Java Exceptions

harezmi bilişim çözümleri

Java'da Exception ve Hata Yönetimi

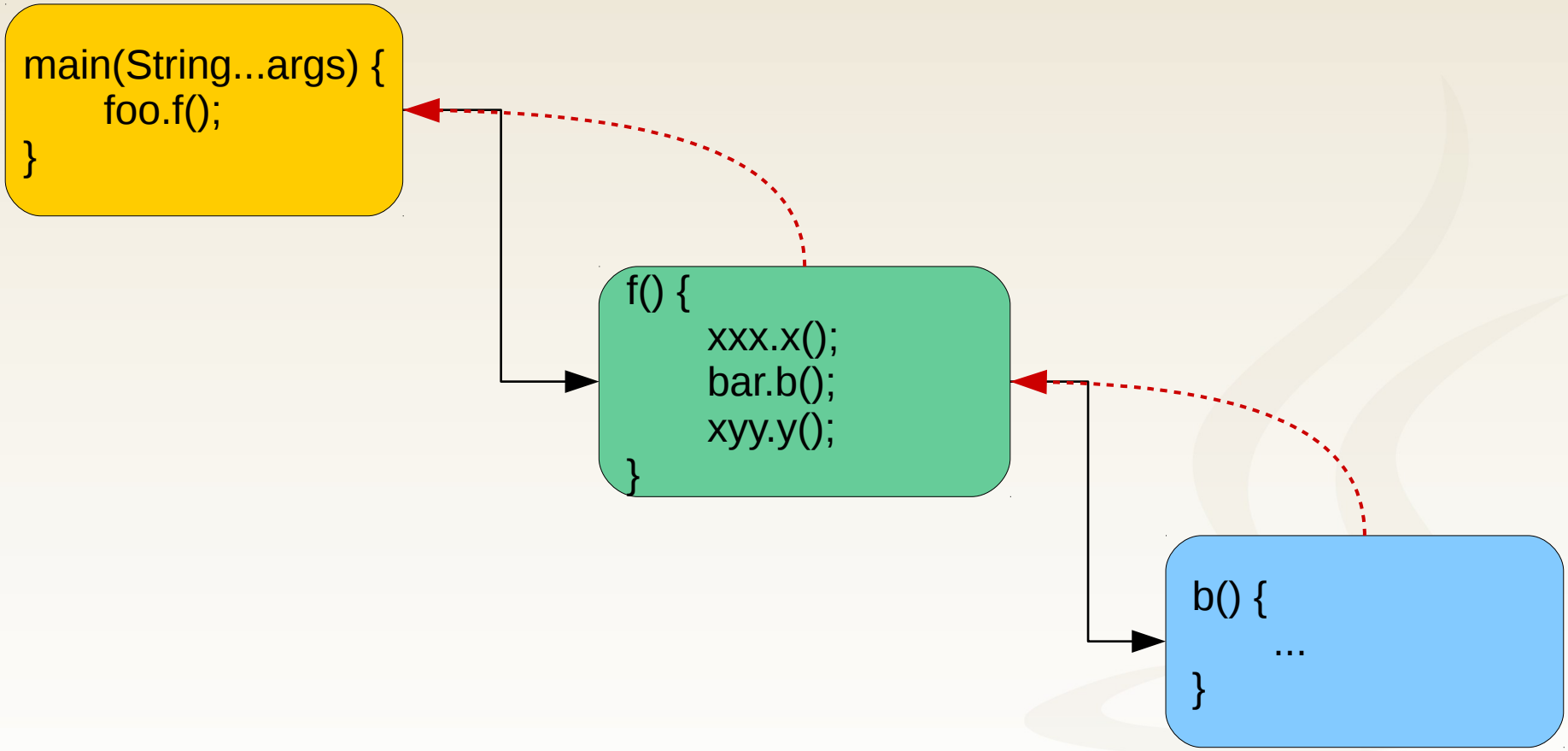
JAVA Eğitimleri

- Uygulamalarda **ana akışın** yanı sıra, **hatalı durumların** da düşünülüp ele alınması gerekir
- Java'da **hatalı durumları ana akıştan ayrı ele almak** için exception mekanizması mevcuttur
- Hatalı durum söz konusu olduğu vakit ana akış **exception fırlatılarak** yarıda kesilir
- Fırlatılan bu exception uygulamanın başka bir yerinde **yakalanarak** işlem görür

Java'da Exception ve Hata Yönetimi

- Exception sayesinde programcılar normal akış üzerinde **sürekli olarak hata kontrolü yapmak zorunda kalmazlar**
- Exception'lar sayesinde hata yakalama işlemi **normal akıştan bağımsız** hale gelmektedir

Exception ile Hata Yönetimi




```
Exception in thread "main" java.lang.RuntimeException: something went wrong here!
        at com.example.demo.Bar.b(Bar.java:5)
        at com.example.demo.Foo.f(Foo.java:58)
        at com.example.demo.Main.main(Main.java:6)
```

Exception'lar ile Çalışmak

JAVA Eğitimleri

- Exception'lar da **normal Java sınıfları**dır
- Genel olarak **java.lang.Exception** veya **java.lang.RuntimeException** sınıflarından türerler
- **Exception nesnesi** de ilgili exception sınıfından yaratılır
- Exception (nesnesi) hatanın olduğu noktadan fırlatılır (**throw**)
- Hatayı ele alabilecek uygun noktada ise fırlatılan Exception (nesnesi) yakalanır (**catch**)

Exception Tanımı

```
public class GecersizHizDegeriException extends Exception {
}


public class HizLimitiException extends RuntimeException {
    public HizLimitiException() {
    }

    public HizLimitiException(String msg, Throwable cause) {
        super(msg, cause);
    }

    public HizLimitiException(String msg) {
        super(msg);
    }

    public HizLimitiException(Throwable cause) {
        super(cause);
    }
}
```

Exception Fırlatmak

```
public class Motor {
   private int hiz;

   public Integer getHiz() {
      return new Integer(hiz);
   }

   public void setHiz(Integer hiz) throws
GecersizHizDegeriException, HizLimitiException {
      if(hiz > 200) {
         throw new HizLimitiException("Hız limiti aşıldı");
      } else if(hiz < 0) {
         throw new GecersizHizDegeriException();
      }

      this.hiz = hiz.intValue();
   }
}
```

Exception'ın Yakalanması

```
public class Honda extends Araba {

    @Override
    public Integer hizlan() {
        Integer hiz = motor.getHiz() + 1;
        try {
            motor.setHiz(hiz);
            return hiz;
        } catch(GecersizHizDegeriException ex) {
            throw new RuntimeException("Geçersiz hız
değeri", ex);
        } finally {
            System.out.println("Hızlan metodu çalıştı");
        }
    }

}
```

Try bloğu içerisinde bir yerde Exception fırlatıldığında devreye girer

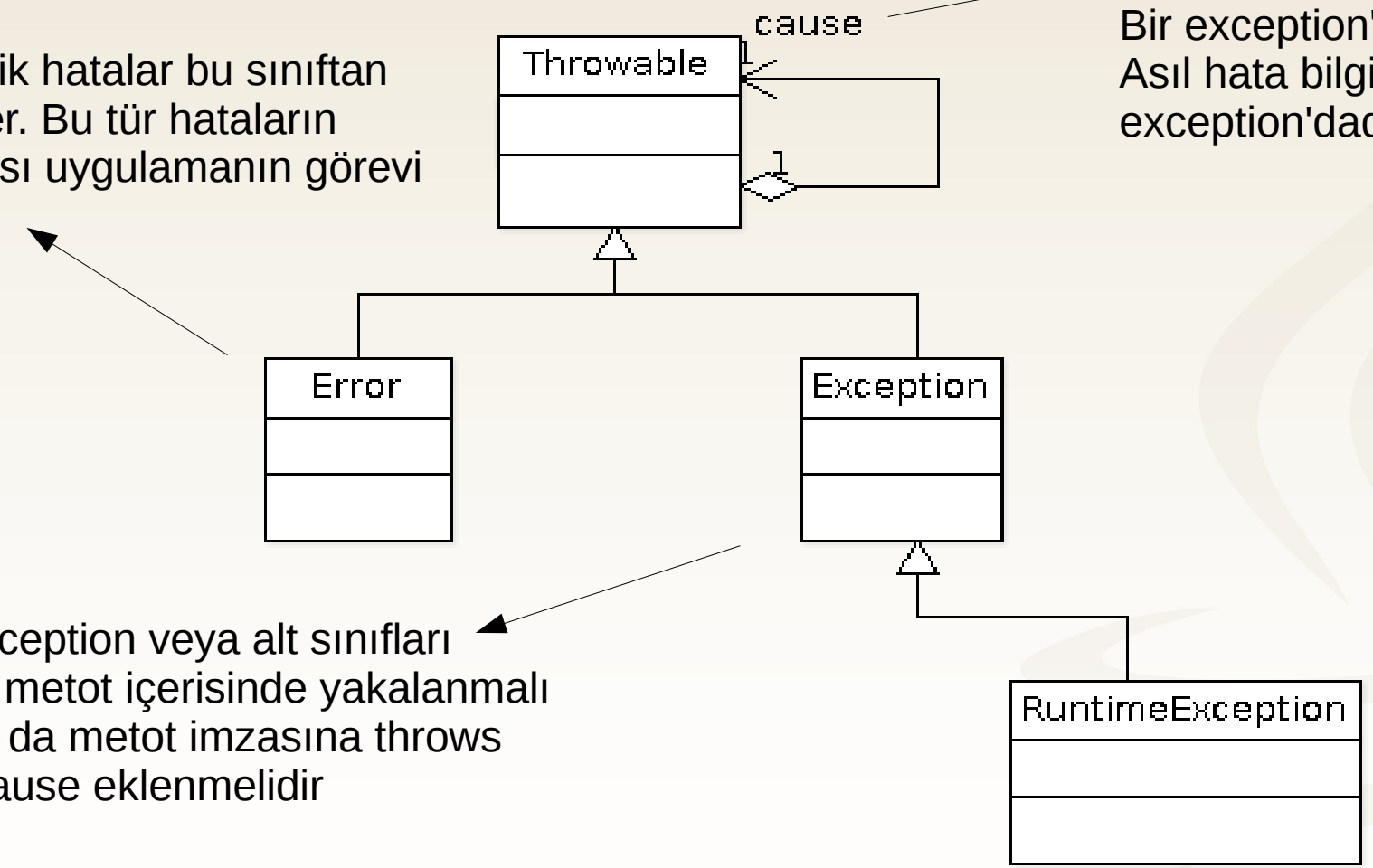
Cause exception

Try bloğunda exception olsa da olmasada çalışır

Exception'ın Yakalanması

- Fırlatılan exception'ların uygulama içerisinde **bir noktada yakalanmaları** gerekir
- Exception herhangi bir yerde **yakalanmaz ise uygulamanın terminate olması**na sebep olur

Exception Sınıf Hiyerarşisi



JVM spesifik hatalar bu sınıftan
Extend eder. Bu tür hataların
Yakalanması uygulamanın görevi
değildir

Bir exception başka
Bir exception'ı wrap edebilir
Asıl hata bilgisi cause
exception'dadır

Exception veya alt sınıfları
ya metot içerisinde yakalanmalı
Ya da metot imzasına throws
Clause eklenmelidir

RuntimeException ve alt sınıflarının metot içerisinde yakalanması veya metot İmzasına throws clause eklenmesi şart değildir

java.lang.Exception

- **Uygulama exception'ları** da denir
- Java'yı tasarlayanlar, uygulamaya özel exception sınıflarının bu sınıftan türetilmesini arzu etmişlerdir
- Fırlatılan **java.langException** nesneleri metot içerisinde yakalanmaz ise, **metot imzasına** bu exception'ların her birisinin tanımlanması gerekir
- Bu zaruret programcıların kötü kod yazmasına neden olmaktadır

java.lang.RuntimeException

- **Sistem exception'ları**nın da bu sınıftan türemesi düşünülmüştür
- Bu tür exceptionların **yakalanma veya deklere edilme zorunluluğu yok**tur
- Uygulama içerisinde nerede yakalanması uygun ise orada yakalanabilmektedir
- Uygulama içerisindeki bütün exception sınıflarının **RuntimeException**'dan türetilmesi iyi bir pratik olarak görülmektedir

Exception'ın Yeniden Fırlatılması

- Bir metot içerisinde yakalanan exception **farklı bir exception ile wrap edilerek** yeniden fırlatılabilir
- Ya da yakalanan exception **aynen yeniden** fırlatılabilir

```
try {
    motor.setHiz(hiz);
    return hiz;
} catch(GecersizHizDegeriException ex) {
    throw new RuntimeException("Geçersiz hız değeri", ex);
}
```

Cause exception

Exception'ın Yeniden Fırlatılması

```
public void doSomeWork() throws NumberFormatException,
                                ArrayIndexOutOfBoundsException {
    try {
        //...
    } catch(Exception ex) {
        ex.printStackTrace();
        throw ex;
    }
}
```

Try bloğunda yapılan işlemlerden fırlatılan herhangi bir Exception catch bloğunda yakalanarak Exception tipi ile tekrar rethrow edilebilir

Java 7 öncesinde metodun throws clause'unda belirtilen Exception tiplerinden birisine denk gelen exception tipi ile rethrow etmek gerekirdi, ancak Java 7 ile bu zorunluluk ortadan kalktı

Exception Zinciri

- Throwable sınıfı içerisinde **cause değişkeni** vardır
- Bu da bir **Throwable**'dır
- Bu durumda bir exception başka bir **exception'ı wrap** edebilmektedir
- Bu **recursive** biçimde birkaç defa tekrar edebilir
- Asıl hata **exception zincirinin en son noktası**ndadır

Exception Zinciri ve Stacktrace

```
Exception in thread "main"
java.lang.RuntimeException: Hız limiti hatası
at com.javaegitimleri.Honda.hizlan(Honda.java:40)
at com.javaegitimleri.Formula1.main(Formula1.java:18)
Caused by: com.javaegitimleri.HizLimitiException:
Hız limiti aşıldı
at com.javaegitimleri.Motor.setHiz(Motor.java:12)
at com.javaegitimleri.Honda.hizlan(Honda.java:35)
... 1 more
```

Catch Exception Hiyerarşisi

```
@Override
public Integer hizlan() {
    Integer hiz = motor.getHiz() + 1;
    try {
        motor.setHiz(hiz);
        return hiz;
    } catch(GecersizHizDegeriException ex) {
        throw new RuntimeException("Geçersiz hız değeri", ex);

    } catch(HizLimitiException ex) {
        throw new RuntimeException("Hız limiti hatası",ex);

    } catch(Exception ex) {
        System.out.println("Genel bir hata oldu");
        return 0;
    } finally {
        System.out.println("Hizlan metodu çalıştı");
    }
}
```

Çoklu Catch (Multi-Catch) Bloğu

- **Java 7** ile birlikte gelen bir kabiliyettir
- Birden fazla exception'ı ayrı ayrı catch blokları ile yakalamak yerine **tek bir catch bloğu** ile yakalamayı sağlar
- Önemli nokta bu exception'ların birbirleri arasında **hiyerarşik olarak herhangi bir ilişki** olmamasıdır
- Başka bir ifade ile **birbirlerinden türeyen exception'lar** multi-catch blok içerisinde yakalanamaz

Çoklu Catch Bloğu

```
@Override
public Integer hizlan() {
   Integer hiz = motor.getHiz() + 1;
   try {
      motor.setHiz(hiz);
      return hiz;
   } catch( GecersizHizDegeriException |
            HizLimitiException ex) {
      throw new RuntimeException("Hızlanmada hata", ex);
   } finally {
      System.out.println("Hizlan metodu çalıştı");
   }
}
```

Pipe operatörü ile yakalanacak exception'lar sıralanarak yazılır

Çoklu Catch Bloğu

```
@Override
public Integer hizlan() {
   Integer hiz = motor.getHiz() + 1;
   try {
      motor.setHiz(hiz);
      return hiz;
   } catch( GecersizHizDegeriException |
            Exception ex) {
      throw new RuntimeException("Hızlanmada hata", ex);
   } finally {
      System.out.println("Hizlan metodu çalıştı");
   }
}
```

Aralarında hiyerarşik olarak bağlantı olduğu için bu şekilde bir multi-catch blok kullanımına Java derleyicisi izin vermez

Yakalanmayan Exception’lar

- Herhangi bir catch bloğu tarafından yakalanmayan exception’lar metot call hiyerarşisinin en üstüne kadar çıkar ve **uygulamanın terminate etmesi**ne yol açabilir
- Bu tür exception’ları en uç noktada yakalamak için **UncaughtExceptionHandler** tanımlayabiliriz

Yakalanmayan Exception'lar

```
public class AppErrorHandler implements UncaughtExceptionHandler {

    @Override
    public void uncaughtException(Thread t, Throwable e) {
        System.out.println("Handled error :" + e.getMessage());
    }

}
```

```
public class Main {
    public static void main(String[] args) throws Exception {
        Thread.currentThread().setUncaughtExceptionHandler(
                                                new AppErrorHandler());
        ...
    }
}
```

veya

```
public class Main {
    public static void main(String[] args) throws Exception {
        Thread.setDefaultUncaughtExceptionHandler(new AppErrorHandler());
        ...
    }
}
```

Try With Resources Kabiliyeti

JAVA Eğitimleri

- Java 7 ile gelen diğer bir yenilik ise **try-with-resources** kabiliyetidir
- Uygulama içerisinde işi bittikten sonra **programcı tarafından** kapatılması gereken SQL **Connection**, File **InputStream** gibi resource'lar try bloğu sonlandığında otomatik olarak kapatılır
- Bu tür resource'lar **java.lang.AutoCloseable** veya **java.io.Closeable** arayüzlerinden birisini implement etmelidir

Try With Resources Kabiliyeti

```
try(FileInputStream fin =
        new FileInputStream("/input.txt")) {
    int i = -1;

    while((i = fin.read()) != -1) {
        //...
    }

} catch(FileNotFoundException ex) {

    System.out.println("/input.txt dosyası bulunamadı");

} catch(IOException ex) {

    System.out.println("I/O hatası oldu");

}
```

Try-with-resources ifadesi içerisinden fırlatılabilir

Hem try-with-resources ifadesinden (close), hem de try bloğu içerisinden (read) fırlatılabilir

Try With Resources Kabiliyeti

JAVA Eğitimleri

- Try-with-resources ifadesi de normal try bloğu gibi **catch** ve **finally** bloklarına sahip olabilir
- Catch ve finally blokları try-with-resources ifadesinde açılan **resource kapatıldıktan sonra** çalıştırılacaktır

Suppressed Exceptions

- **Hem try-with-resources** ifadesinden, **hem de try bloğu**nun kendi içinden exception fırlatılabilir
- Böyle bir durumda **sadece try bloğundan fırlatılan** exception yakalanabilir
- Try-with-resources içerisinde fırlatılan exception ise **suppress** edilecektir
- Suppress edilen exception(lar)a **yakalanan exception içerisinden** ulaşılabilir

Suppressed Exceptions

```
try(FileInputStream fin =
        new FileInputStream("/input.txt")) {

    int i = -1;
    while((i = fin.read()) != -1) {
        //...
    }

} catch(FileNotFoundException ex) {
    System.out.println("/input.txt dosyası bulunamadı");
} catch(IOException ex) {

    System.out.println("I/O hatası oldu");

    Throwable[] suppressedExs = ex.getSuppressed();
    for(Throwable t:suppressedExs) {
        System.out.println(t.getMessage());
    }

}
```

Try-with-Resources bloğu

Try bloğunun içerisi

Java Assertions

- Geliştirme ve test safhalarında **hataları ve bug'ları daha kolay tespit edebilmek** için kullanılırlar
- **Assert ifadesi** ile gerçekleştirilirler

```
public void setHiz(Integer hiz)
        throws  GecersizHizDegeriException, HizLimitiException {
    assert hiz > 0 && hiz < 200 : "Hız değeri 0 ile 200
arasında olmalı";

    if(hiz > 200) {
        throw new HizLimitiException("Hız limiti aşıldı");
    } else if(hiz < 0) {
        throw new GecersizHizDegeriException();
    }
    this.hiz = hiz.intValue();
}
```

Assert ifadesi nesnenin durumunu değiştirecek, side-effect üretecek bir ifade olmamalıdır

Java Assertions

- Çalışma zamanında assert ifadesi true değer üretmez ise **AssertionError** fırlatılır
- JVM'de **-ea parametresi** ile etkinleştirilmeleri gerekir
- Sadece geliştirme ortamında aktif olmalıdırlar, **canlı sistemde kesinlikle devre dışı** olmaları gerekir

Assert İfadesinin Kullanım Yerleri

- Assert ifadeleri aşağıdaki durumlarda kullanılabilir
  - **Ön ve son-koşulları**n denetlenmesinde
  - Sınıf içerisindeki **değişmezler**in (class invariant) kontrol edilmesinde
  - Kodun içerisinde **erişilmemesi gereken yerler**in denetlenmesinde (switch ifadesinde default değer gibi)
- Assertion'ların en uygun kullanımı **"class invariant"ların denetlenmesi**dir

İletişim

www.harezmi.com.tr

www.java-egitimleri.com

info@harezmi.com.tr

info@java-egitimleri.com

@HarezmiBilisim

@JavaEgitimleri