Java Exceptions

| **Hata Tipleri** | **Hatanın Anlamı** |
| --- | --- |
| ArithmeticException | Aritmatiksel bir işlem sırasında oluşan hata |
| ArrayIndexOutOfBoundsException | Array sınırlarından çıkılma hatası |
| ArrayStoreException | Array yapısına uymayan bir değer kaydedilmeye çalışıldığında |
| ClassCastException | Uygunsuz class atama işlemi |
| IllegalArgumentException | Uygun olmayan argüman kullanımı |
| IllegalMonitorStateException | Uygun olmayan monitör işlemi |
| IllegalStateException | Sistemin uygun pozisyonda olmadığını belirtir |
| IllegalThreadStateException | Tread durumunun uygun olmadığını belirtir |
| IndexOutOfBoundsException | Index'in sınırlardan çıkılma hatası |
| NegativeArraySizeException | Array'in boyutuna negatif değer verilmiş olması hatası |
| NullPointerException | Boş pointer hatası |
| NumberFormatException | Uygun olmayan sayı formatı hatası |
| SecurityException | Güvenlik hatası |
| StringIndexOutOfBounds | String yapısının sınırlarından çıkma hatası |
| UnsupportedOperationException | Desteklenmeyen işlem hatası |
| ClassNotFoundException | Class bulunmama hatası |
| CloneNotSupportedException | Çoğalmaya çalışılan bir objenin çoğalamayacağını belirten hata |
| IllegalAccessException | Uygunsuz erişim hatası |
| InstantiationException | Bir objenin oluşturulması esnasında oluşan hata |
| InterruptedException | Bir tread'in diğer tread'i durdurma hatası |
| NoSuchFieldException | Aranılan alanın olmadığı hatası |
| NoSuchMethodException | Kıllanılan methodun bulunmama hatası |

public class MyClass {

public static void main(String[ ] args) {

int[] myNumbers = {1, 2, 3};

System.out.println(myNumbers[10]); // error!

}

}

public class MyClass {

public static void main(String[ ] args) {

try {

int[] myNumbers = {1, 2, 3};

System.out.println(myNumbers[10]);

} catch (Exception e) {

System.out.println("Something went wrong.");

} }}

public class MyClass {

public static void main(String[] args) {

try {

int[] myNumbers = {1, 2, 3};

System.out.println(myNumbers[10]);

} catch (Exception e) {

System.out.println("Something went wrong.");

} finally {

System.out.println("The 'try catch' is finished.");

} }}

İstisna, işlerin kontrolden çıkarak beklenmeyen durumlarda ortaya çıkamasıdır. Java’da istisna işleme konusunda 5 anahtar sözcük kullanılmaktadır. Bunlar; try-catch, throw, throws, finally ‘dir.

## Try — Catch Blokları

İstisnaların oluşma aşamasını incelersek;  
1- İstisna meydana geldiğinde ilk olarak bir nesne oluşturulur.  
2- Bu istisnayı yakalamak için catch bloğu olup olmadığı kontrol edilir. Catch bloğu varsa uygulama sonlanmaz bu catch bloğundan devam eder.  
3- Beklenen Catch bloğu yok ise istisna; istisnanın meydana geldiği komuta çağıran satıra istisna nesnesi fırlatılır.  
4- Şayet söz konusu istisnanın old. komutu çağıran komut satırında ilgili istisnaya ait catch bloğu var ise uyg. bu catch bloğundan devam eder.  
5- Burada da catch bloğu yoksa istisna oluşturulduğu metodu çağıran bir üst metoda fırlatılır. Catch olup olmadığı kontrol edilir varsa buradan yoksa bir üst metoda fırlatılır bu işlem uygulamadaki main metoduna kadar devam eder yoksa uygulama sonlanır.

## İstisna Sınıfları

İstisnaların yakalanarak ele alındığı sınıflardır. Bu sınıflar en süper sınıf olan Object sınıfından türetilmiş THROWABLE sınıfından türetilmektedir. Throwable sınıfı 2 alt başlıkta toplanır. Error ve Exception sınıfları. Error sınıfı sistem kaynaklı sorunlar meydana geldiğinde çalıştırılmaktadır. Bu durumda yazılımcıların birşey yapması imkansızdır. Exception sınıfı ise çalışma sırasında oluşan hataları ele almaktadır. Bu hatalar çoğunlukla uygulama geliştirilirken yapılan hatalardan kaynaklanmaktadır.

**Kontrollü İstisnalar:** Derleyicinin try-catch bloklarını kullanmaya zorlandığı istisna şeklidir.

**Kontrolsüz İstisnalar:** Derleyicinin try-catch bloklarını kullanmaya zorlamadığı istisna şeklidir.

## İstisna Sınıfının Yapılandırıcıları

Tüm istisna sınıfları 2 yapılandırıcı barındırırlar. Bir tanesi hiç parametre almayan yapılandırıcı diğeri ise istisnanın açıklandığı String tipinden parametre alan yapılandırıcıdır. Bu parametre istisnanın mesaj alanına atanmakta ve getMessage yöntemi ile alınmaktadır.

Error ve Exception sınıfına ait birkaç alt sınıfı inceleyelim:

1- **Error Sınıfı:**  
**VirtualMachineError:** JVM’nin çalışmasını etkileyen durumları inceler.  
**AWTError:** Grafik arayüze ait hataları inceler.  
**OutOfMemoryError:** Bellek yetersizliği durumlarını inceler.

2- **Exception Sınıfı:**  
**ClassNotFoundException**: Olmayan bir dosyaya erişme istediği durumlarını inceler.  
**IOException**: Giriş çıkış işlemlerindeki istenmeyen durumları inceler.  
**RunTimeException**: Çalışma zamanı hatalarını inceler.  
**AritmeticException**: Aritmetik hataları inceler.  
**NullPointerException**: Herhangi bir nesneye null referanslı bir değişken ile ulaşılmaya çalışılan durumlarda fırlatılır.  
**IllegalArgumentException**: Metotlara geçersiz argüman atamalarında fırlatılır.

istisna durumları bunlarla sınırlı değildirler. Değişik istisna sınıflarıda bulunmaktadır.

## İstisna Sınıflarında Ortak Kullanılan Metotlar

**getMessage() metodu:** İstisna nesnesi içersindeki mesajı döner.  
**toString():** istisna mesajını istisna adını ili nokta ve arada bir boşluk ile döner.  
**printStackTrace():** oluşan istisnanın bilgilerini aşama aşama ekrana yazdırır.  
**getStackTrace():** oluşan istisnanın bilgilerini aşama aşama stacktrace tipinde bir dizi olarak döner.

## Throw Exceptions

Throw ifadesini kullanarak açıkça bir hata fırlatılabilir. Throw Exceptions ‘na ait detaylı dökümana [buradan](https://docs.oracle.com/javase/tutorial/essential/exceptions/throwing.html" \t "_blank) ulaşabilirsiniz.

throw new IOException ("Hata");

## Throws Exceptions

Eğer bir metot istisnaya sebep olabilirse, metot çağırıcılarına karşı kendini koruyabilmesi adına, metodun bu davranışı belirtmesi gerekir. Bunu aşağıdaki biçimde yapabiliriz:

public class App {  
 public static void main(String[] args) throws FileNotFoundException {

class Example1 {

public static void main(String args[]) {

int num1, num2;

try {

  /\* We suspect that this block of statement can throw

\* exception so we handled it by placing these statements

\* inside try and handled the exception in catch block

\*/

num1 = 0;

num2 = 62 / num1;

System.out.println(num2);

System.out.println("Hey I'm at the end of try block");

}

  catch (ArithmeticException e) {

/\* This block will only execute if any Arithmetic exception

\* occurs in try block

\*/

System.out.println("You should not divide a number by zero");

}

catch (Exception e) {

/\* This is a generic Exception handler which means it can handle

\* all the exceptions. This will execute if the exception is not

\* handled by previous catch blocks.

\*/

System.out.println("Exception occurred");

}

System.out.println("I'm out of try-catch block in Java.");

}

}

class Example2{

public static void main(String args[]){

try{

int a[]=new int[7];

a[4]=30/0;

System.out.println("First print statement in try block");

}

catch(ArithmeticException e){

System.out.println("Warning: ArithmeticException");

}

catch(ArrayIndexOutOfBoundsException e){

System.out.println("Warning: ArrayIndexOutOfBoundsException");

}

catch(Exception e){

System.out.println("Warning: Some Other exception");

}

System.out.println("Out of try-catch block...");

}

}