Nesne Yönelimli Programlama

Hazırlayan: M.Ali Akcayol
Gazi Üniversitesi
Bilgisayar Mühendisliği Bölümü

Not: Bu dersin sunumları, "Java Programlama Dili ve Yazılım Tasarımı, Altuğ B. Altıntaş, Papatya Yayıncılık, 2016" kitabı kullanılarak hazırlanmıştır.

Konular

- İstisnalar
- İstisna Oluşumu
- İstisna Yakalama
- İstisna İfadeleri
- İstisna Tip Hiyerarşisi
- **`RuntimeException`** İstisna Tipleri
- İstisna Mesajları
- Yeni İstisna Oluşturmak
- **`finally`** Bloğu
- **`finally`** Bloğu ve **`return`**
- **`finally`** Bloğu ve **`System.exit()`**

İstisnalar

- Uygulamaların **iki önemli görevi vardır:**
  - **Doğruluk:** Kendisinden beklenen görevleri doğru bir şekilde yerine getirmelidir.
  - **Sağlamlık:** Hatalı davranışlara karşı dayanıklı olmalıdır.
- Örneğin, (A / B - B) gibi bir aritmetik işlemde sonucu doğru hesaplamalı ve yanlış girişlere karşı dayanıklı olmalıdır.
- Java, **hata oluşmasına neden olabilecek bir durum var ise yazılan kodu derlemez.**
- Java, ileride oluşabilecek ve bulunması çok güç olan hataların erkenden engellenmesini amaçlar.

3

Konular

- İstisnalar
- İstisna Oluşumu
- İstisna Yakalama
- İstisna İfadeleri
- İstisna Tip Hiyerarşisi
- `RuntimeException` İstisna Tipleri
- İstisna Mesajları
- Yeni İstisna Oluşturmak
- `finally` Bloğu
- `finally` Bloğu ve `return`
- `finally` Bloğu ve `System.exit()`

İstisna Oluşumu

- Aşağıdaki örnekte diziye erişim hatası oluşur.
- **`ArrayIndexOutOfBoundsException`** istisnası oluşur.

```
public class Istisnalar {

    public static void main(String args[]) {

        int sayilar[] = { 1, 2, 3, 4 };
        System.out.println("Basla");
        for (int i = 0; i < 5; i++) {
            System.out.println("--> " + sayilar[i]);
        }
        System.out.println("Bitti");
    }
}
```

```
Basla
--> 1
--> 2
--> 3
--> 4
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 4
        at Istisnalar.Istisnalar.main(Istisnalar.java:10)
```

5

İstisna Oluşumu

- Bir **uygulama içerisinde** genel olarak **aşağıdaki istisnalar oluşabilir:**
  - Açılmak istenilen fiziksel **dosya olmayabilir.**
  - Kullanıcılar tarafından **beklenmedik bir veri girişi yapılabilir.**
  - **Ağ bağlantısı kopmuş olabilir.**
  - Yazılmak istenilen **dosya**, **başkası tarafından açılmış olduğundan yazma hakkı olmayabilir.**

6

Konular

- İstisnalar
- İstisna Oluşumu
- İstisna Yakalama
- İstisna İfadeleri
- İstisna Tip Hiyerarşisi
- **`RuntimeException`** İstisna Tipleri
- İstisna Mesajları
- Yeni İstisna Oluşturmak
- **`finally`** Bloğu
- **`finally`** Bloğu ve **`return`**
- **`finally`** Bloğu ve **`System.exit()`**

İstisna Yakalama

- Oluşan bu **istisnayı yakalayıp uygulamanın devam etmesini sağlamak mümkündür.**
- İstisnaya neden olabilecek kod, **try-catch** bloğunun içerisinde tutularak güvenlik altına alınmış olur.
- **İstisna oluşursa**, istisna **yakalama mekanizması devreye girer.**
- Oluşan istisnanın tipine göre, uygulamanın akışı catch bloklarından birinin içerisine yönlenerek devam eder.

```
try {
      // Istisnaya sebebiyet verebilecek olan kod
} catch(Exception1 e1) {
      //Eger Exception1 tipinde istisna firlatilirsa buraya
} catch(Exception2 e2) {
      //Eger Exception2 tipinde istisna firlatilirsa buraya
}
```

- İstisna nesnedir ve oluştuğunda çok sayıda olay gerçekleşir.

8

İstisna Yakalama

- Bir istisna nesnesi belleğin heap alında **`new()`** ile oluşturulur.
- İstisna nesnesinin içerisine hatanın oluştuğu satır yerleştirilir.
- İstisnayı yakalamak için **catch bloğunun olup olmadığına bakılır.**
- catch bloğu varsa, **uygulamanın akışı** uygun **catch bloğunun içerisinden devam eder.**
- catch bloğu yoksa, hatanın oluştuğu **yordamı çağıran yordama istisna nesnesi gönderilir.**
- Eğer bu yordamda da istisnayı yakalamak için catch bloğu tanımlanmamış ise, istisna nesnesi bir üst yordama gönderilir.
- Bu sıra devam eder ve **en sonunda main() yordamına ulaşan istisna nesnesi için bir catch bloğu aranır.**
- Eğer **main içerisinde de catch bloğu tanımlanmamış ise**, **uygulananın akışı sonlanır.**

9

İstisna Yakalama

```
public class Istisnalar {

    public void calis() {
        int sayilar[] = { 1, 2, 3, 4 };
        for (int i = 0; i < 5; i++) {
            try {
                System.out.println("--> " + sayilar[i]);
            } catch (ArrayIndexOutOfBoundsException ex) {
                System.out.println("Dizide olmayan elemana erişim hatası oluştu! " + ex);
            }
        } // for
    }

    public static void main(String args[]) {

        System.out.println("Başlangıç");
        Istisnalar de2 = new Istisnalar();
        de2.calis();
        System.out.println("Bitiş");
    }
}
```

```
Başlangıç
--> 1
--> 2
--> 3
--> 4
Dizide olmayan elemana erişim hatası oluştu! java.lang.ArrayIndexOutOfBoundsException: 4
Bitiş
```

10

İstisna Yakalama

- Önceki örnekte try-catch bloğu for bloğunun dışında da kullanılabilir.

```
public class Istisnalar {

    public void calis() {
        int sayilar[] = { 1, 2, 3, 4 };
        try {
            for (int i = 0; i < 5; i++) {
                System.out.println("--> " + sayilar[i]);
            } // for
        } catch (ArrayIndexOutOfBoundsException ex) {
            System.out.println("Dizide olmayan elemana erişim hatası oluştu! " + ex);
        }
    }

    public static void main(String args[]) {

        System.out.println("Başlangıç");
        Istisnalar de2 = new Istisnalar();
        de2.calis();
        System.out.println("Bitiş");
    }
}
```

11

Konular

- İstisnalar
- İstisna Oluşumu
- İstisna Yakalama
- İstisna İfadeleri
- İstisna Tip Hiyerarşisi
- **`RuntimeException`** İstisna Tipleri
- İstisna Mesajları
- Yeni İstisna Oluşturmak
- **`finally`** Bloğu
- **`finally`** Bloğu ve **`return`**
- **`finally`** Bloğu ve **`System.exit()`**

İstisna İfadeleri

- Bir yordam hangi tür istisnanın oluşabileceğini önceden belirtebilir veya belirtmek zorunda kalabilir.
- Yordamı çağıran diğer **yordamlar** oluşabilecek istisnayı, ya **yakalayıp gerekli işlemleri yaparlar** ya da **kendilerini çağıran yordama iletirler.**
- **İstisnanın oluştuğu yordam içerisinde**, o istisna nesnesi ile **ne yapılacağı bilenemeyebilir.**
- İstisnanın oluştuğu yordamdan, **kendisini çağıran yordama istisna nesnesi gönderilebilir.**
- Bir istisna oluştuğunda, **catch bloğunun içerisinde gerekli işlemler yapılarak uygulamanın çalışması sağlanmalıdır.**
- Telafi edilemez bir hata oluşursa, **hata mesajını kaydederek program sonlandırılmalıdır.**

13

İstisna İfadeleri

- Aşağıdaki örnekte **dosya işlemlerinin try-catch bloğu içerisine alınması gereklidir.**

```
import java.io.*;

public class Istisnalar {

    public void cokCalis() {
        File f = new File("ornek.txt");
        BufferedReader bf = new BufferedReader(new FileReader(f));
        System.out.println(bf.readLine());
    }

    public void calis() {
        cokCalis();
    }

    public static void main(String args[]) {
        Istisnalar io1 = new Istisnalar();
        io1.calis();
    }
}
```

14

İstisna İfadeleri

- Önceki örnekteki programa **try-catch bloğu eklenmiştir.**

```
import java.io.*;

public class Istisnalar {

    public void cokCalis() {
        try {
            File f = new File("ornek.txt");
            BufferedReader bf = new BufferedReader(new FileReader(f));
            System.out.println(bf.readLine());
        } catch (IOException ex) {
            System.out.println("Hata Yakalandi =" + ex);
        }
    }

    public void calis() {
        cokCalis();
        System.out.println("calis() yordamı");
    }

    public static void main(String args[]) {
        Istisnalar io2 = new Istisnalar();
        io2.calis();
        System.out.println("main() yordamı");
    }
}
```

```
Hata Yakalandi =java.io.FileNotFoundException: ornek.txt (Sistem belirtilen dosyayı bulamıyor)
calis() yordamı
main() yordamı
```

15

İstisna İfadeleri

- İstisna nesnesi **çağıran yordama** gönderilmektedir.

```
import java.io.*;

public class Istisnalar {
    public void cokCalis() throws IOException {
        File f = new File("ornek.txt");
        BufferedReader bf = new BufferedReader(new FileReader(f));
        System.out.println(bf.readLine());
    }
    public void calis() {
        try {
            cokCalis();
            System.out.println("calis() yordamı");
        } catch (IOException ex) {
            System.out.println("Hata Yakalandi-calis() =" + ex);
        }
    }
    public static void main(String args[]) {
        Istisnalar io3 = new Istisnalar();
        io3.calis();
        System.out.println("main() yordamı");
    }
}
```

```
Hata Yakalandi-calis() =java.io.FileNotFoundException: ornek.txt (Sistem belirtilen dosyayı bulamıyor)
main() yordamı
```

İstisna İfadeleri

- İstisna nesnesi **main() yordamına** gönderilmektedir.

```
public class Istisnalar {
    public void cokCalis() throws IOException {
        File f = new File("ornek.txt");
        BufferedReader bf = new BufferedReader(new FileReader(f));
        System.out.println(bf.readLine());
    }
    public void calis() throws IOException {
        cokCalis();
        System.out.println("calis() yordamı");
    }
    public static void main(String args[]) {
        try {
            Istisnalar io4 = new Istisnalar();
            io4.calis();
            System.out.println("main() yordamı");
        } catch (IOException ex) {
            System.out.println("Hata Yakalandi-main() =" + ex);
        }
    }
}
```

```
Hata Yakalandi-main() =java.io.FileNotFoundException: ornek.txt (Sistem belirtilen dosyayı bulamıyor)
```

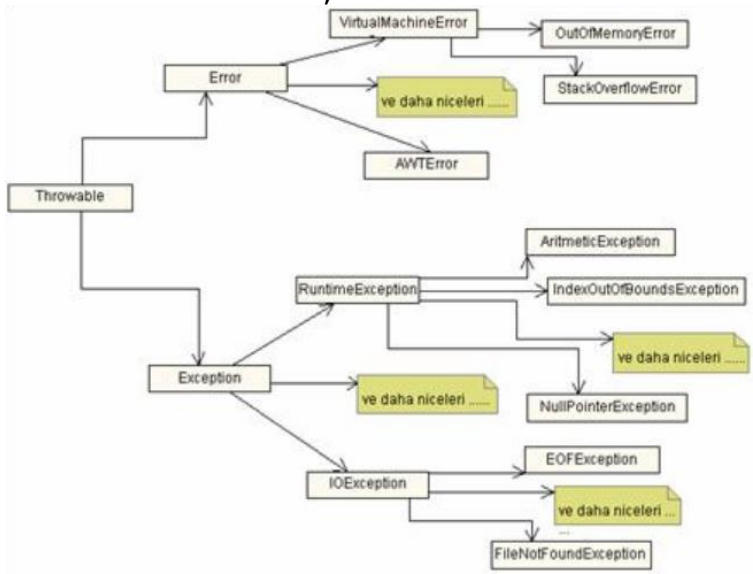
17

Konular

- İstisnalar
- İstisna Oluşumu
- İstisna Yakalama
- İstisna İfadeleri
- İstisna Tip Hiyerarşisi
- **`RuntimeException`** İstisna Tipleri
- İstisna Mesajları
- Yeni İstisna Oluşturmak
- **`finally`** Bloğu
- **`finally`** Bloğu ve **`return`**
- **`finally`** Bloğu ve **`System.exit()`**

İstisna Tip Hiyerarşisi

- Throwable istisna nesnesi, tüm istisna nesnelerinin atasıdır.



19

İstisna Tip Hiyerarşisi

- **Throwable** sınıfı **Object sınıfından türetilmiştir.**
- İstisnalar 3 gruba ayrılabilir:
  - **Error:** Ölümcül bir hatayı işarettir ve telafisi neredeyse imkansızdır. Örneğin, **OutOfMemoryError** (yetersiz bellek) istisnası oluşmuş ise uygulamanın buna müdahele edip düzeltmesi imkansızdır.
  - **RuntimeException:** Uygulama normal çalışırsa ortaya çıkmaması gereken istisna tipleridir. Kontrolsüz kodlamadan dolayı meydana gelen istisna tipleridir
    Örneğin, **ArrayIndexOutOfBoundsException** istisna tipi, bir dizinin olmayan elemanına eriştiğimiz zaman ortaya çıkan bir istisnadır.
  - **Diğer Exception tipleri:** Bu istisnalar çevresel koşullardan dolayı meydana gelebilir.
    Örneğin, erişmeye çalışan dosyanın yerinde olmaması **(FileNotFoundException)** veya network bağlantısının kopması sonucu ortaya çıkabilecek istisnalardır ve bu istisnalar için önceden bir tedbir alınması şarttır.

20

İstisna Tip Hiyerarşisi

- Bir uygulama içerisinde oluşabilecek olan **tüm istisna tiplerini yakalamak için aşağıdaki ifade kullanılabilir:**

```
catch (Exception ex) {
      //......
}
```

- **Tüm istisnaları yakalamak** (`Error`, `RuntimeException` ve diğer Exception türleri) için **`Throwable` istisna tipi kullanılabilir.**
- Ancak, oluşabilecek istisnalar için bu üç gruba ait istisna tiplerinin kullanılması daha uygundur.

21

Konular

- İstisnalar
- İstisna Oluşumu
- İstisna Yakalama
- İstisna İfadeleri
- İstisna Tip Hiyerarşisi
- **`RuntimeException`** İstisna Tipleri
- İstisna Mesajları
- Yeni İstisna Oluşturmak
- **`finally`** Bloğu
- **`finally`** Bloğu ve **`return`**
- **`finally`** Bloğu ve **`System.exit()`**

`RuntimeException` İstisna Tipleri

- Bir dizide olmayan bir elemanına erişilmeye çalışılırsa, **`ArrayIndexOutOfBoundsException`** hatası olur (**`RuntimeException`**).
- **Bu tür hatalar derleme anında bilinemez.**
- Java, **bir dosyaya erişirken** oluşacak **istisnaya karşı tedbir alınmasını zorunlu tutar** (diğer Exception tipleri).
- Runtime istisna hataları:
  - **AritmeticException:** Bir sayının sıfıra bölünmesiyle ortaya çıkar.
  - **NullPointerException:** Bir sınıf tipindeki referansı, o sınıfa ait bir nesneye bağlamadan kullanmaya kalkınca ortaya çıkar.
  - **NegativeArraySizeException:** Bir diziyi negatif bir sayı vererek oluşturmaya çalışınca ortaya çıkar.
  - **ArrayIndexOutOfBoundsException:** Bir dizinin olmayan elemanına ulaşmak istendiğinde ortaya çıkar.

23

Konular

- İstisnalar
- İstisna Oluşumu
- İstisna Yakalama
- İstisna İfadeleri
- İstisna Tip Hiyerarşisi
- **`RuntimeException`** İstisna Tipleri
- İstisna Mesajları
- Yeni İstisna Oluşturmak
- **`finally`** Bloğu
- **`finally`** Bloğu ve **`return`**
- **`finally`** Bloğu ve **`System.exit()`**

İstisna Mesajları

- **İstisna nesnesinden bir çok veri elde edilebilir.**
- **İstisna oluşumunun yol haritası izlenebilir** veya istisna oluşana kadar **hangi yordamların çağrıldığı öğrenilebilir.**
- **Bu bilgileri elde etmek için** kullanılan **Throwable sınıfına ait yordamlar:**
  - **getMessage():** İstisnaya ait bilgileri **`String`** tipinde geri döndürür.
  - **getLocalizedMessage():** Exception sınıfından türetilmiş alt sınıflar tarafından override yapılmalıdır.
    Bu yordam alt sınıflar tarafından iptal edilmemişse **`getMessage()`** yordamı ile aynı sonucu döndürür.
  - **toString():** İstisna hakkındaki açıklamayı **`String`** tipinde geri döndürür.

İstisna Mesajları

```
public class Istisnalar {
    public void oku() throws Exception {
        throw new Exception("istisna firlatildi"); // dikkat
    }

    public static void main(String args[]) {
        try {
            Istisnalar im = new Istisnalar();
            im.oku();
        } catch (Exception ex) {
            System.out.println("Hata- ex.getMessage() : " + ex.getMessage());
            System.out.println("Hata-ex.getLocalizedMessage() : " + ex.getLocalizedMessage());
            System.out.println("Hata- ex.toString() : " + ex);
        }
    }
}
```

```
Hata- ex.getMessage() : istisna firlatildi
Hata-ex.getLocalizedMessage() : istisna firlatildi
Hata- ex.toString() : java.lang.Exception: istisna firlatildi
```

Konular

- İstisnalar
- İstisna Oluşumu
- İstisna Yakalama
- İstisna İfadeleri
- İstisna Tip Hiyerarşisi
- **`RuntimeException`** İstisna Tipleri
- İstisna Mesajları
- Yeni İstisna Oluşturmak
- **`finally`** Bloğu
- **`finally`** Bloğu ve **`return`**
- **`finally`** Bloğu ve **`System.exit()`**

Yeni İstisna Oluşturmak

- Java'nın kendi içerisinde tanımlanmış istisna tiplerinin dışında, **uygulamaya özgü istisna tipleri oluşturulabilir.**

```
public class BenimHatam extends Exception {
    private int id;
    public BenimHatam() {
    }
    public BenimHatam(String aciklama) {
        super(aciklama); // dikkat
    }
    public BenimHatam(String aciklama, int id) {
        super(aciklama); // dikkat
        this.id = id;
    }
    public String getLocalizedMessage() { // iptal etme (override)
        switch (id) {
        case 0:
            return "ÖNEMSİZ HATA";
        case 1:
            return "HATA";
        case 2:
            return "ÖNEMLİ HATA";
        default:
            return "TANIMSIZ HATA";
        }
    }
    public int getId() {
        return id;
    }
}
```

28

Yeni İstisna Oluşturmak

Örnek

```
class BenimHatam extends Exception {
    private int id;

    public BenimHatam() {
    }

    public BenimHatam(String aciklama) {
        super(aciklama); // dikkat
    }

    public BenimHatam(String aciklama, int id) {
        super(aciklama); // dikkat
        this.id = id;
    }

    public String getLocalizedMessage() { // iptal etme (override)
        switch (id) {
        case 0:
            return "ÖNEMSİZ HATA";
        case 1:
            return "HATA";
        case 2:
            return "ÖNEMLİ HATA";
        default:
            return "TANIMSIZ HATA";
        }
    }

    public int getId() {
        return id;
    }
}
```

29

Yeni İstisna Oluşturmak

Örnek

```
class SeninHatan extends Exception {
    public SeninHatan() {
    }

    public SeninHatan(String aciklama) {
        super(aciklama); // dikkat
    }
}

public class Istisnalar {
    public void cikart(int a, int b) throws BenimHatam, SeninHatan {
        if (a == 0) {
            throw new SeninHatan("a parametresi sifir geldi");
        }
        if (b == 0) {
            throw new SeninHatan("b parametresi sifir geldi");
        }
        if ((a < 0) || (b < 0)) {
            throw new SeninHatan(); // kotu, aciklama yok
        }
        int sonuc = a - b; // hesaplama islemi

        if (sonuc < 0) {
            throw new BenimHatam("sonuc eksi", 2);
        } else if (sonuc == 0) {
            throw new BenimHatam("sonuc sifir", 1);
        }
    }
```

30

Yeni İstisna Oluşturmak

Örnek

```
----------------------
Hata Olustu-1:sonuc eksi
ÖNEMLİ HATA
2
----------------------
Hata Olustu-2:Istisnalar.SeninHatan: b parametresi sifir geldi
----------------------
Hata Olustu-2:Istisnalar.SeninHatan
```

```
    public static void main(String args[]) {
        System.out.println("----------------------");
        try {
            Istisnalar it = new Istisnalar();
            it.cikart(1, 2);
        } catch (BenimHatam ex1) {
            System.out.println("Hata Olustu-1:" + ex1.getMessage());
            System.out.println(ex1.getLocalizedMessage());
            System.out.println(ex1.getId());
        } catch (SeninHatan ex2) {
            System.out.println("Hata Olustu-2:" + ex2);
        }
        System.out.println("----------------------");
        try {
            Istisnalar it = new Istisnalar();
            it.cikart(1, 0);
        } catch (BenimHatam ex1) {
            System.out.println("Hata Olustu-1:" + ex1.getMessage());
            System.out.println(ex1.getLocalizedMessage());
            System.out.println(ex1.getId());
        } catch (SeninHatan ex2) {
            System.out.println("Hata Olustu-2:" + ex2);
        }

        System.out.println("----------------------");
        try {
            Istisnalar it = new Istisnalar();
            it.cikart(1, -124);
        } catch (BenimHatam ex1) {
            System.out.println("Hata Olustu-1:" + ex1.getMessage());
            System.out.println(ex1.getLocalizedMessage());
            System.out.println(ex1.getId());
        } catch (SeninHatan ex2) {
            System.out.println("Hata Olustu-2:" + ex2);
        }
    }
}
```

Konular

- İstisnalar
- İstisna Oluşumu
- İstisna Yakalama
- İstisna İfadeleri
- İstisna Tip Hiyerarşisi
- **`RuntimeException`** İstisna Tipleri
- İstisna Mesajları
- Yeni İstisna Oluşturmak
- **`finally`** Bloğu
- **`finally`** Bloğu ve **`return`**
- **`finally`** Bloğu ve **`System.exit()`**

finally Bloğu

- **Bir işlemin her koşulda** (istisna olsun ya da olmasın) kesinlikle **yapılması isteniyorsa finally bloğu kullanılır.**

```
try {
        // riskli kod
        // bu kod BenimHatam, SeninHatan
        // OnunHatasi, BizimHatamiz
        //  tipinde istisnalar firlatabilir
    } catch(BenimHatam bh) {
      //  BenimHatam olusursa buraya
    } catch(SeninHatan sh) {
      //  SeninHatan olusursa buraya
    } catch(OnunHatasi oh) {
      //  OnunHatasi olusursa buraya
    } catch(BizimHatamiz bizh) {
      //  BizimHatamiz olusursa buraya
    } finally {
      // ne olursa olsun calisacak kod buraya
}
```

33

finally Bloğu

Örnek

```
package Istisnalar;

class BeklenmeyenHata1 extends Exception {
    public BeklenmeyenHata1(String ekAciklama) {
        super(ekAciklama);
    }
}

class BeklenmeyenHata2 extends Exception {
    public BeklenmeyenHata2(String ekAciklama) {
        super(ekAciklama);
    }
}

public class Oda2 {
    public void isiklariKapat() {
        System.out.println("isiklar kapatildi");
    }

    public void isiklariAc() {
        System.out.println("isiklar acildi");
    }
```

34

`finally` Bloğu

Örnek

```
    public void aramaYap() throws BeklenmeyenHata1, BeklenmeyenHata2 {
        // istisna firlatabilecek olan govde
    }

    public void basla() {
        try {
            // riskli kod
            isiklariAc();
            aramaYap();
        } catch (BeklenmeyenHata1 bh1) {
            System.out.println("BeklenmeyenHata1 yakalandi");
        } catch (BeklenmeyenHata2 bh2) {
            System.out.println("isiklar acildi");
        } finally {
            isiklariKapat(); // dikkat
        }
    }

    public static void main(String args[]) {
        Oda2 o2 = new Oda2();
        o2.basla();
    }
}
```

```
isiklar acildi
isiklar kapatildi
```

35

Konular

- İstisnalar
- İstisna Oluşumu
- İstisna Yakalama
- İstisna İfadeleri
- İstisna Tip Hiyerarşisi
- **`RuntimeException`** İstisna Tipleri
- İstisna Mesajları
- Yeni İstisna Oluşturmak
- **`finally`** Bloğu
- **`finally`** Bloğu ve **`return`**
- **`finally`** Bloğu ve **`System.exit()`**

`finally` Bloğu ve `return`

- Bir yordamdan çıkış için **return kullanılırsa**, **finally bloğu yordamdan çıkmadan önce çalıştırılır.**

```
public class ReturnOrnek {
    public void calis(int deger) {
        try {
            System.out.println("calis yordamı cagrildi, gelen deger: " + deger);
            if (deger == 0) {
                return; // yordamı sessizce terk et
            }
            System.out.println("-- calis yordamı normal bir sekilde bitti--");
        } catch (Exception ex) {
            System.out.println("catch blogu icerisinde");
        } finally {
            System.out.println("finally blogu cagrildi");
            System.out.println("----------------------");
        }
    }

    public static void main(String args[]) {
        ReturnOrnek ro = new ReturnOrnek();
        ro.calis(1);
        ro.calis(0); // dikkat
    }
}
```

```
calis yordamı cagrildi, gelen deger: 1
-- calis yordamı normal bir sekilde bitti--
finally blogu cagrildi
----------------------
calis yordamı cagrildi, gelen deger: 0
finally blogu cagrildi
----------------------
```

37

Konular

- İstisnalar
- İstisna Oluşumu
- İstisna Yakalama
- İstisna İfadeleri
- İstisna Tip Hiyerarşisi
- **`RuntimeException`** İstisna Tipleri
- İstisna Mesajları
- Yeni İstisna Oluşturmak
- **`finally`** Bloğu
- **`finally`** Bloğu ve **`return`**
- **`finally`** Bloğu ve **`System.exit()`**

`finally` Bloğu ve `System.exit()`

- **System.exit() yordamı çağrılırsa**, JVM kapatılır ve **finally bloğuna hiç girilmez.**

```
public class SystemExitOrnek {
    public void calis(int deger) {
        try {
            System.out.println("calis yordamı cagrildi, gelen deger: " + deger);
            if (deger == 0) {
                System.exit(-1); // JVM'i kapat
            }
            System.out.println("-- calis yordamı normal bir sekilde bitti--");
        } catch (Exception ex) {
            System.out.println("catch blogu icerisinde");
        } finally {
            System.out.println("finally blogu cagrildi");
            System.out.println("----------------------");
        }
    }

    public static void main(String args[]) {
        SystemExitOrnek seo = new SystemExitOrnek();
        seo.calis(1);
        seo.calis(0); // dikkat
    }
}
```

```
calis yordamı cagrildi, gelen deger: 1
-- calis yordamı normal bir sekilde bitti--
finally blogu cagrildi
----------------------
calis yordamı cagrildi, gelen deger: 0
```

39