ...
  class InnerClass {
    ...
  }
}
```



C# Class Composition in UML

- UML kompozisyon (Composition) ilişkilendirmesini kullanarak bir sınıfı başka bir sınıfın alan türü olarak kullanılabilir.
- Bu şekilde, sınıflar karmaşık olarak oluşturabilir.

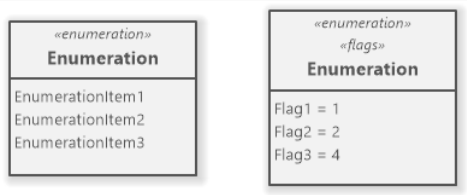


C# Interfaces in UML

- Arayüzler (Interfaces), bir sınıflandırıcıdan hangi üyelerin beklendiğini tanımlar.
- C# Interfaces, UML arayüzleri kullanılarak modellenebilir. UML arabirimleri, notkalı çizgi ilişkisi kullanılarak sınıflarla ilişkilendirilir.

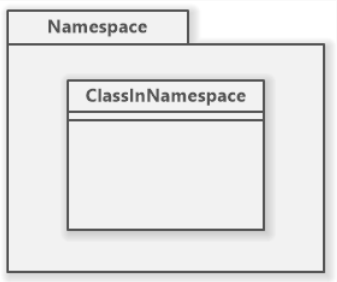
C# Enum in UML

- Numaralandırılan değerler (enumeration), bir adlandırılmış değerler kümesini tanımlar. Belirli numaralandırma öğeleri için sayısal değerler tanımlanabilir. C# numaralandırmaları, UML numaralandırmaları kullanılarak modellenmiştir. Enumaration için flag kullanılır.



C# Namespace in UML

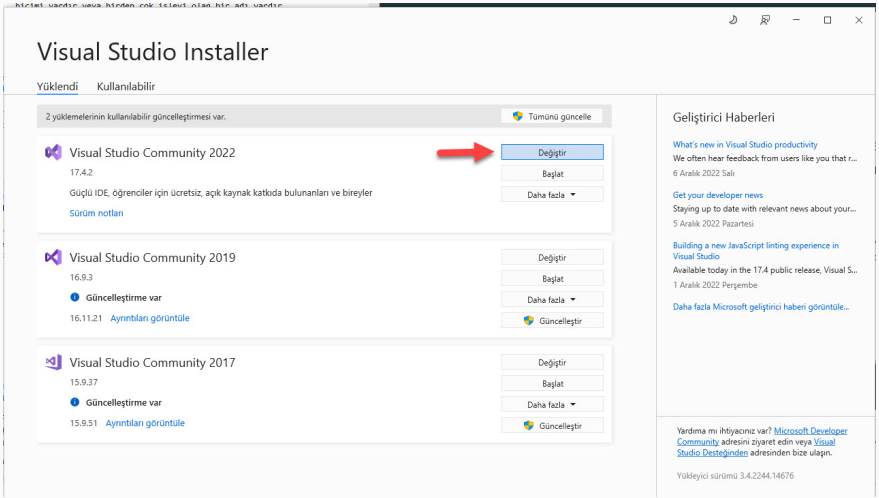
- **Namespace** ortak sınıfları bir arada tutmak için kullanılan yapıdır. Kütüphane adı olarak geçer.
- **Using** ibaresi ile kullanılır ve UML içerisinde aşağıda gösterildiği gibi ifade edilir.



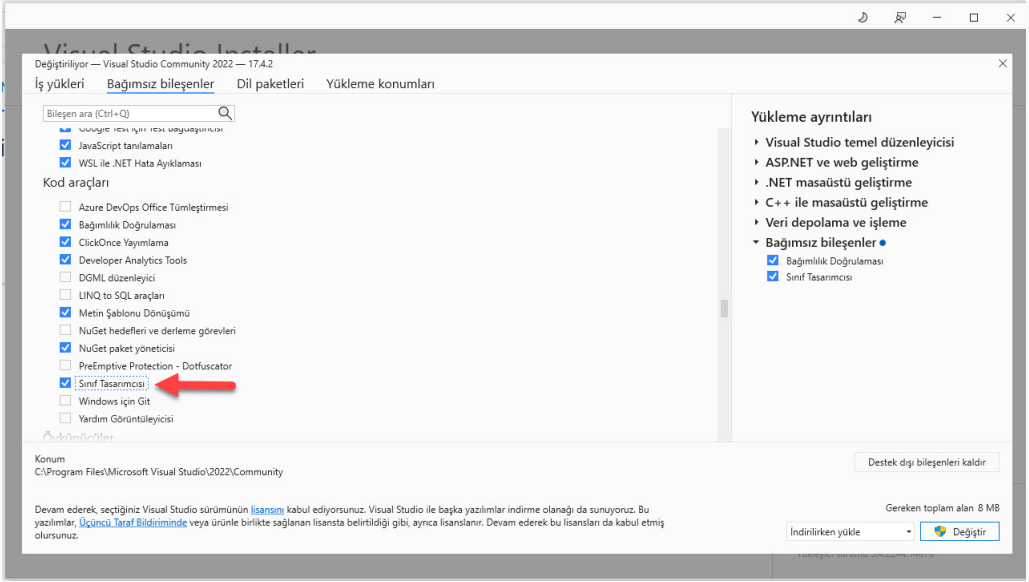
Visual Studio eklenti yükleme

- UML oluşturmak için birçok hazır program bulunmaktadır.
- Visual studio için geliştirilen eklenti ile otomatik olarak UML diyagramınızı oluşturabilirsiniz.
- Windows başlangıç menüsünden **Visual Studio Installer** açın.
- Visual studo sürümününden **değiştir (change)** işaretleyin.
- **Individual components (bağımsız bileşenler)** tabında bulunan **Code tools (Kod araçları)** kategorisi alındaki **Class Designer (Sınıf Tasarımcısı)** işaretleyin.

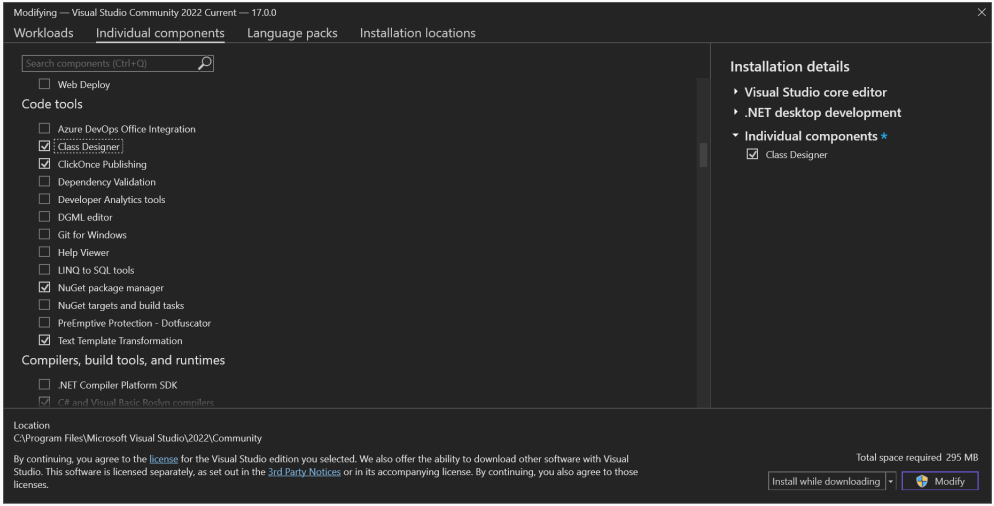
Visual Studio Eklenti



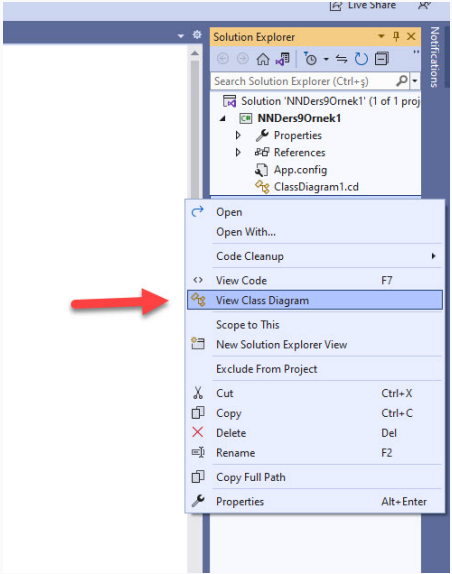
Visual Studio Eklenti



Visual Studio Eklenti (ingilizce)



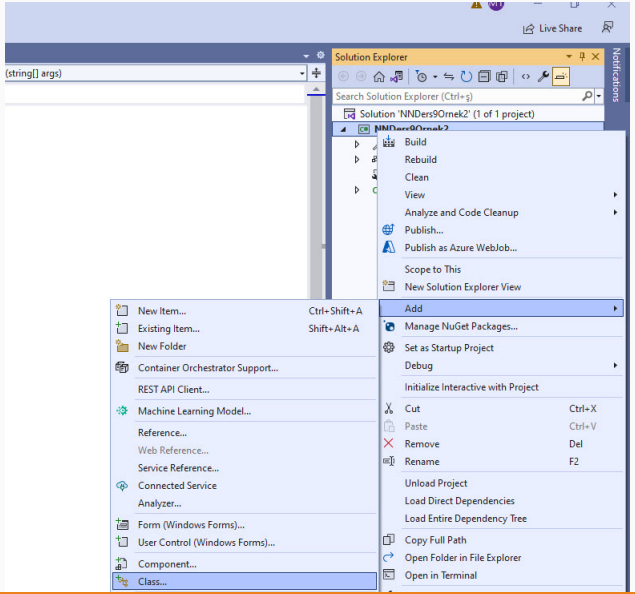
Kurulum işleminden sonra



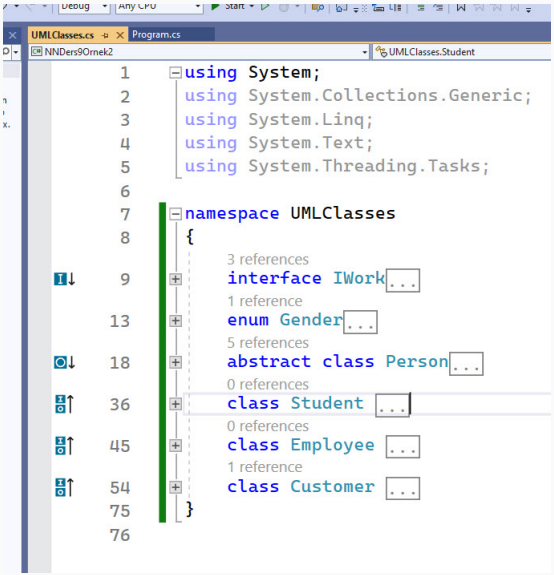
Örnek

- Sınıfları yeni bir namespace içerisinde oluturan,
- Cinsiyet için Enum kullanan
- Person sınıfı için abstract class kullanan
- Employee, student ve Customer sınıflarınının birer person olduğunu ve doWork yeteneklerini olduğunu
- Sistemin kodlayalım ve UML diyagramını oluşturulalım.

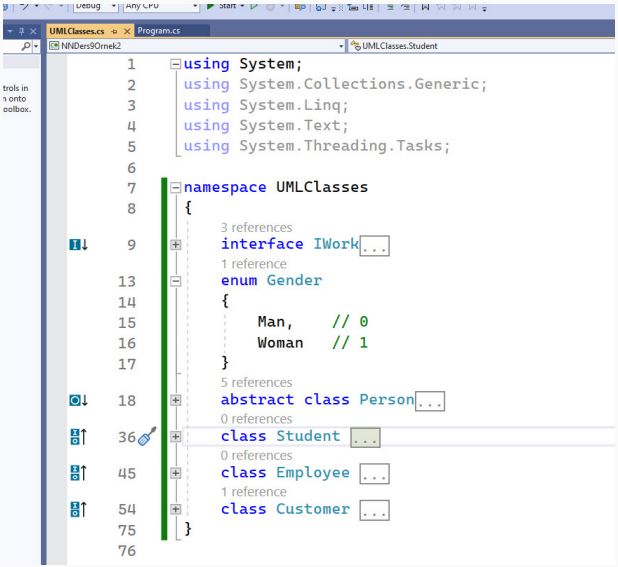
Harici Namespace ve doğal olarka sınıflar ekleme



Namespace


Debug
Any CPU
Start
UMLClasses.cs
Program.cs
NNDers9Ornek2
UMLClasses.Student
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace UMLClasses
{
3 references
interface IWork...
1 reference
enum Gender...
5 references
abstract class Person...
0 references
class Student ...
0 references
class Employee ...
1 reference
class Customer ...
}

Enum


UMLClasses.cs
Program.cs
NNDers9Ornek2
UMLClasses.Student
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace UMLClasses
{
3 references
interface IWork...
1 reference
enum Gender
{
Man, // 0
Woman // 1
}
5 references
abstract class Person...
0 references
class Student ...
0 references
class Employee ...
1 reference
class Customer ...
}

Person Class

```
abstract class Person
{
    0 references
    public int age { get; set; }
    0 references
    public string name { get; set; }
    0 references
    protected string adress { get; set; }
    private int _tcKimlikNO;
    public Gender _gender;
    1 reference
    public Person()
    {
        Random rnd = new Random();
        this._tcKimlikNO = rnd.Next(0, 12345678);

    }
    0 references
    public int getTcKimlikNO
    {
        get { return _tcKimlikNO; }
    }
}
```

Student ve Empoloyee Clesses

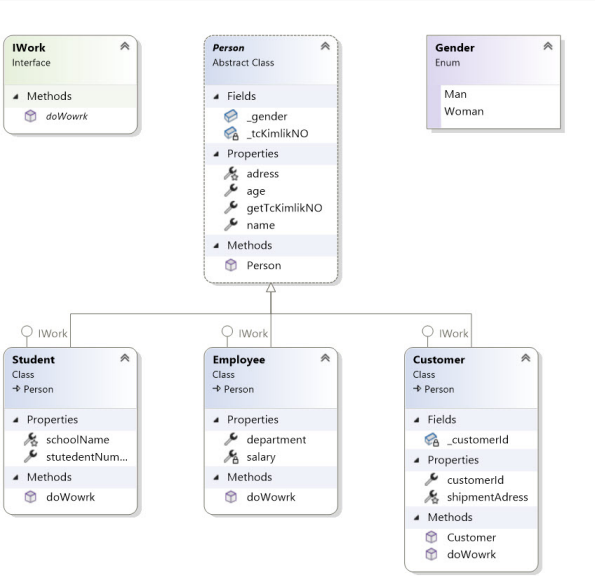
```
0 references
class Student : Person, IWork
{
    0 references
    public int stutedentNumber { get; set; }
    0 references
    protected string schoolName { get; set; }
    1 reference
    public void doWowrk()
    {
        Console.WriteLine("I am stutedent.");
    }
}
```

```
0 references
class Employee : Person, IWork
{
    0 references
    public string department { get; set; }
    0 references
    int salary { get; set; }
    1 reference
    public void doWowrk()
    {
        Console.WriteLine("I am worker.");
    }
}
```

Customer Class

```
class Customer : Person, IWork
{
    private int _customerId;
    1 reference
    public int customerId
    {
        get
        {
            return customerId;
        }
    }
    0 references
    protected string shipmentAdress { get; set; }
    0 references
    public Customer() : base()
    {
        Random rn = new Random();
        this._customerId = rn.Next(0, 1000);
    }
    1 reference
    public void doWowrk()
    {
        Console.WriteLine("I am customer.");
    }
}
```

UML Class Diagram



Ödev

- Asp.Net Web Application veya Windows Form ile football oyunu kodlayınız.
- Yetenekleri göstermek (şut atma vs top tutma) için Interface yapısı kullanılmalıdır.
- Futbolcuları oluştururken (kaleci, ön libero, forvet vb.) inheritance class kullanılmalıdır.
- Ödevleriniz özgün olsun.
- Kullandığınız objeler (btn vb.) classlardan oluşturduğunuz nesneler ile bağlantılı olsun.
- Kodlarınız ayrı bir Namespace olarak oluşturun.
- Oluşturduğunuz modelin UML Class Diagramını gösteriniz.
- 18 Aralık 07.59'a kadar gönderilenler 10+2, finale kadar gönderenler 10 puan (kayubmprogramlama1@gmail.com).