

# PART 16

## 16. İstisnalar(Exceptions)

İstisna, programın çalışması sırasında ortaya çıkan bir sorundur. Bir istisna ,birçok değişik nedenden dolayı meydana gelebilir, aşağıdaki gibi:

- Kullanıcı geçersiz bir veri girebilir.
- Açılması gereken dosya bulunamayabilir.
- İletişimin ortasından ağ bağlantısı kaybolabilir veya JVM belleği tükenebilir.

Bu istisnaların bazılarını kullanıcı hatası sebep olur, diğerleri programcının hatası ve fiziksel kaynaklardan dolayı olabilir.

İstisna işlemenin Javada nasıl çalıştığını anlamak için, istisnaların üç kategorisini anlamanız gerekmektedir.

- **Kontrol edilmiş istisnalar(Checked exceptions):** Kontrol edilmiş istisnalar, genellikle kullanıcı hatları veya programcı tarafından öngörülememiş sorunlardır. Örnek olarak, bir dosya açılacak ancak dosya bulunamıyorsa, istisna meydana gelir.
- **Çalışma anındaki istisnalar(Runtime exceptions):** Çalışma anındaki istisnalar, muhtemelen programcı tarafından önlenebilir istisnalardır.
- **Hatalar:** Başlı başına istisna değildir, kullanıcı veya programcının kontrolü dışında ortaya çıkan sorunlardır. Örnek, bir stack overflow meydana gelirse, bir hata ortaya çıkacaktır. Bunlar, derleme süresinde yok sayılır.

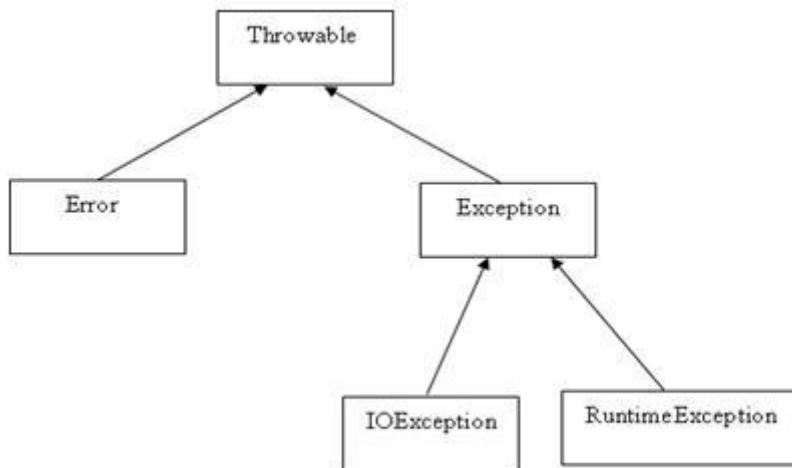
## 16.1 İstisna Hiyerarşisi

Bütün istisna sınıfları, `java.lang.Exception` sınıfının alt türleridir. `Exception` sınıfı, `Throwable` sınıfının alt sınıfıdır. `Throwable` sınıfından elde edilen diğer bir alt sınıfta `Error` sınıfıdır.

Hatalar, normalde Java programları tarafından tutulmazlar. Bu koşullar normalde java programları tarafından işlenmeyen ciddi başarısızlık durumunda meydana gelirler. Hatalar, çalışma ortamı tarafından oluşturulan hataları belirtmek için oluşturulur.Örnek: JVM belleği tükenmiş. Normalde programlar hatalardan kurtulamaz.

`Exception` sınıfı iki ana altsınıfa sahiptir:

- `IOException` Sınıfı
- `RuntimeException` Sınıfı



## 16.2 Java Dahili İstisnaları

En genel istisnalar, standard tip RuntimeException sınıfının altsınıflarıdır.

Java, çeşitli sınıf kütüphaneleri ile ilgili bir çok türde istisna tanımlar. Aşağıdaki liste, Unchecked RuntimeException'ları göstermektedir.

Exception	Description
ArithmeticException	Aritmetic hata, sıfıra bölünme gibi
ArrayIndexOutOfBoundsException	Dizi index'i sınırların dışında.
ArrayStoreException	Dizi elemanına uyumsuz türde atama.
ClassCastException	Geçersiz cast.
IllegalArgumentException	Metodu çalıştırmak için illegal argüman.
IllegalMonitorStateException	Illegal monitor işlemi, unlocked bir thread üzerinde bekleme gibi.
IllegalStateException	Ortam veya uygulama doğru olmayan durumda.
IllegalThreadStateException	İstenen işlem anlık thread durumu ile uyumlu değil.
IndexOutOfBoundsException	Bazı türde index sınır dışında.
NegativeArraySizeException	Dizi negatif boyutta yaratılmış.
NullPointerException	Null referansının geçersiz kullanımı.
NumberFormatException	Bir string'in sayısal formata geçersiz dönüşümü.
SecurityException	Güvenlik ihlali denemesi.
StringIndexOutOfBoundsException	Bir stringe sınırları dışında index verme.
UnsupportedOperationException	Desteklenmeyen işlem karşılatırıldı.

Aşağıdaki liste, Javada ki Checked Exception'ları göstermektedir:

Exception	Description
ClassNotFoundException	Sınıf bulunamadı.
CloneNotSupportedException	Nesne kopyalama, Kopyalanabilir arayüzde uygulanamadı.
IllegalAccessException	Sınıfa erişim engellendi.
InstantiationException	Abstract bir sınıf veya arayüzde nesne oluşturma.
InterruptedException	Bir thread, diğer bir thread tarafından kesildi.
NoSuchFieldException	İstenen dosya yok.
NoSuchMethodException	İstenen metot yok.

## 16.3 İstinaları Yakalamak

Bir metod, try ve catch anahtar kelimelerinin kombinasyonunu kullanarak bir istisnayı yakalayabilir. Try/catch bloğu, istisna oluşması muhtemel kodun etrafına yerleştirilir. Try/catch bloğu içindeki kod, korumalı kod(protected code) olarak adlandırılır ve try/catch kullanımı aşağıdaki gibidir:

```
try
{
    //Protected code
}catch(ExceptionName e1)
{
    //Catch block
}
```

Bir catch ifadesi, yakalamak istediğiniz istisnanın deklare edilmiş türünü içerir. İstisna korumalı kod içinde oluşuyorsa, catch bloğu(veya blokları), try bloğunu kontrol eder. Eğer istisna türü catch bloğu içinde listelenmiş istisnalardan biriye, istisna catch bloğuna geçirilir.

### Örnek

Aşağıdaki dizi, 2 elemanlı bir dizi olarak deklare edilmiştir. Ardından kod, 3. elemete erişmeyi deniyor.Burada bir istisna oluşmaktadır:

```
// File Name : ExcepTest.java
import java.io.*;
public class ExcepTest{

    public static void main(String args[]){
        try{
            int a[] = new int[2];
            System.out.println("Access element three :" + a[3]);
        }catch(ArrayIndexOutOfBoundsException e){
            System.out.println("Exception thrown  :" + e);
        }
    }
}
```

Bu aşağıdaki sonucu üretecektir:

```
Exception thrown :java.lang.ArrayIndexOutOfBoundsException: 3
```

## 16.4 Birden çok catch Bloğu

Bir try bloğu, birden çok catch bloğu tarafından takip edilebilir. Birden çok catch bloğunun syntax'ı aşağıdaki gibidir:

```
try
{
    //Protected code
} catch (ExceptionType1 e1)
{
    //Catch block
} catch (ExceptionType2 e2)
{
    //Catch block
} catch (ExceptionType3 e3)
{
    //Catch block
}
```

Önceki ifadeler üç catch bloğu göstermektedir, ama siz tek bir try bloğundan sonra herhangi bir sayıda catch bloğu kullanabilirsiniz. Eğer istisna korumalı kod içinde meydana gelirse, listedeki ilk catch bloğuna atılır. Atılan istisnanın veri tipi ExceptionType1 ile eşleşiyorsa, burada yakalanır. Eşleşmiyorsa, istisna ikinci catch bloğuna geçer. Bu işlem istisna yakalanana kadar devam eder, bu durumda metot çalışmayı durdurur ve istisna call stack üzerindeki bir önceki metodu altına alır.

### Örnek

Birden çok try/catch ifadesinin kullanımını gösteren kod parçası:

```
try
{
    file = new FileInputStream(fileName);
    x = (byte) file.read();
} catch (IOException i)
{
    i.printStackTrace();
    return -1;
} catch (FileNotFoundException f) //Not valid!
{
    f.printStackTrace();
    return -1;
}
```

## 16.5 throws/throw Anahtar Kelimeleri

Eğer bir metot kontrol edilmiş bir istisnayı işleyemiyorsa, o zaman **throws** anahtar kelimesi kullanarak deklare etmek zorundadır. throws anahtar kelimesi metot

imzasının sonunda görülür. **Throws** anahtar kelimesi istisnayı bir başka sınıfa gönderir.

Aşağıdaki metot bir RemoteException atmanın deklare edilmesidir:

```
import java.io.*;
public class className
{
    public void deposit(double amount) throws RemoteException
    {
        // Method implementation
        throw new RemoteException();
    }
    //Remainder of class definition
}
```

Bir metot, birden çok istisnanın atılmasını deklare eder, bu durumda istisnalar virgülle ayrılarak bir listede deklare edilir. Örnek, aşağıdaki metot bir RemoteException ve bir InsufficientFundsException atılmasını deklare ediyor:

```
import java.io.*;
public class className
{
    public void withdraw(double amount) throws RemoteException, InsufficientFundsException
    {
        // Method implementation
    }
    //Remainder of class definition
}
```

## 16.6 finally Anahtar Kelimesi

Finally anahtar kelimesi, try bloğunun ardından gelen bir kod bloğu oluşturmak için kullanılır. Bir finally kod bloğu, istisna olsa da olmasa da, her zaman çalışır.

Catch bloklarından sonra gelen finally bloğunun syntax'ı şu şekildedir.

```
try
{
    //Protected code
} catch (ExceptionType1 e1)
{
    //Catch block
} catch (ExceptionType2 e2)
{
    //Catch block
} catch (ExceptionType3 e3)
{
    //Catch block
}
```

```
}finally  
{  
    //The finally block always executes.  
}
```

## Örnek

```
public class ExcepTest{  
  
    public static void main(String args[]){  
        int a[] = new int[2];  
        try{  
            System.out.println("Access element three :" + a[3]);  
        }catch(ArrayIndexOutOfBoundsException e){  
            System.out.println("Exception thrown  :" + e);  
        }  
        finally{  
            a[0] = 6;  
            System.out.println("First element value: " +a[0]);  
            System.out.println("The finally statement is executed");  
        }  
    }  
}
```

Bu aşağıdaki sonucu üretecektir:

```
Exception thrown  :java.lang.ArrayIndexOutOfBoundsException: 3  
First element value: 6  
The finally statement is executed
```

Aşağıdakilere dikkat edin:

- Bir catch koşulu, try ifadesi olmadan var olamaz.
- try/catch bloğu olsa bile, finally koşulu kullanmak zorunlu değildir.
- Catch koşulu veya finally koşulu olmazsa, try bloğu olmaz.