JAVA EXCEPTION HANDLING

İSTİSNA YÖNETİMİ

Try-Catch, Throw, Throws, Custom Exceptions

GİRİŞ: EXCEPTION NEDİR?

Exception (İstisna) Nedir?

Exception = Program çalışırken oluşan beklenmeyen hatalar

Gerçek hayattan örnekler:

- Bir dosyayı açmaya çalışıyorsunuz → Dosya yok!
- Sayıları bölüyorsunuz → Sıfıra bölme hatası!
- Kullanıcıdan sayı bekliyorsunuz → Yazı girdi!
- Dizinin 10. elemanını almak istiyorsunuz → Dizide sadece 5 eleman var!

Exception Handling = Bu hataları yakalayıp, programın çökmesini engellemek

1. EXCEPTION OLMADAN NE OLUR?

Program Çöker!

Kod Örneği:

```
public class HatasizKod {
    public static void main(String[] args) {

        System.out.println("Program başladı");

        int sayi1 = 10;
        int sayi2 = 0;

        // Sıfıra bölme - Program ÇÖKER!
        int sonuc = sayi1 / sayi2;

        System.out.println("Sonuç: " + sonuc);
        System.out.println("Program bitti");  // Buraya asla gelemez
    }
}
```

Çıktı:

```
Program başladı
Exception in thread "main" java.lang.ArithmeticException: / by zero
    at HatasizKod.main(HatasizKod.java:9)
```

Sorun: Program 9. satırda çöktü ve devam edemedi!

2. TRY-CATCH: HATAYI YAKALAMA

2.1 Try-Catch Nedir?

try: "Şunu dene, hata olabilir"

catch: "Hata olursa, bunu yap"

Kod Örneği:

```
public class TryCatchOrnek {
    public static void main(String[] args) {

        System.out.println("Program başladı");

        try {
            // HATA OLABİLECEK KOD BURAYA
            int sayi1 = 10;
            int sayi2 = 0;
            int sonuc = sayi1 / sayi2;  // Hata olacak!
            System.out.println("Sonuç: " + sonuc);

        } catch (ArithmeticException e) {
            // HATA OLURSA BURASI ÇALIŞIR
            System.out.println("❌ HATA: Sıfıra bölme yapılamaz!");
            System.out.println("Hata mesajı: " + e.getMessage());
        }

        System.out.println("Program devam ediyor...");
        System.out.println("Program bitti ✓");
    }
}
```

Çıktı:

```
Program başladı
❌ HATA: Sıfıra bölme yapılamaz!
Hata mesajı: / by zero
Program devam ediyor...
Program bitti ✓
```

Fark: Program çökmedi, devam etti!

3. YAYGIN EXCEPTION TÜRLERİ

3.1 ArithmeticException - Matematiksel Hatalar

Kod Örneği:

```
public class ArithmeticExceptionOrnek {
    public static void main(String[] args) {

        try {
            int sonuc = 10 / 0;  // Sıfıra bölme

        } catch (ArithmeticException e) {
            System.out.println("Matematiksel hata: " + e.getMessage());
        }

        System.out.println("Program devam ediyor");
    }
}
```

3.2 NullPointerException - Null Hatası

En yaygın hatalardan biri!

Kod Örneği:

```
public class NullPointerOrnek {
    public static void main(String[] args) {

        String isim = null;  // Boş referans

        try {
            // Null üzerinde metod çağırıyoruz - HATA!
            int uzunluk = isim.length();

        } catch (NullPointerException e) {
            System.out.println("❌ HATA: Değişken null!");
            System.out.println("İsim değeri kontrol edilmeli");
        }

        System.out.println("Program devam ediyor");
    }
}
```

3.3 ArrayIndexOutOfBoundsException - Dizi Hatası

Kod Örneği:

```
public class ArrayIndexOrnek {
    public static void main(String[] args) {

        int[] sayilar = {10, 20, 30};  // 3 eleman (index: 0,1,2)
```

```
        try {
            // 5. index yok! (sadece 0, 1, 2 var)
            int sayi = sayilar[5];
            System.out.println(sayi);

        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("❌ HATA: Dizide bu index yok!");
            System.out.println("Dizinin boyutu: " + sayilar.length);
            System.out.println("İstenilen index: 5");
        }

        System.out.println("Program devam ediyor");
    }
}
```

3.4 NumberFormatException - Sayı Dönüşüm Hatası

Kod Örneği:

```
public class NumberFormatOrnek {
    public static void main(String[] args) {

        String metin = "abc123";  // Sayı değil!

        try {
            // String'i sayıya çevirmeye çalışıyoruz
            int sayi = Integer.parseInt(metin);
            System.out.println("Sayı: " + sayi);

        } catch (NumberFormatException e) {
            System.out.println("❌ HATA: Bu bir sayı değil!");
            System.out.println("Girilen değer: " + metin);
            System.out.println("Sadece rakam girilmeli!");
        }

        System.out.println("Program devam ediyor");
    }
}
```

4. ÇOKLU CATCH BLOKLARI

4.1 Farklı Hataları Farklı Şekilde Yakalamak

Kod Örneği:

```
public class CokluCatch {
    public static void main(String[] args) {

        try {
            // Farklı hatalar oluşabilir
            String metin = null;
            int uzunluk = metin.length();  // NullPointerException

            int[] sayilar = {1, 2, 3};
            int sayi = sayilar[10];  // ArrayIndexOutOfBoundsException

            int sonuc = 10 / 0;  // ArithmeticException

        } catch (NullPointerException e) {
            System.out.println("❌ Null hatası yakalandı!");

        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("❌ Dizi index hatası yakalandı!");

        } catch (ArithmeticException e) {
            System.out.println("❌ Matematiksel hata yakalandı!");

        } catch (Exception e) {
            // Tüm diğer hataları yakalar
            System.out.println("❌ Bilinmeyen hata yakalandı!");
        }

        System.out.println("Program devam ediyor");
    }
}
```

ÖNEMLİ: Catch Sırası

Catch blokları ÖZELDEN GENELE sıralanmalı!

```
// YANLIŞ SIRA! ❌
try {
    // kod
} catch (Exception e) {          // En genel - önce
    // ...
} catch (ArithmeticException e) { // Özel - sonra (ULAŞILAMAZ!)
    // ...
}

// DOĞRU SIRA! ✅
try {
```

```
    // kod
} catch (ArithmeticException e) { // Özel - önce
    // ...
} catch (Exception e) {          // Genel - sonra
    // ...
}
```

5. FINALLY BLOĞU

5.1 Finally Nedir?

finally = Hata olsa da olmasa da MUTLAKA çalışır

Kullanım Alanları:

- Dosya kapatma
- Veritabanı bağlantısı kapatma
- Kaynakları serbest bırakma

Kod Örneği:

```
public class FinallyOrnek {
    public static void main(String[] args) {

        System.out.println("1. Program başladı");

        try {
            System.out.println("2. Try bloğu çalışıyor");
            int sonuc = 10 / 0;  // Hata!
            System.out.println("3. Bu satır çalışmaz");

        } catch (ArithmeticException e) {
            System.out.println("4. Catch bloğu çalışıyor");
            System.out.println("   Hata: " + e.getMessage());

        } finally {
            System.out.println("5. Finally bloğu çalışıyor");
            System.out.println("   (Her durumda çalışır!)");
        }

        System.out.println("6. Program bitti");
    }
}
```

Çıktı:

```
1. Program başladı
2. Try bloğu çalışıyor
4. Catch bloğu çalışıyor
   Hata: / by zero
5. Finally bloğu çalışıyor
   (Her durumda çalışır!)
6. Program bitti
```

6. THROW - HATA FIRLATMA

6.1 Throw Nedir?

throw = "Kendin hata fırlat"

Kod Örneği:

```
public class ThrowOrnek {

    // Yaş kontrolü yapan metod
    public static void yasKontrol(int yas) {

        if (yas < 18) {
            // Kendi hatamızı fırlatıyoruz
            throw new IllegalArgumentException(
                "Yaş 18'den küçük olamaz! Girilen: " + yas
            );
        }

        System.out.println("✓ Yaş uygun: " + yas);
    }

    public static void main(String[] args) {

        try {
            yasKontrol(20);  // Uygun - sorun yok
            yasKontrol(15);  // Uygun değil - hata fırlatılır!

        } catch (IllegalArgumentException e) {
            System.out.println("❌ HATA: " + e.getMessage());
        }
    }
}
```

6.2 Banka Hesabı Örneği

```
public class BankaHesabi {
    private double bakiye;

    public BankaHesabi(double ilkBakiye) {
        this.bakiye = ilkBakiye;
    }

    public void paraCek(double miktar) {

        // Miktar kontrolü
        if (miktar <= 0) {
            throw new IllegalArgumentException(
                "Çekilecek miktar pozitif olmalı!"
            );
```

```
        }

        // Bakiye kontrolü
        if (miktar > bakiye) {
            throw new IllegalArgumentException(
                "Yetersiz bakiye! Bakiye: " + bakiye +
                ", İstenilen: " + miktar
            );
        }

        // Her şey yolunda - para çek
        bakiye -= miktar;
        System.out.println("✓ " + miktar + " TL çekildi");
        System.out.println("  Kalan: " + bakiye + " TL");
    }
}
```

7. THROWS - HATAYI BİLDİRME

7.1 Throws Nedir?

throws = "Bu metod hata fırlatabilir, sen halledeceksin"

Kod Örneği:

```
import java.io.*;

public class ThrowsOrnek {

    // Bu metod IOException fırlatabilir
    // throws ile bildiriyoruz
    public static void dosyaOku(String dosyaAdi)
            throws IOException {

        FileReader dosya = new FileReader(dosyaAdi);
        System.out.println("Dosya okunuyor...");
        dosya.close();
    }

    public static void main(String[] args) {

        // dosyaOku() throws kullanıyor
        // Biz try-catch ile yakalıyoruz
        try {
            dosyaOku("test.txt");

        } catch (IOException e) {
            System.out.println("❌ Dosya hatası: " + e.getMessage());
        }
    }
}
```

7.2 Throw vs Throws Farkı

- throw: Metod içinde kullanılır, hata fırlatır
- throws: Metod imzasında kullanılır, hata bildirir

```
public class ThrowVsThrows {

    // THROWS - Metod imzasında
    public static void metod1() throws IOException {
        // Bu metod IOException fırlatabilir
    }

    // THROW - Metod içinde
    public static void metod2(int sayi) {
        if (sayi < 0) {
            throw new IllegalArgumentException("Negatif!");
        }
```

```
    }
}
```

8. CUSTOM EXCEPTION - ÖZEL HATA

8.1 Kendi Exception Sınıfınızı Oluşturma

Kod Örneği:

```
// Özel exception sınıfı
class YetersizBakiyeException extends Exception {

    private double bakiye;
    private double istenilenMiktar;

    public YetersizBakiyeException(double bakiye, double istenilen) {
        super("Yetersiz bakiye!");
        this.bakiye = bakiye;
        this.istenilenMiktar = istenilen;
    }

    @Override
    public String getMessage() {
        return String.format(
            "Yetersiz bakiye! Bakiye: %.2f TL, " +
            "İstenilen: %.2f TL, Eksik: %.2f TL",
            bakiye, istenilenMiktar, (istenilenMiktar - bakiye)
        );
    }
}

// Kullanım
class BankaHesabi2 {
    private double bakiye;

    public BankaHesabi2(double ilkBakiye) {
        this.bakiye = ilkBakiye;
    }

    public void paraCek(double miktar)
            throws YetersizBakiyeException {

        if (miktar > bakiye) {
            throw new YetersizBakiyeException(bakiye, miktar);
        }

        bakiye -= miktar;
        System.out.println("✓ Para çekildi. Kalan: " + bakiye);
    }
}
```

9. CHECKED vs UNCHECKED EXCEPTIONS

9.1 Exception Hiyerarşisi

```
Throwable
├── Error (Sistem hataları - yakalanamaz)
│   └── OutOfMemoryError
│
└── Exception
    ├── Checked Exceptions (Yakalanmalı)
    │   ├── IOException
    │   ├── SQLException
    │   └── FileNotFoundException
    │
    └── RuntimeException (Unchecked)
        ├── NullPointerException
        ├── ArithmeticException
        ├── ArrayIndexOutOfBoundsException
        └── NumberFormatException
```

9.2 Checked Exception - Yakalanması Zorunlu

```
import java.io.*;

public class CheckedOrnek {

    // Checked exception - try-catch VEYA throws ZORUNLU
    public static void dosyaOku() throws IOException {
        FileReader dosya = new FileReader("test.txt");
        // ...
    }

    // VEYA
    public static void dosyaOku2() {
        try {
            FileReader dosya = new FileReader("test.txt");
            // ...
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

9.3 Unchecked Exception - Yakalanması İsteğe Bağlı

```
public class UncheckedOrnek {

    // Unchecked - try-catch ZORUNLU DEĞİL
    public static void bolme(int a, int b) {
        int sonuc = a / b;  // ArithmeticException olabilir
        System.out.println(sonuc);
```

```
    }

    // Ama yine de yakalayabilirsiniz
    public static void bolmeGuvenli(int a, int b) {
        try {
            int sonuc = a / b;
            System.out.println(sonuc);
        } catch (ArithmeticException e) {
            System.out.println("Sıfıra bölme hatası!");
        }
    }
}
```

10. BEST PRACTICES - EN İYİ UYGULAMALAR

10.1 Doğru Kullanım

1. Özel Exception Yakalayın

```
// ✅ İYİ
try {
    int sonuc = 10 / 0;
} catch (ArithmeticException e) {  // Spesifik
    System.out.println("Matematik hatası");
}

// ❌ KÖTÜ
try {
    int sonuc = 10 / 0;
} catch (Exception e) {  // Çok genel!
    System.out.println("Hata");
}
```

2. Anlamlı Hata Mesajları

```
// ✅ İYİ
throw new IllegalArgumentException(
    "Yaş 0 ile 150 arasında olmalı. Girilen: " + yas
);

// ❌ KÖTÜ
throw new IllegalArgumentException("Hata");
```

3. Exception'ı Loglayın

```
// ✅ İYİ
try {
    // kod
} catch (Exception e) {
    System.err.println("Hata: " + e.getMessage());
    e.printStackTrace();  // Tam detay
}
```

10.2 Yanlış Kullanım

1. Boş Catch Bloğu (ÇOK KÖTÜ!)

```
// ❌ ÇOK KÖTÜ!
try {
    // kod
} catch (Exception e) {
    // Hiçbir şey yapma - HATA GİZLENİR!
}
```

2. Exception'ı Yutma

```
// ❌ KÖTÜ
```

```
try {
    // kod
} catch (Exception e) {
    System.out.println("Hata oldu");
    // e.printStackTrace() YOK - detay yok!
}
```

11. TRY-WITH-RESOURCES (Modern Java)

Java 7+ - Otomatik Kaynak Yönetimi

```
// ESKİ YOL (Java 6 ve öncesi)
FileReader dosya = null;
try {
    dosya = new FileReader("test.txt");
    // işlemler
} catch (IOException e) {
    e.printStackTrace();
} finally {
    // MANUEL KAPATMA
    if (dosya != null) {
        try {
            dosya.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

// YENİ YOL (Java 7+)
// Try'ın içinde kaynak aç - OTOMATİK KAPANIR!
try (FileReader dosya = new FileReader("test.txt")) {

    // işlemler

} catch (IOException e) {
    e.printStackTrace();
}
// Dosya otomatik kapandı!
```

12. GERÇEK HAYAT ÖRNEĞİ

Kullanıcı Girişi Validasyonu

```
import java.util.Scanner;

public class KullaniciGirisi {

    public static int sayiAl(Scanner scanner, String mesaj) {
        while (true) {
            try {
                System.out.print(mesaj);
                String girdi = scanner.nextLine();

                // Boş kontrol
                if (girdi.trim().isEmpty()) {
                    throw new IllegalArgumentException(
                        "Boş bırakılamaz!"
                    );
                }

                // Sayıya çevir
                int sayi = Integer.parseInt(girdi);

                // Negatif kontrol
                if (sayi < 0) {
                    throw new IllegalArgumentException(
                        "Negatif olamaz!"
                    );
                }

                return sayi;  // Başarılı!

            } catch (NumberFormatException e) {
                System.out.println(
                    "❌ Hata: Lütfen sadece rakam girin!"
                );

            } catch (IllegalArgumentException e) {
                System.out.println("❌ Hata: " + e.getMessage());
            }
        }
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("=== HESAP MAKİNESİ ===");

        int sayi1 = sayiAl(scanner, "Birinci sayı: ");
```

```
        int sayi2 = sayiAl(scanner, "İkinci sayı: ");

        System.out.println("Toplam: " + (sayi1 + sayi2));
        System.out.println("Fark: " + (sayi1 - sayi2));

        try {
            System.out.println("Bölüm: " + (sayi1 / sayi2));
        } catch (ArithmeticException e) {
            System.out.println("Bölüm: Sıfıra bölünemez!");
        }

        scanner.close();
    }
}
```

SONUÇ

Bu dokümanda Java Exception Handling'in tüm temel konularını detaylı olarak inceledik:

- **Try-Catch-Finally:** Hataları yakalama ve işleme
- **Yaygın Exception Türleri:** Null, Arithmetic, Array, NumberFormat
- **Throw ve Throws:** Hata fırlatma ve bildirme
- **Custom Exceptions:** Özel hata sınıfları oluşturma

Exception Handling, güvenilir ve dayanıklı programlar yazmanın temelidir. Hataları doğru bir şekilde yönetmek, kullanıcı deneyimini iyileştirir ve programınızın beklenmedik durumlarda çökmesini engeller.

ÖNEMLİ HATIRLATMALAR

- Her zaman spesifik exception yakalayın
- Boş catch bloğu kullanmayın
- Exception'ları loglayın
- Finally ile kaynakları temizleyin
- Anlamlı hata mesajları yazın