C# Hata Yönetimi

Hata Yönetimi

```
7       static void Main(string[] args)
8       {
9               Console.Write("a: ");
10              int a = int.Parse(Console.ReadLine());
11
12              Console.Write("b: ");
13              int b = int.Parse(Console.ReadLine());
14
15              var sonuc = a / b;
16
17              Console.WriteLine("{0} / {1} = {2}",a,b,sonuc);
18      }
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  POLYGLOT NOTEBOOK

bilal@DESKTOP-JLFMFHL MINGW64 ~/OneDrive/Masaüstü/Udemy/Web Udemy/.Net Core C#/C#-7-Hata Yönetimi
$ dotnet run
a: 1
b: a
Unhandled exception. System.FormatException: Input string was not in a correct format.
   at System.Number.ThrowOverflowOrFormatException(ParsingStatus status, TypeCode type)
   at System.Number.ParseInt32(ReadOnlySpan`1 value, NumberStyles styles, NumberFormatInfo info)
```

- Örnek b'ye int değer girmek gerekirken string bir değer girince exception altında Formatexception hatası verdi.
- Bu tür örnekleri alıp, kullanıcıların anlayacağı dile getirmek gerekir.

```
15              // Exception
16
17              // Unhandled exception. System.FormatException
18              // Unhandled exception. System.DivideByZeroException
19              // Unhandled exception. System.IndexOutOfRangeException
20              // Unhandled exception. System.NullReferenceException
21
22              // Exception Handling
23
24              try
25              {
26                  Console.Write("a: ");
27                  int a = int.Parse(Console.ReadLine());
28
29                  Console.Write("b: ");
30                  int b = int.Parse(Console.ReadLine());
31
32                  var sonuc = a / b;
33
34                  Console.WriteLine("{0} / {1} = {2}",a,b,sonuc);
35              }
36              catch (DivideByZeroException ex)
37              {
38                  Console.WriteLine("b sıfır olamaz");
39                  Console.WriteLine(ex.Message);
40              }
41              catch(FormatException ex)
42              {
43                  Console.WriteLine("Sayısal bilgi girmelisiniz.");
44                  Console.WriteLine(ex.Message);
45              }
46              catch(Exception ex)
47              {
48                  Console.WriteLine("bir hata oluştu.");
49                  Console.WriteLine(ex.Message);
50              }
51              finally
52              {
53                  Console.WriteLine("finally bloğu çalıştı.");
54              }
```

- Tyr-Catch arasında bu hatalar yakalanabilir.
- 24.Satırdaki tyr içerisine yapılacak işlemler yazılmaktadır.
- Catch içerisine hata yakalanması durumunda yapılacak işlemler yazılmaktadır. Örnek DivideByZeroException ex yazıldığında ex değişken olarak duşunun. 39.satırda hata mesajı yazdırılabilir veya 38. Satırdaki gibi console yazı yazdırabiliriz.
- 46. Satırdaki gibi genel hataları tutabiliriz. Örnek b sıfır olması durumunda veya int yerine string girildiğinde hata vermektedir. Bu hatalar dışında farklı bir hata gelirse 46. Satıra düşecektir.
- 51. Satırdaki finally kısmı Hata olsada olmasada çalışacaktır. Örnek bir dosya açma işleminde sırayla ne olur dosya açılır, yazılır, kapatılır. Şimdi dosya açıldı, yazarken hata oluşsa dahi dosya kapatılmalı.

Hata Yönetimi

```
7       class Person
8       {
9           string _name;
10          public string Name
11          {
12              get {
13                  return _name;
14              }
15              set {
16                  if(value.Length>15)
17                      throw new Exception("Name için en fazla 15 karakter girmelisiniz.");
18                  _name = value;
19              }
20          }
21      }
22      class Program
23      {
24          static void check_password(string password)
25          {
26              if (password.Length<8 || password.Length>15)
27                  throw new Exception("Parola 7-15 karakter arasında olmalıdır.");
28              if (!password.Any(char.IsDigit))
29                  throw new Exception("Parola en az bir rakam içermelidir.");
30              if (!password.Any(char.IsLetter))
31                  throw new Exception("Parola en az bir harf içermelidir");
32          }
```

```
36              // Ornek 1
37
38              string parola = "12345222s";
39              try
40              {
41                  check_password(parola);
42                  Console.WriteLine("parola geçerli.");
43              }
44              catch (Exception ex)
45              {
46                  Console.WriteLine(ex.Message);
47              }
48
49
50              // Ornek 2
51              try
52              {
53                  var p = new Person();
54                  p.Name = "Bilal                                    ";
55              }
56              catch (System.Exception ex)
57              {
58                  Console.WriteLine(ex.Message);
59              }
60          }
61      }
62  }
```

- Örnek 1, 24. Satırdaki metodu kullanmakta, Örnek 2, 7. Satırdaki class'ı kullanmaktadır.
- Hata fırlatma işlemi, bizler belirli durumda programın hata vermesini sağlayabiliriz. Bu işlemthrow ile yapılmaktadır.
- 24. Satırdaki metot ile bir şifrenin karakter uzunluğu 8 ile 15 arasında olması, en az 1 harf içermeli, en az bir harf şartlarını sağlamaması durumunda program hata verecektir.
- 7. Satırdaki class içerisinde oluşturulan name bilgisi eğer karakter uzunluğu 15den fazla ise hata verecektir.
- 54. Satırda boşlukta karakterden saydığı için hata verecektir.

Hata Yönetimi: Uygulama

```
static void Main(string[] args)
{
    // "1","2","5a","10b","abc","10","50"
    // 1- Liste icerisindeki sayisal degerleri bulunuz.

    var liste = new List<string>()
    {
        "1","2","5a","10b","abc","10","50"
    };

    foreach (var item in liste)
    {
        try
        {
            int a = int.Parse(item);
            Console.WriteLine(a);
        }
        catch (Exception)
        {
            continue;
        }
    }

    // 2- Kullanici 'q' degerini girmedikce aldiginiz her degerin
    // sayisal olup olmadigini kontrol edin aksi halde hata mesaji veriniz.

    while(true)
    {
        Console.Write("sayi: ");
        string input = Console.ReadLine();

        if (input == "q")
            break;

        try
        {
            int sayi = int.Parse(input);
            Console.WriteLine("girdiginiz sayi: {0}",sayi);
        }
        catch (Exception)
        {
            Console.WriteLine("gecersiz sayi");
            continue;
        }
    }
```

```
    // 3- Girilen parola icinde turkce karakter araması yapiniz.

    Console.Write("parola: ");
    string parola = Console.ReadLine();

    try
    {
        CheckPassword(parola);
    }
    catch (Exception ex)
    {
        Console.Write(ex.Message);
    }
}

static void CheckPassword(string parola)
{
    string turkce_karakterler = "ğ,Ğ,ü,Ü,ş,Ş,ı,İ,ö,Ö,ç,Ç";

    foreach (var karakter in parola)
    {
        if (turkce_karakterler.IndexOf(karakter)>-1)
            throw new Exception("Parola turkce karakter iceremez");
    }
    Console.Write("gecerli parola");
}
```

- Yukarıda uygulamalar verilmiştir. Genel olarak yeni bir içerik yok, daha önce incelediğimiz kodlar bulunmaktadır.

Kendi Hata Sınıfımızı Yazalım

```
// Exception

// IndexOutOfRangeException => dizi siniri asma durumunda
// Login => username, password

class LoginException: Exception
{
    public LoginException(string message):base(message)
    {

    }
}

class Program
{
    static void Main(string[] args)
    {
        try
        {
            Login("bilalalgann","123456777");
            Console.WriteLine("login basarili.");
        }
        catch (LoginException ex)
        {
            Console.WriteLine(ex.Message);
        }

    }

    static void Login(string username, string password)
    {
        if (username.Contains(" "))
            throw new LoginException("username bosluk iceremez.");

        if (password.Length<8)
            throw new LoginException("Parola min. 8 karakter olmalidir.");
    }
}
```

- 10. Satırdaki gibi bir hata sınıfı oluşturulabilir. Tipini Exception yaptık.