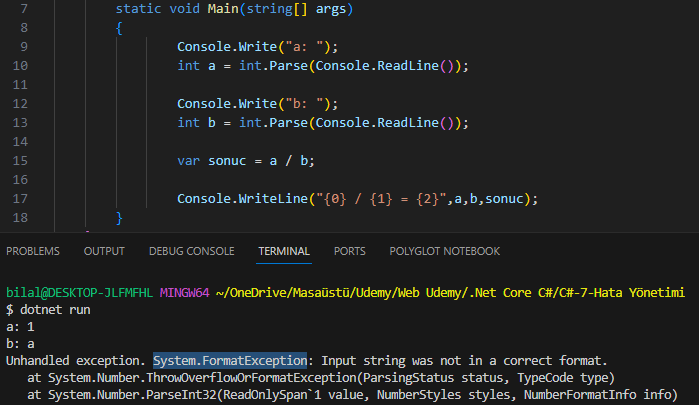
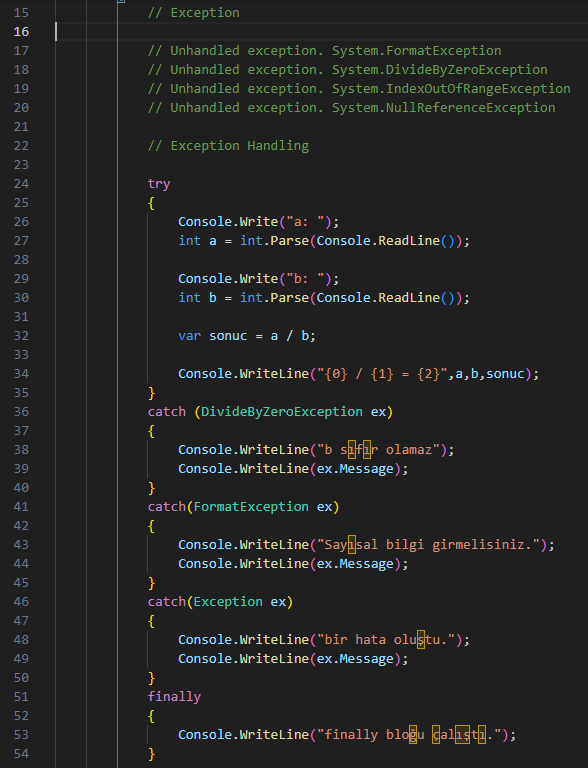
**C# Hata Yönetimi**

**Hata Yönetimi**

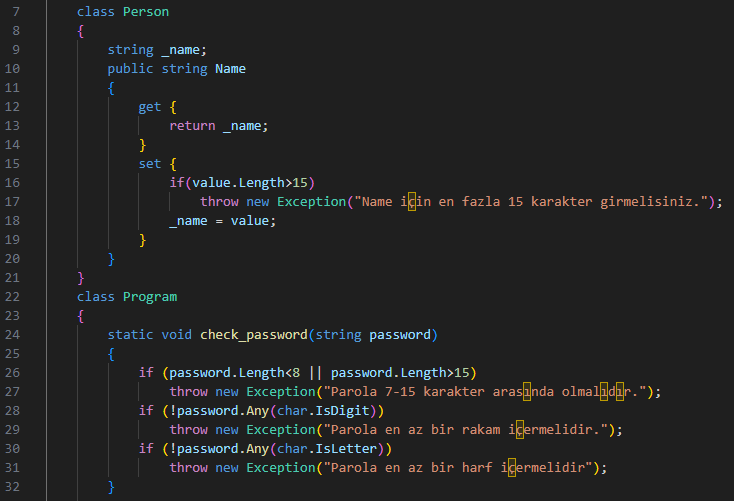


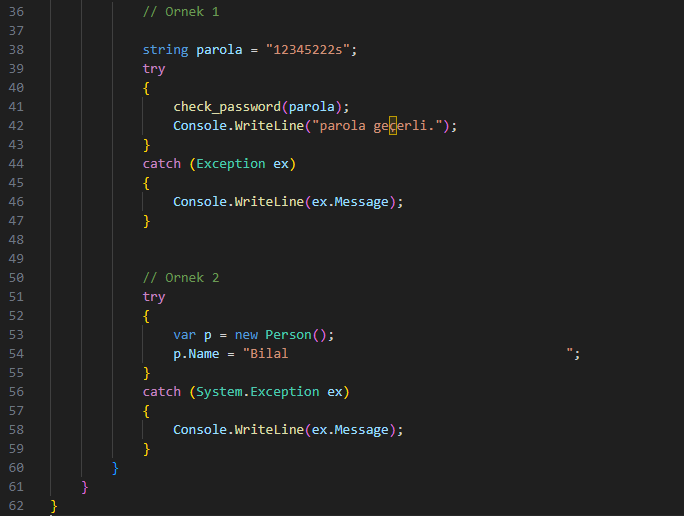
* Örnek b’ye int değer girmek gerekirken string bir değer girince exception altında Formatexception hatası verdi.
* Bu tür örnekleri alıp, kullanıcıların anlayacağı dile getirmek gerekir.



* 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.

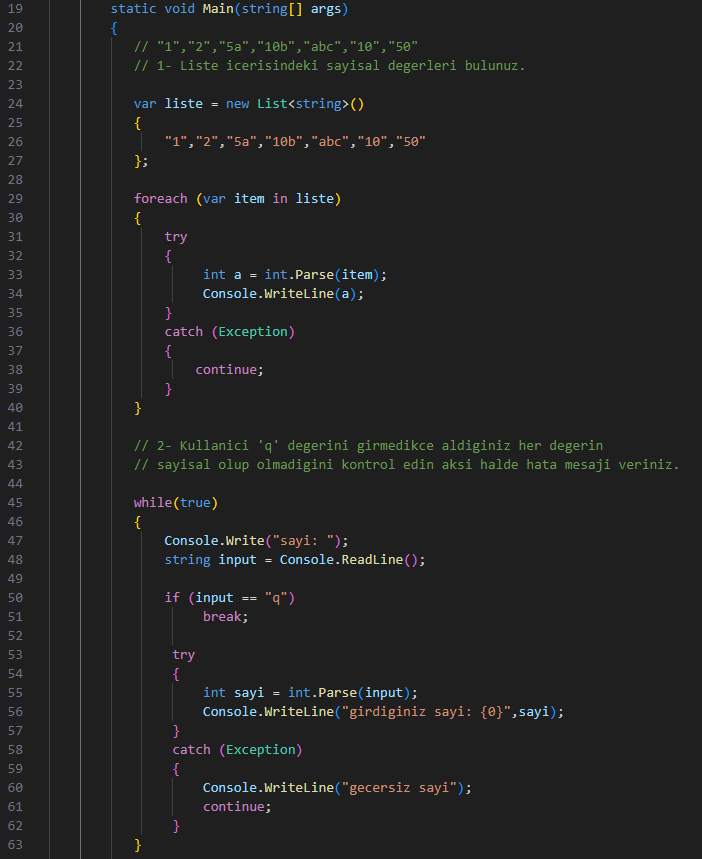
**Hata Yönetimi**

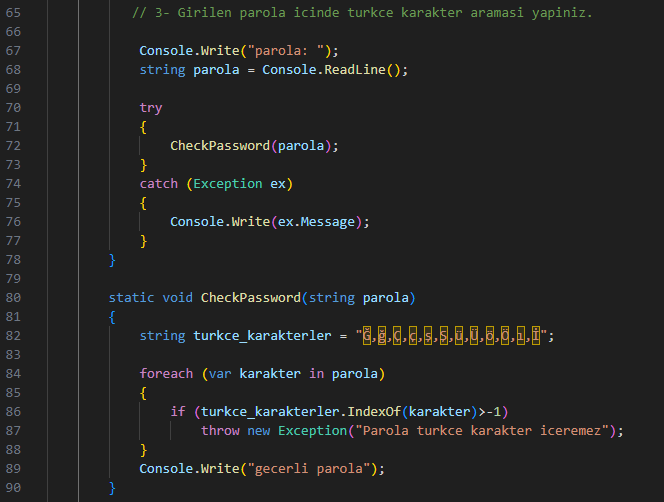




* Örnek 1, 24. Satırdaki metodu kullanmakta, Örnek 2, 7. Satırdaki class’ı kullanmaktadır.
* Hata fırlatma işlemi, bizler belirli durumda programın hata vermesini sağlayabliriz. Bu işlemthrow ile yapılmaktadır.
* 24. Satırdaki metot ile bir şifrenin karakter uzunluğu 8 ile 15 arasında olması, en az 1 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.

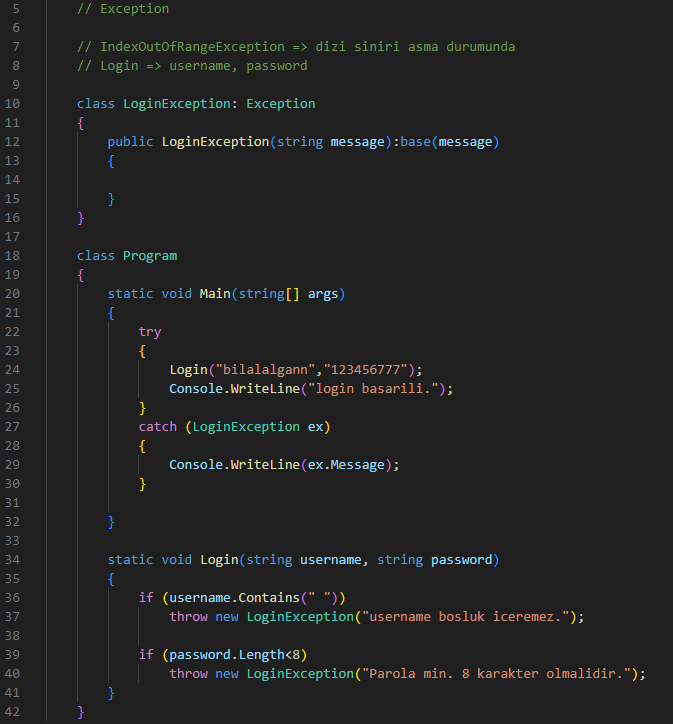
**Hata Yönetimi: Uygulama**





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

**Kendi Hata Sınıfımızı Yazalım**



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