IN1010 Våren 2018

Feilsituasjoner og unntak i Java

Stein Gjessing,

Institutt for informatikk,
Universitetet i Oslo

Jeg prøvde å bestille billett med Air France:

<LINK rel="stylesheet" href="/NO/common/common/css/headerfooter.

java.util.NoSuchElementException
at java.util.AbstractList$Itr.next(AbstractList.java:426)
at com.airfrance.pce.resainfovol.infovols.bean.ActuVolsBean.getNextVol(Unknown Source)
at NO.en.local.resainfovol.infovols._0002fNO_jsp_0._jspService (_0005fvol_jsp_0.java:555)
at org.apache.jasper.runtime.HttpJspBase.service(HttpJspBase.java:126)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:853)
at org.apache.jasper.runtime.JspServlet$JspServletWrapper.service(JspServlet.java:174)
at org.apache.jasper.runtime.JspServlet.serviceJspFile(JspServlet.java:274)
at org.apache.jasper.runtime.JspServlet.service(JspServlet.java:387)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:853)
at com.broadvision.servlet.ServletContainer.service(ServletContainer.java:415)
at com.broadvision.servlet.BVRequestDispatcher.forward(BVRequestDispatcher.java:143)
at org.apache.struts.action.RequestProcessor.doForward(RequestProcessor.java:1069)
at com.airfrance.struts.action.AFRequestProcessor.processForwardConfig(AFRequestProcessor.java:146)
at org.apache.struts.action.RequestProcessor.process(RequestProcessor.java:279)
at org.apache.struts.action.ActionServlet.process(ActionServlet.java:1482)
at com.airfrance.struts.action.AFActionServlet.process(Unknown Source)
at org.apache.struts.action.ActionServlet.doPost(ActionServlet.java:525)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:760)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:853)
at com.broadvision.servlet.ServletContainer.service(ServletContainer.java:415)
at com.broadvision.servlet.ServletContainer.processRequest(ServletContainer.java:359)
at com.broadvision.servlet.ServletContextContainer.processRequest(ServletContextContainer.java:825)
at com.broadvision.servlet.BVServletEngine.service(BVServletEngine.java:255)
at com.broadvision.servlet.BVServletConnector.service(BVServletConnector.java:151)

Og jeg skulle sjekke når T-banen går

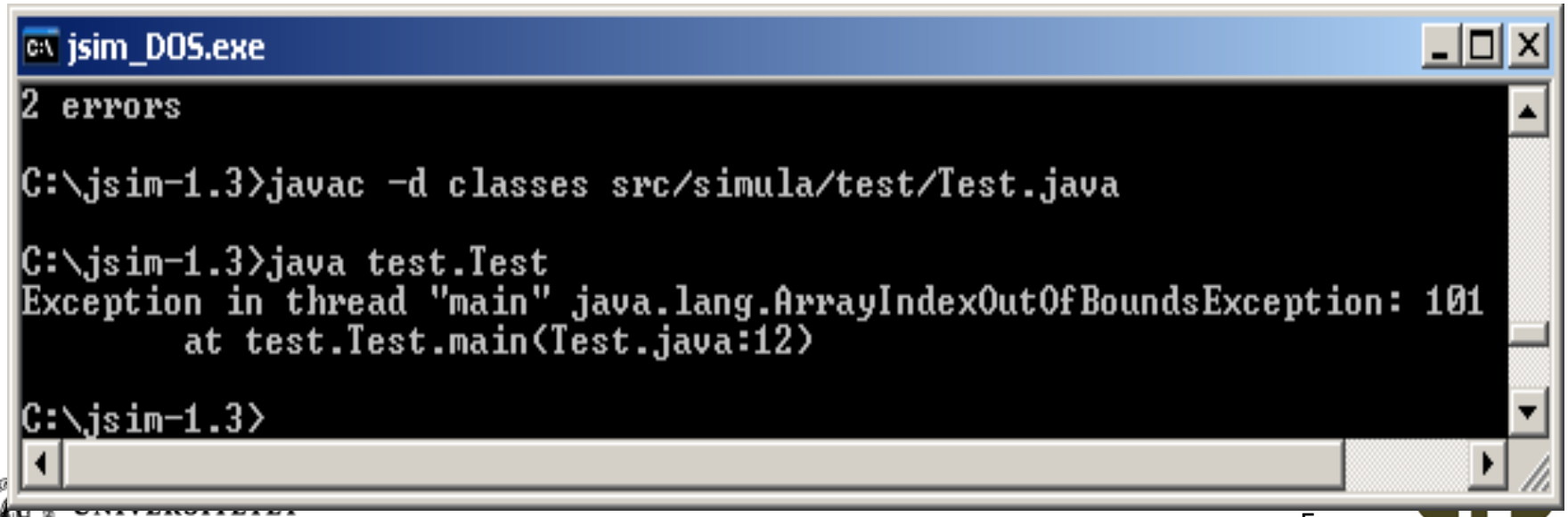
Avganger fra Nationaltheatret [T-bane]							
				Klokka er: 17:15:34			
Linje	**Mot**	**Plf**	**Avgangstid**	**Merknad**			
		()	253:46	Microsoft VBScript runtime error '800a01a8' Object required: 'ObjXML.documentElement' /Common.asp, line 331			

Oversikt

- Hva er en feil (er det ikke mulig å unngå feil ?)
- Hva skjer når et program feiler
- Mål 1: Å ikke få feilmeldinger fra kjøretidsystemet, men isteden la programmet få kontrollen tilbake etter en feilsituasjon:
  - Der programmet normalt ville ha avsluttet med en feilmelding
  - Eks: Divisjon med null, greier ikke åpne en fil, filen finnes ikke, knytte kontakt over nettet mislykkes, utenfor array-grensen, + + +
- Mål 2: Enklere, mer vedlikeholdbar og mer forståelig kode
- Mål 3: Oppdage feil så tidlig som mulig
  - Da er det letter å finne årsaken til feilen

Array indeks utenfor sine grenser

```
int [ ] tallVektor;
tallVektor = new int [100];
tallVektor[101] = 17;
```



Unntak / feil, behandling i Java

- Mye kode kan feile og feilaktige situasjoner (unntak) kan oppstå.
- Kode som kan feile ***kan -*** og som oftest ***må*** - vi legge følgende rundt:

Kode som kan feile skrives her

```
try {
      ... Kode som kan feile ...;
}
catch (Exception e)  {
      .... Gjør noe med feilen , prøv å
           rett opp ...
 }
 finally { Blir alltid utført }
```

Feiler koden blir denne blokken kalt med feilobjektet e som parameter

Må ha minst en av catch og finally

Fem reserverte Java ord

- **try** - Står foran en blokk som er usikker
  dvs. der det kan oppstå et unntak
- **catch** - Står foran en blokk som behandle et unntak.
  Har en peker til et unntaksobjekt som parameter
- **finally** - blir alltid utført

- **throw** - Starter å kaste et unntak – lag et unntaksobjekt
  throw <en peker til et unntaksobjekt>
  f.eks throw new Unntak();
- **throws** - Kaster et unntak videre
  Brukes i overskriften på en metode som ikke selv vil behandle et unntak

- **Bruk:**
  - **try** { <usikker kode>}
    catch (Unntaksklasse u) {
    <behandle unntaket, u peker på et objekt som beskriver unntaket>
    }
    finally { rydd opp }

Unntaksbehandling – egen unntaksklasse

```
try {
   <USIKKER KODE>
         <Hvis det skjer noe galt:>
         throw new Unntaksklasse( );
 . . . .
}
catch (Unntaksklasse  unt) {
  < Unntaksbehandling.
     Dette hoppes over  når intet
     unormalt/galt har hendt >
} finally { hit kommer  programmet alltid,
          også om unntaket ikke ble fanget }

< her fortsetter programmet
  både etter normal utføring og etter
  behandling av eventuelle unntak ,
  men ikke når et unntak er kastet uten
  at det er fanget (men da må throws med)>
```

Enkleste form for unntaksbehandling.

På forhånd har vi deklarert:

```
class Unntaksklasse
        extends Exception {
  .........
}
```

Når unntak oppstår i en metode og ikke behandles der

A a

```
try { . . . .
      x = b ( );
 . . . .
}
catch (Unntaksklassen  unt) {
  < Unntaksbehandling.
     Dette hoppes over
     når intet unormalt
     har hendt >
}
finally { hit kommer
     programmet alltid }

< her fortsetter programmet
  både etter normal utføring og
  etter behandling av
  eventuelle unntak ,  men
  ikke hvis unntaket  blir kastet
  videre>
```

a kaller b

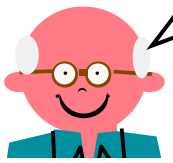
int b( ) **throws** Unntaksklassen {

b oppdager en feil:

throw new Unntaksklassen ( ) ;

Normal retur fra b til a:
return 17;

Unntaksklassen er en klasse som vi på forhånd har deklarert som en subklasse av klassen Exception.



Unntak som ikke behandles

Uten og med opprydding

```
int b( ) throws Unntaksklassen {
    . . . .
    . . . .
      Hvis b oppdager en feil:
       throw new Unntaksklassen ( ) ;
          Normal retur:
      return 17;

}
```

```
int b( ) throws Unntaksklassen {
    . . . .
    try {
    Hvis b oppdager en feil:
     throw new Unntaksklassen ( ) ;
    . . . .
     Normal retur:
     return 17;
    }
    finally { rydd opp før kontrollen går
    tilbake til kallstedet }
}
```

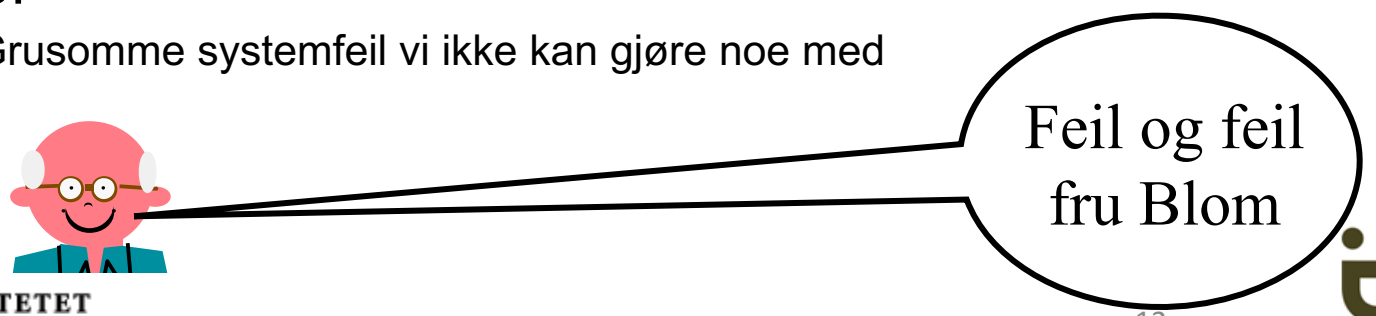
Om å sende unntak tilbake til kaller (throws)

- Vi kan bare sende dem videre til den metoden som kalte oss: **throws**
  (og helt til kjøresystemet: `> java` hvis det er `'main`' som kaster unntak/feilmeldinger videre).
- Vi må da etter metodens parameter-parentes, men før begynnende krøll-parentes, skrive:
  - **throws UnttakType1, UnntakType2, ... {**

    hvor UnttakType1, UnntakType2,... er de typer (klassenavnene) på de unntak som oppstår (eller superklasser av disse, f.eks. Exception) og **ikke** selv fanger med try-catch.
- Merk at vi bruker ordet **både** for unntak metoden vår selv genererer **og** de unntak/feil metoden mottar (fra metoder den selv har kalt) og bare videresender.
- Ulempe med 'videre-kasting' av unntak: Jo nærmere feilkilden feilen blir oppdaget og rettet, jo bedre.

Unntak - oversikt

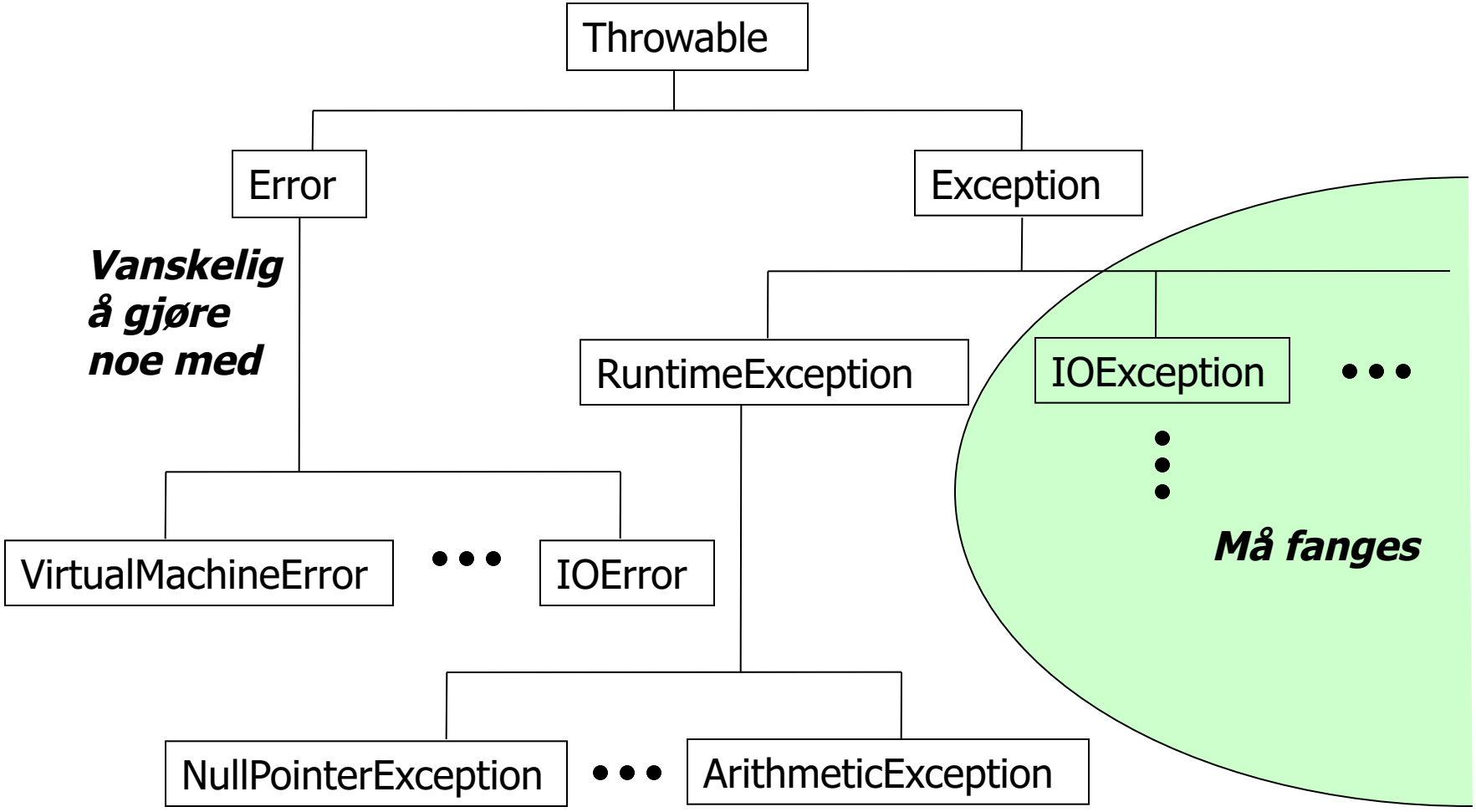
- Når en feilsituasjon oppstår:
  - Lages det et objekt
  - Dette objektet brukes av feilbehandlingen som enten skjer i samme metode (**try-catch**) eller blir sendt tilbake til kallende metode (**throws**)
- Vi må ikke (men kan) behandle feil av typen:
  - RunTimeException
    - Artimetriske feil
    - Array-grense-feil
    - Behandler vi **ikke** slike feil, avsluttes programmet av runtime-systemet
  - Error
    - Grusomme systemfeil vi ikke kan gjøre noe med



Hvilke klasser av feil og unntak har vi i Java

- Exception – unntak (med alle subklassene)
  - Disse unntakene kan og **må** vi fange (Unntatt RunTimeException)
    - F.eks. IOException
  - RunTimeException (en subklasse av Exception) igjen med sine subklasser som : ArrayIndexOutOfBoundsException, NumberFormatException, ArithmeticException, ...
  - Disse kan, men *må vi ikke* fange
  - Vi kan, men må ikke skrive try-catch for disse feilsituasjonene
  - Vi kan kort sagt ignorere disse (men da terminerer programmet stygt)
    - Eks: Divisjon med 0, en peker er null, gal indeks i en array,..
- Error (med alle subklassene)
  - Noe galt skjer, vi kan som oftest ikke gjøre noe med det
  - Eks: .InternalError, OutOfMemoryError, NoClassDefFoundError
- Error og Exception er subklasser av Throwable

Klassehierarki for unntak



Unntak i dette subtreet bør fanges

Unntak – strategier

Flere måter å behandle unntak/avbrudd:

1. Løs problemet og kall metoden som ga unntak om igjen
2. Lapper sammen ting uten å kalle metoden som ga unntak, eller beregn et alternativt (’beste’) resultat istedenfor det unntaks-metoden skulle ha beregnet
3. Avslutt programmet: System.exit(1);
4. Ignorere dem
   - hvis de er av typen RunTimeException eller Error
   - men hvis de oppstår terminerer programmet

Unntak - oversikt, forts.

5. Kaste det videre

   f.eks. public static void main(....) throws IOException

   når "main" kaster en feil videre er det til kjøretidsystemet og programmet terminerer

6. Ta imot / 'fange' det og behandle det ferdig:

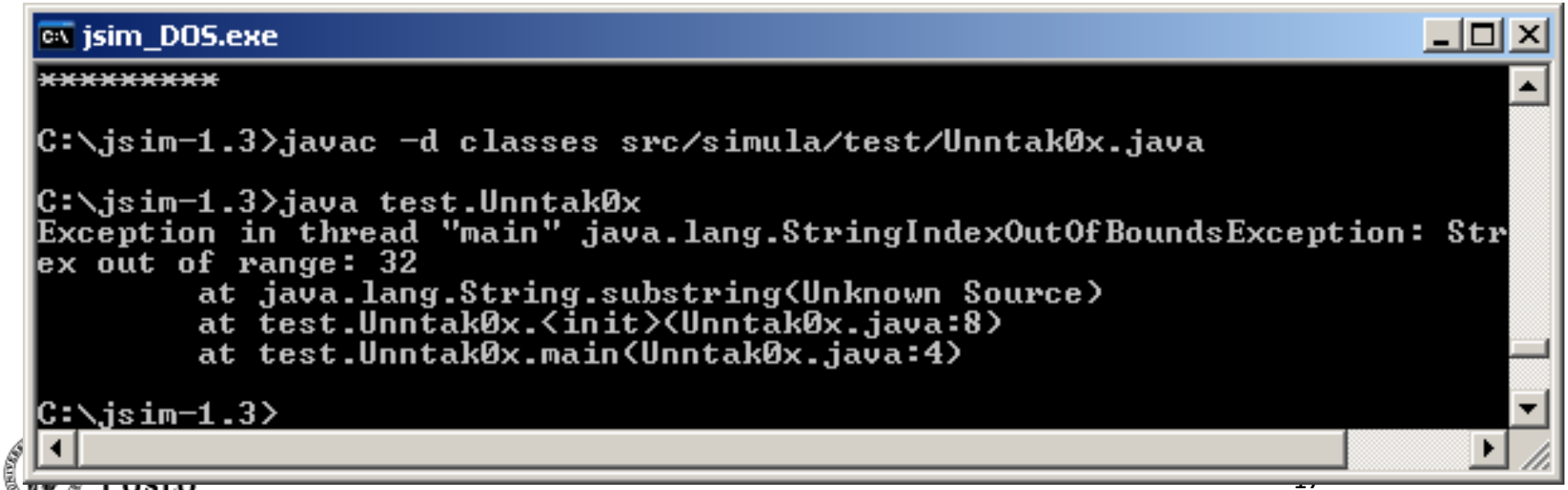
```
try {
    .... farlig kode.…
} catch ( Exception e)  {
    ... gjør noe fornuftig og rett opp feilen
} finally {  det som alltid må utføres  }
```

7. Ta imot, gjøre noe/litt og så kaste det (eller et annet) videre:

```
try {
    .... farlig kode.…
} catch ( Exception e ) {
    ... gjør noe fornuftig, f.eks. rett opp litt av feilen  og så
        throw e;
} finally {det som alltid må utføres OGSÅ ved viderekasting}
```

String-indeks utenfor stringen

```
class Unntak0x {
        public static void main(String[ ] args) { new Unntak0x ( ); }
    Unntak0x ( ) {
       String s = "Dette er en tekst med 29 tegn", s1;
          s1 = s.substring(30,32);   // string-indeks utenfor "enden"
    }
 }
```



Behandler Stringindeks-feil

```
class Unntak1x {
   public static void main(String[ ] args) { new Unntak1x ( ); }

   Unntak1x ( ) {
      String s = "Dette er en tekst med 29 tegn", s1;
      try {
         s1 = s.substring(30,32); // string-indeks utenfor "enden"
      } catch (StringIndexOutOfBoundsException e) {
         System.out.println("Her er det noe galt med string-indeksen ”);
      }
   }
}
```

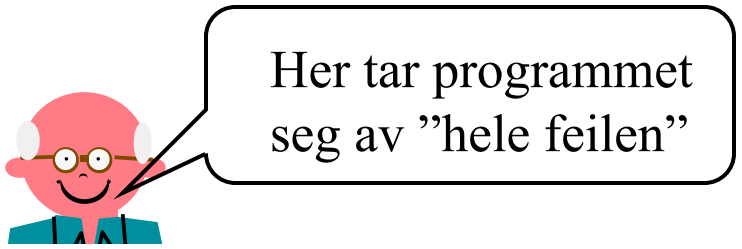
```
>java Unntak1x
Her er det noe galt med string-indeksen
```

Fange divisjon med ‘0’

```
public class TryTest
{
    public static void  main ( String [ ] args)
     {     int i=1;
          for (int j=0; j < 5; j++)
               try{
                 i = 10/j;
                 System.out.println("Det gikk OK, i:" + i + ", j:" + j);

               } catch (Exception e)  {
                  System.out.println("Feil i uttrykk: "+ e.getMessage( ));
               }

      }
 } // end TryTest
```




```
snidil> java TryTest
Feil i uttrykk: / by zero
Det gikk OK, i:10, j:1
Det gikk OK, i:5, j:2
Det gikk OK, i:3, j:3
Det gikk OK, i:2, j:4
snidil>
```

Eksempel på bruk av unntaksobjektet

```
class Unntak2x {
   public static void main(String[ ] args) { new Unntak2x ( ); }

   Unntak2x ( ) {
      String s = "Dette er en tekst med 29 tegn", s1;
      try {
         s1 = s.substring(30,32); // string-indeks utenfor "enden"
      } catch (StringIndexOutOfBoundsException e) {
         System.out.println("Her er det noe galt med string-indeksen " +
                                                              e.getMessage( ));
     } //end try
   }
}
```

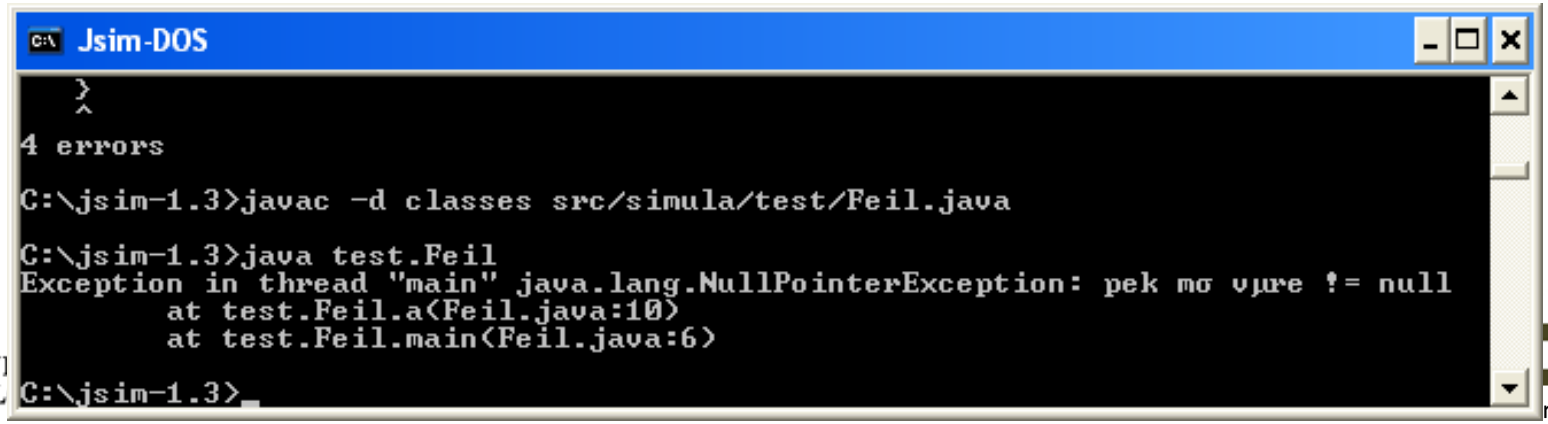
```
>java Unntak2x
Her er det noe galt med string-indeksen String index out of range: 32
```

Starte å sende feil/unntak selv

```
class Feil1 {
    int i;
    public static void main(String[] args) {
        new Feil1().a(null);
    }

    void a(Feil1 pek) {
      if (pek == null )
            throw new NullPointerException("pek må være != null");
        pek.i = 14;
    }
}
```

- Merk at her kastes et objekt av en klasse som er subklasse av `RunTimeException`, så da trenger vi ikke `try-catch` rundt kallet på metoden 'a'



Flere **catch** pluss **finally**

- Vi kan ha flere catch etter hverandre:

```
try {
    ... Kode som kan gi unntak / feile ...
} catch (Type1 t1 ) {

} catch (Type2 t2 ) {
            ...
} catch (Type3 t3 ) {
            ...
} finally {
      // Dette gjøres alltid  - også om unntaket ikke blir behandlet
      // men bare kastet videre
}
```

- En og en catch-parameter testes:
  - Bare den første der klassenavnet (Type1, Type2,..) er **super**klasse (eller samme klasse) til det innkomne unntakets klasse blir utført.
- Finally vil *alltid* bli utført enten det ble unntak eller ikke og enten noen av catchene fikk tilslag eller ikke
  - Spesielt viktig å ha med hvis unntak blir kastet direkte videre og det er nødvendig å rydde opp

Fange flere unntak - eksempel

```
class Unntak3x {
   public static void main(String[ ] args) {
     int dividend=7, divisor = 0; int kvotient=0;
     String s="Dette er en tekst med 29 tegn";    String s1="*********";
     try {
         s1 = s.substring(15,17);        // OK string-indeks
         kvotient = dividend/divisor; // Feil: divisjon med 0
     }   catch (StringIndexOutOfBoundsException e) {
         System.out.println("Her er det noe galt med string-indeksen");
     }   catch (ArithmeticException e1) {
         System.out.println("Divisjon med 0: " + e1.getMessage( ) );
     }
     System.out.println(s1);
     System.out.println(kvotient);
   }
}
```

```
>java Unntak3x
Divisjon med 0: / by zero
st
0
```

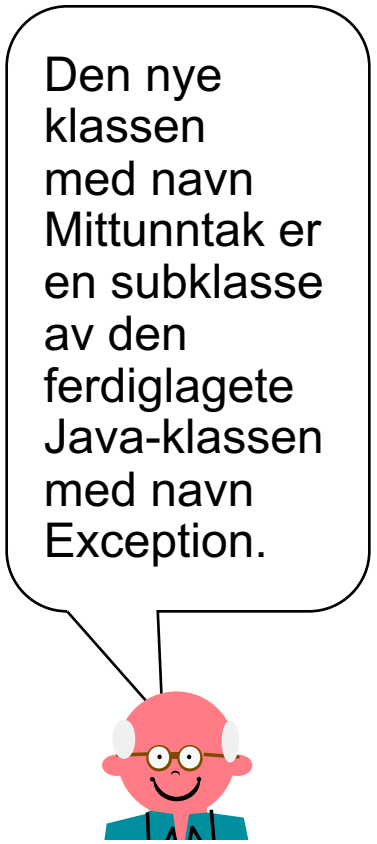
Mer egendefinerte unntak

```
class Mittunntak extends Exception {
        public Mittunntak ( ) { }
        public Mittunntak (String s) { super(s); }
        // end konstruktører
} // end class Mittunntak
```

```
try{ . . . }
catch (Mittunntak e) { . . e.getMessage( ) . .  }
```

```
throw new Mittunntak("feilmelding");
```

```
throw new Mittunntak( );
```



Eksempel

Konto med OvertrekkUnntak - VIKTIG

```
class Konto {
   private double saldo = 0;
   private int kontonr;

   Konto (int nr) {  kontonr = nr;  }

   public void taUt (double belop) throws OvertrekkUnntak {
        if  (saldo - belop < 0) {
               throw new OvertrekkUnntak(Integer.toString(kontonr));
         }
         else   saldo = saldo - belop;
    }

    public void settInn …….

}
```

```
class OvertrekkUnntak extends Exception {
      public OvertrekkUnntak (String s)  {
            super(s);
      }
}
```

Bruk av Konto med OvertrekkUnntak

```
class Bank{
    static void main (String [] args) {
        Konto pek = new Konto(234);

        try    {

                pek.settInn(1000);

                 pek.taUt(500);

                 pek.taUt(5000);
        }
        catch (OvertrekkUnntak e) {
              System.out.print(" Overtrekk på konto nummer ");
              System.out.println( e.getMessage());
        }
    }
}
```