Bölüm 3
C# Metod, Sınıf

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

6 Mart 2024

Kayseri Üniversitesi, Bilgisayar Mühendisliği Bölümü

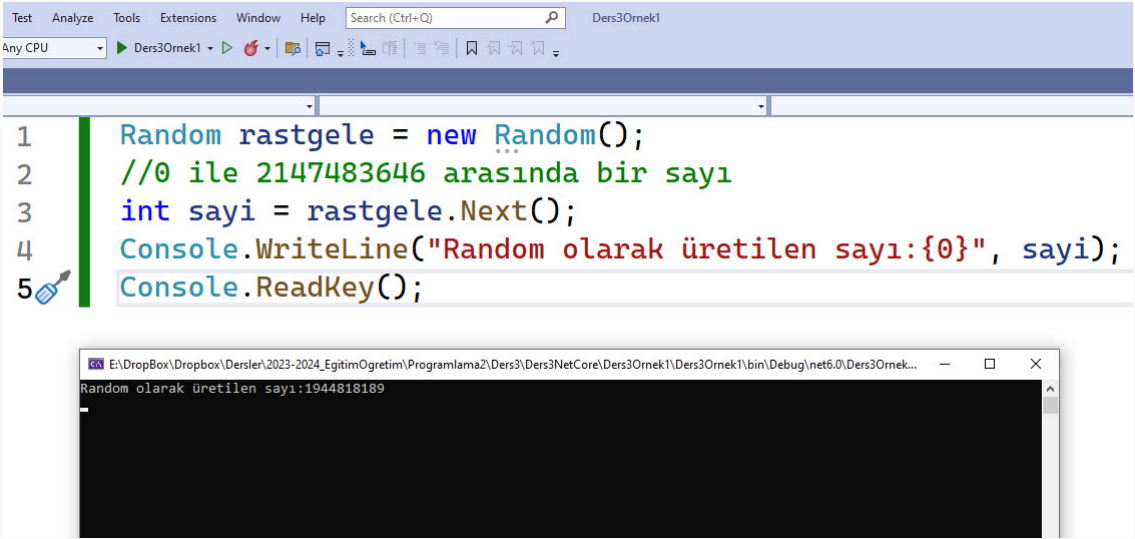
Ders İçeriği

- Random değer üretme
- Static/ nonstatic Metot yazma
- Recursive/iterative faktoriyel metotları
- Sınıf yazma/nesne tanımlama
- Nesne üretme ve metotları çağırma
- Nesne dizileri oluşturma
- Ödev

Random Sınıfı

- "Random", İngilizce'de "rastgele" anlamına gelir.
- Bilgisayar biliminde , "random" veya "rastgele" terimi, öngörülemeyen, belirli bir düzen veya model olmadan ortaya çıkan veya seçilen bir olayı veya değeri ifade eder.
- Rastgelelik, belirli bir desen, kurallar veya tahminler olmadan gerçekleşen bir durumu tanımlar
- C#'da Random sınıfı, rastgele sayılar oluşturmak için kullanılan bir sınıftır.
- Bu sınıf, çeşitli rastgele sayı üretme yöntemlerini ve işlevlerini sağlar.
- Random sınıfı kullanılarak, rastgele sayılar üretmek için farklı metotlar kullanılabilir.

Random değer üretme



Random Örnek

- 1 ile 50 arasında 5 adet random sayı üretilmek istenmektedir.
- For döngüsü kullanarak bu random sayıları üretininiz.
- Üretilen değerleri ekrana yazdırınız.
- İlgili örneği .Net Core ile kodlayınız.

Random değer üretme örnek 2

```
1  for (int i = 0; i < 5; i++)
2  {
3      Random rnd = new Random();
4      int sayi = rnd.Next(1, 50);
5      Console.Write("{0}. üretilen sayı:", i + 1);
6      //.ToStrning() yaygın kullanılan bir yöntemdir.
7      Console.WriteLine(sayi.ToString());
8  }
9  Console.ReadKey();
10
```

```
E:\DropBox\Dropbox\Dersler\2023-2024_EgitimOgretim\Programlama2\Ders3\Ders3NetCore\Ders3Ornek2\Ders3Ornek2\bin\Debug\net6.0\Ders3Ornek...
1. üretilen sayı:46
2. üretilen sayı:5
3. üretilen sayı:2
4. üretilen sayı:1
5. üretilen sayı:33
```

Fonksiyon

- "Fonksiyon", belli bir işlemi gerçekleştiren ve bir sonuç döndüren kod bloklarıdır.
- Bu bloklar, programın belirli bir işlevini gerçekleştirmek üzere tasarlanır ve çağrıldıklarında çalışırlar.
- Fonksiyonlar, kodun tekrar kullanılabilirliğini artırır,
- Karmaşık işlemleri modüler hale getirir,
- Kodun daha okunabilir ve sürdürülebilir olmasını sağlar.

Fonksiyon

- Fonksiyon aşağıda gösterildiği gibi tanımlanır.
- Fonksiyonlar **statik** olarak tanımlanabilir.
- Fonksiyonun "**static**" olarak belirtilmesi, o fonksiyonun bir sınıfa ait olduğunu ve sınıfın örneği oluşturulmadan doğrudan sınıf adıyla erişilebileceğini belirtir.
- **C#** ve diğer birçok programlama dilinde "**static**" kelimesi bu amaçla kullanılır

```
<görünürlük> <dönüş tipi> <ad>(<parametreler>)
{
    <fonksiyon kodları>
}
```

Fonksiyon Örnek

- Kullanıcıdan iki adet tam sayı değeri isteyen ve bu değerlerin toplamını kullanıcıya dönderen **fonskiyonu** kodlayınız.
- Tanımlanmış olduğunuz **fonksiyonu** iki defa kod içerisinde çağırınız.
- **Fonskiyondan** dönen değeri ekrana yazdırınız.
- İlgili örneği **.Net Core** ile kodlayınız.

Örnek fonksiyon

```
arinToplami
1   int toplam1 = SayilarinToplami();
2   Console.WriteLine("İlk sayı toplama sonucu:{0}",toplam1);
3
4   int toplam2=SayilarinToplami();
5   //otomatik .ToString() metodu ile sayıyı stringe çevirip toplar.
6   Console.WriteLine("İkinci sayı toplama sonucu:" + toplam2);
7   Console.ReadKey();
8
    2 references
9   static int SayilarinToplami() //statik fonksiyon
10  {
11      int Toplam = 0;
12      Console.WriteLine("Toplam için 1. sayıyı giriniz:");
13      int sayi1= Convert.ToInt32(Console.ReadLine());
14      Console.WriteLine("Toplam için 2. sayıyı giriniz:");
15      int sayi2 = Convert.ToInt32(Console.ReadLine());
16      Toplam = sayi1 + sayi2;
17      //bu bir fonksiyondur
18      //fonksiyon geriye değer dönderir.
19      return Toplam;
20  }
```

Procedure

- C# dilinde "**procedure**" (**prosedür**), genellikle işlemciyi belirli bir görevi gerçekleştirmesi için yönlendiren ve geri dönüş değeri olmayan bir kod bloğunu ifade eder.
- C# dilinde bu terim daha az yaygın olarak kullanılır ve genellikle "**method**" (**metot**) terimi kullanılır.
- Bir "**metot**", belirli bir işlevi gerçekleştiren ve bir sınıfa bağlı olan kod bloğudur.
- **Procedure** geriye değer döndermez.
- **Procedure void** ile tanımlanır.

Örnek Procedure

- Kullanıcıdan bir tam sayı değeri alan ve bu tam sayı değerini faktoriyel hesabı yapan **procedure** göndereriniz.
- **Global bir değişken** tanımlayınız.
- Faktoriyel hesabı için yazmış olduğunuz **procedurde** global değişkene faktoriyel değerini atayınız.
-
- İlgili örneği **.Net Core** ile kodlayınız.

Örnek Procedure

```
int statikFaoktiyerlDegier = 1; //global değişken

Console.WriteLine("Faktoriyel hesabı için bir sayı giriniz.");
int sayi = Convert.ToInt32(Console.ReadLine());
FaktoriyelHesabi(sayi);
Console.WriteLine("Girilen sayı:{0}, sayının faktoriyeli:{1}",
    sayi, statikFaoktiyerlDegier);
Console.ReadKey();


1 reference
void FaktoriyelHesabi(int sayi)
{
    statikFaoktiyerlDegier = 1;
    for (int i = 1; i <= sayi; i++)
    {
        //statikFaoktiyerlDegier *= i;
        statikFaoktiyerlDegier = statikFaoktiyerlDegier * i;
    }
}
```

```
E:\DropBox\Dropbox\Dersler\2023-2024_EgitimOgretim\Programlama2\Ders3\Ders3NetCore\Ders3FaktoriyelProcedure\Ders3FaktoriyelProcedure\bin\...
Faktoriyel hesabı için bir sayı giriniz.
5
Girilen sayı:5, sayının faktoriyeli:120
```

Recursive/iterative Fonksiyon/Procedure

- **"rekürsif fonksiyon" (recursive function) ve "rekürsif prosedür" (recursive procedure)** terimleri, bir işlemi gerçekleştirmek için kendini çağıran bir fonksiyonu veya prosedürü ifade eder.
- **Rekürsif** fonksiyonlar ve prosedürler, bir problemi çözmek için kendilerini tekrar tekrar çağırarak çalışırlar.
- Genellikle, bir problemi daha küçük alt problemlere bölme ve her alt problemin çözümünü tekrar tekrar çağırarak elde edilen sonuçları birleştirme şeklinde çalışırlar.

Recursive/iterative Fibonacci örneği

- Fibonacci dizisi matematiksel olarak ağıdaki şekilde gösterilir.
- Buradaki F(n), n. terimi belirtir. Yani, Fibonacci dizisinin üçüncü ve sonsuz sayıdaki diğer terimleri F(1)=1=F(2) başlangıç değerleriyle ve F(n)=F(n-1)+F(n-2) eşitliği ile üretilir
- **Kullanıcıdan bir sayı isteyen ve bu sayıya kadar olan sayıların fibonacci değerlerini hesaplayan yazılımı kodlayınız.**

$$F(n) = \begin{cases} 1 & n = 1 \\ 1 & n = 2 \\ F(n-1) + F(n-2) & n \geqslant 3 \end{cases}$$

Recursive/iterative Fibonacci örneği

```
onacci
1   Console.Write("Bir sayı giriniz : ");
2   int sayi = Int32.Parse(Console.ReadLine());
3
4   for (int i = 0; i <= sayi; i++)
5   {
6       Console.WriteLine("{0}. sayısının fibonacci değeri:{1}",
7           i,FibonacciHesapla(i) + ", ");
8   }
9   Console.ReadKey();
10
11
        3 references
12  int FibonacciHesapla(int sayi)
13  {
14      if (sayi == 1)
15          return 0;
16      else if (sayi == 2)
17          return 1;
18
19      return FibonacciHesapla(sayi - 1) + FibonacciHesapla(sayi - 2);
20  }
```

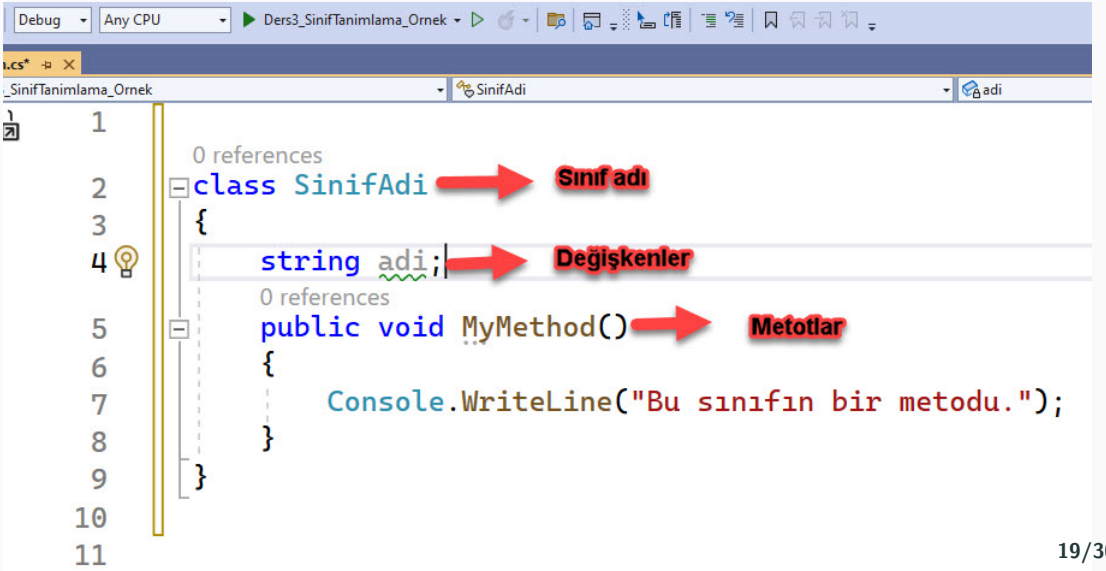
Sınıf

- C# programlama dilinde, "**sınıf**" (**class**) ve "**nesne**" (**object**) kavramları, nesne yönelimli programlamanın temel yapı taşlarını oluşturur.
- Bir sınıf, **veri alanları (fields)** ve bu alanlara erişimi kontrol eden **metotlar (methods)** ve **özellikler (properties)** içeren bir yapıdır.
- Bir sınıf, bir **nesnenin (object)** özelliklerini ve davranışlarını tanımlar.
- **Sınıflar**, genellikle benzer niteliklere sahip nesnelerin bir şablonunu oluşturmak için kullanılır.

Nesne

- Bir **nesne**, bir **sınıfın** örneğidir.
- Yani, bir sınıfın belirli bir durumunu temsil eden bir öğedir.
- Bir **nesne**, bir sınıfın tanımını kullanarak oluşturulur (yani, sınıfın bir örneği olarak).
- Her **nesne**, sınıf tarafından tanımlanan veri alanlarına ve metotlara sahip olur.
- Bir **nesne** oluşturmak için **"new"** anahtar kelimesi kullanılır

Sınıf tanımlama



Örnek Sınıf ve Nesne Tanımlama

- Öğrenci adı, soyadı ve numara bilgilerinin tutulduğu **sınıf**'ı tanımlayınız.
- Öğrenci bilgilerini güncellemek için **OgrenciBilgileriGuncelle** isminde bir metod yazınız.
- İlgili sınıftan bir nesne oluşturunuz ve **nesnenin** bilgilerini giriniz.
- İlgili örneği .NET Core ile kodlayınız.

Öğrenci Sınıf yazma/nesne tanımlama

```
:ore_Ogrenci                    Ogrenci                    soyadi
1   Ogrenci ogr = new Ogrenci();
2   ogr.OgrenciBilgileriGuncelle("Murat", "Taşyürek", 1234);
3   Console.WriteLine("Öğrendi Adı:{0}, Soyadı:{1}, Numarası:{2}",
4       ogr.adi, ogr.soyadi, ogr.numarasi);
5   Console.ReadKey();
6
        2 references
7   public class Ogrenci
8   {
9       public string adi;
10      public string soyadi;
11      public int numarasi;
12
            1 reference
13      public void OgrenciBilgileriGuncelle(string ad, string soyad, int numarasi)
14      {
15          adi = ad;
16          soyadi = soyad;
17          //this demek bu sınıftan üretilen nesne demek.
18          this.numarasi = numarasi;
19      }
20  }
21
```

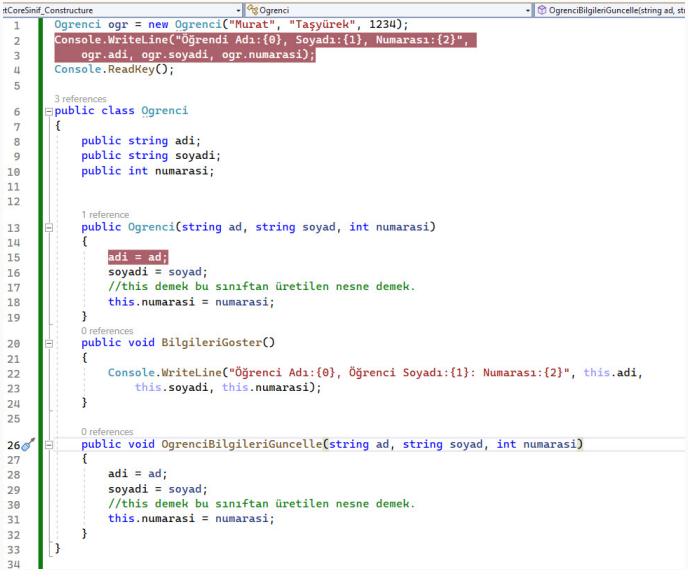
Nesne üretme ve metotları çağırma

```
0 references
static void Main(string[] args)
{
    Ogrenci ogr = new Ogrenci();
    ogr.OgrenciBilgileriGuncelle("Murat", "Taşyürek", 1234);
    ogr.BilgileriGoster();
    Console.ReadKey();
}
2 references
class Ogrenci
{
    public string adi;
    public string soyadi;
    public int numarasi;
    1 reference
    public void OgrenciBilgileriGuncelle(string ad, string soyad, int numarasi)
    {
        adi = ad;
        soyadi = soyad;
        //bu şekilde de kullanılır.
        this.numarasi = numarasi;
    }
    1 reference
    public void BilgileriGoster()
    {
        Console.WriteLine("Öğrenci Adı:{0}, Öğrenci Soyadı:{1}: Numarası:{2}", this.adi,
            this.soyadi, this.numarasi);
    }
}
```

Sınıf Kurucu Metod

- "**constructor**" (kurucu metot), bir sınıfın yeni bir örneği (nesnesi) oluşturulduğunda otomatik olarak çağrılan özel bir metottur.
- **Constructor'lar**, sınıfın başlangıç durumunu ayarlamak, alanları (fields) başlatmak veya diğer başlangıç işlemlerini gerçekleştirmek için kullanılır.
- **Constructor'lar** genellikle sınıfın adıyla aynı isme sahiptir ve geri dönüş türü belirtilmez.
- Bir sınıf birden fazla **constructor**'a sahip olabilir ve bu constructor'lar parametre alabilir veya almayabilirler.
- Aynı örneği **constructor** kullanarak kodlayalım ve canlı olarak tetiklendiğini gösterelim.

Nesne Constructor(canlı gösterim)



Dizi İçeren Sınıf

- Sınıflar da bir tür olduğu için bir sınıf başka bir sınıf içerisinde **tür (tip)** olarak tanımlanabilir.
- Sınıf tür olduğu için **sınıflar** dizi türü de olabilir.
- Öğrenci ve Öğrencinin derslerinin tutulduğu bir **sınıf** yapısı tanımlayalım.
- Bir öğrencinin birden fazla dersi olabilir.
- Dersin vize finali girilince final notu otomatik olarak sınıf içerisinde hesaplansın (Vizenin%30'u kalanı ise final).
- İlgili örneği .Net Core ile kodlayalım.

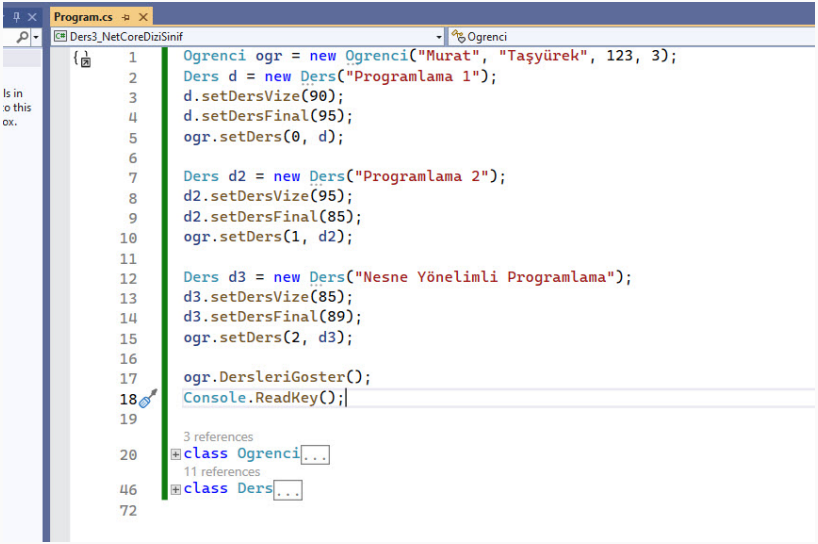
Ders Sınıfı-Nesne dizileri oluşturma

```
11 references
class Ders
{
    public string dersAdi;
    public int vize;
    public int final;

    3 references
    public Ders(string dersAdi)
    {
        this.dersAdi = dersAdi;
    }
    3 references
    public void setDersVize(int vize)
    {
        this.vize = vize;
    }
    3 references
    public void setDersFinal(int final)
    {
        this.final = final;
    }
    1 reference
    public int ortalama()
    {
        //cast
        int sonuc = (int)(0.3 * vize + 0.7 * final);
        return sonuc;
    }

}
```

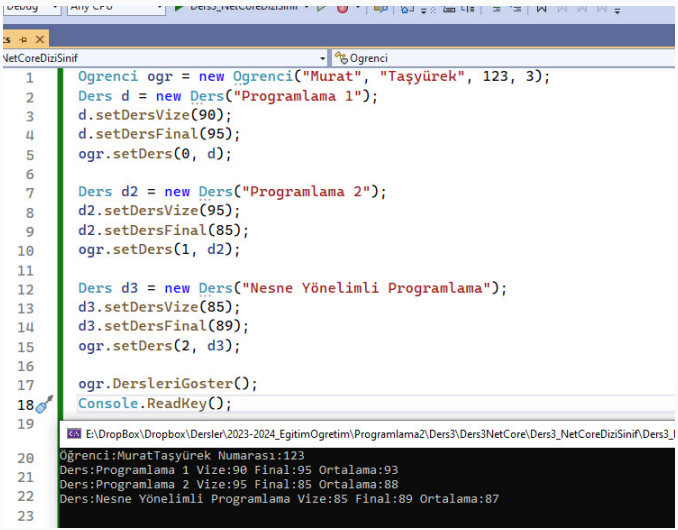
Öğrenci Sınıfı-Nesne dizileri oluşturma

```
3 references
class Ogrenci
{
    public string adi;
    public string soyadi;
    public int OgrenciNo;
    public Ders[] aldigiDersler;
    1 reference
    public Ogrenci(string adi, string soyadi, int ogrenciNO, int dersSayisi)
    {
        this.adi = adi;
        this.soyadi = soyadi;
        this.OgrenciNo = ogrenciNO;
        this.aldigiDersler = new Ders[dersSayisi];
    }
    3 references
    public void setDers(int dersID, Ders d)
    {
        this.aldigiDersler[dersID] = d;
    }
    1 reference
    public void DersleriGoster()
    {
        Console.WriteLine("Öğrenci:{0} Numarası:{1}", this.adi + this.soyadi, this.OgrenciNo);
        foreach (Ders s in aldigiDersler)
        {
            Console.WriteLine("Ders:{0} Vize:{1} Final:{2} Ortalama:{3}", s.dersAdi, s.vize, s.final, s.ortalama());
        }
    }
}
```

Kodun Tamamı



Sonuç


NetCoreDiziSinif
Ogrenci
1 Ogrenci ogr = new Ogrenci("Murat", "Taşyürek", 123, 3);
2 Ders d = new Ders("Programlama 1");
3 d.setDersVize(90);
4 d.setDersFinal(95);
5 ogr.setDers(0, d);
6
7 Ders d2 = new Ders("Programlama 2");
8 d2.setDersVize(95);
9 d2.setDersFinal(85);
10 ogr.setDers(1, d2);
11
12 Ders d3 = new Ders("Nesne Yönelimli Programlama");
13 d3.setDersVize(85);
14 d3.setDersFinal(89);
15 ogr.setDers(2, d3);
16
17 ogr.DersleriGoster();
18 Console.ReadKey();
19
E:\DropBox\Dropbox\Dersler\2023-2024_EgitimOgretim\Programlama2\Ders3\Ders3NetCore\Ders3_NetCoreDiziSinif\Ders3_
20 Öğrenci:MuratTaşyürek Numarası:123
21 Ders:Programlama 1 Vize:90 Final:95 Ortalama:93
22 Ders:Programlama 2 Vize:95 Final:85 Ortalama:88
23 Ders:Nesne Yönelimli Programlama Vize:85 Final:89 Ortalama:87

Ödev

- Bir zar atma oyunu tasarlayın. Bu oyun, iki oyuncu arasında oynanacak ve her bir oyuncu sırasıyla bir zar atacak. Zarların değerleri toplanacak ve en yüksek toplam puanı olan oyuncu kazanacaktır.
- Zar adında bir sınıf tanımlayın. Bu sınıf, bir zarın özelliklerini ve zar atma işlemini gerçekleştirecek metotları içermelidir. Zarın altı yüzü olacak ve her bir yüzü 1 ile 25 arasında bir değere sahip olmalıdır. Ayrıca, zar atma işlemini gerçekleştirecek ZarAt adında bir metot içermelidir.
- Oyuncu adında bir sınıf tanımlayın. Bu sınıf, bir oyuncunun adını ve toplam puanını saklayacak özelliklere sahip olmalıdır. Ayrıca, oyuncunun bir zar atmasını sağlayacak ZarAt adında bir metot içermelidir.
- Her oyuncu için bir zar atma işlemi gerçekleştirin ve zarların değerlerini toplayarak oyuncuların puanlarını güncelleyin. Oyun sonunda, her bir oyuncunun toplam puanını ekrana yazdırarak kazananı belirleyin. Oyuncuların adlarını ve toplam puanlarını kullanıcıdan giriş olarak alabilirsiniz. Her zar atışı tutulmalı ve en sonunda toplu olarak gösterilmeli.
- Özgün bir ödev olmalı, bütün çıktılar ve kaynak kodlar resim olarak gönderilmeli ve ayrıca projenin .zip formanında sıkıştırılmış hali de gönderilmeli
- 13 Mart 07.59'a kadar gönderilenler 8+2 puan (kayubmprogramlama1@gmail.com). Bu tarihden vize tarihine kadar gönderelenler ise 8 puan.