UNIVERSITETET I OSLO

Institutt for informatikk

INF1010 våren 2019

Onsdag 30. januar

Mer om unntak i Java

(med litt repetisjon av I/O først)

Stein Gjessing

Institutt for informatikk

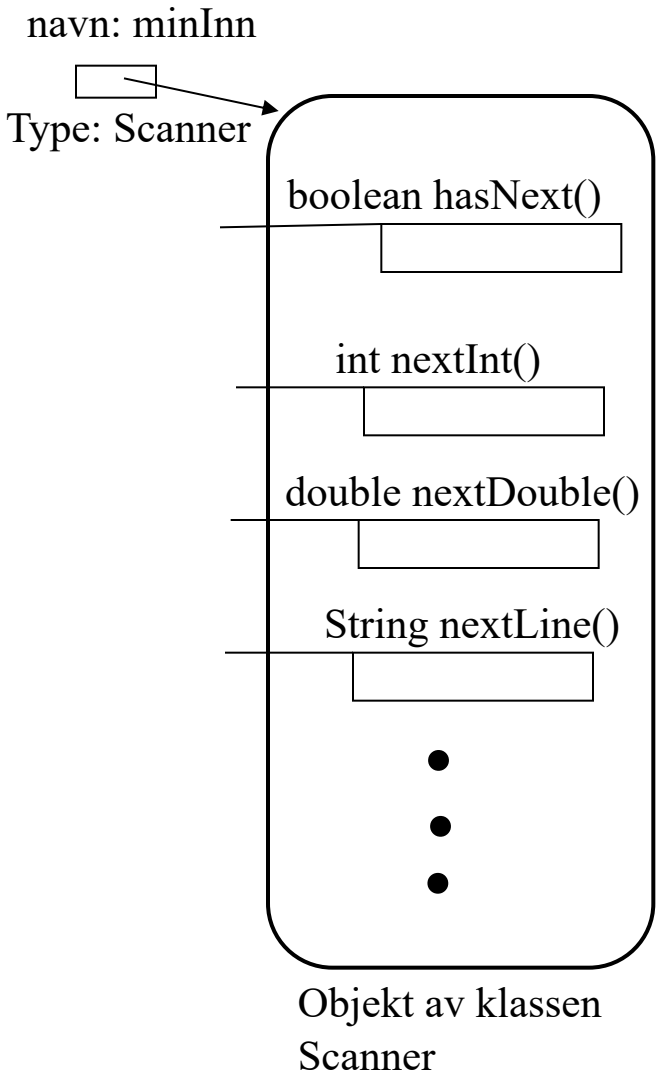
Lesing fra terminal og fil

Bruk Scanner:

Scanner minInn = new Scanner(<fil>);

Se Scanner i java-biblioteket

F.eks. ved å google "java 8 api Scanner"



Klassen Scanner

- Leser en og en linje eller ett og ett "token"
- "Tokens" er vanligvis adskilt av blanke
- Kan sjekke om det er flere linjer eller token igjen på fila
- Kan lese og konvertere token til basale typer
  som f.eks. int og double

- Bruk CNTRL-D for slutt på input fra terminal

Feilhåndtering

- En **metode** som kan komme til å gjøre en IO-feil på fil må enten behandle denne selv, eller *kaste feilen videre* (også i main):

```
public void  mittProgMedIO( ) throws IOException {
    < kode som gjør fil-behandling>
}
```

I Java-biblioteket:

```
class IOException extends Exception {
  .........
}
```

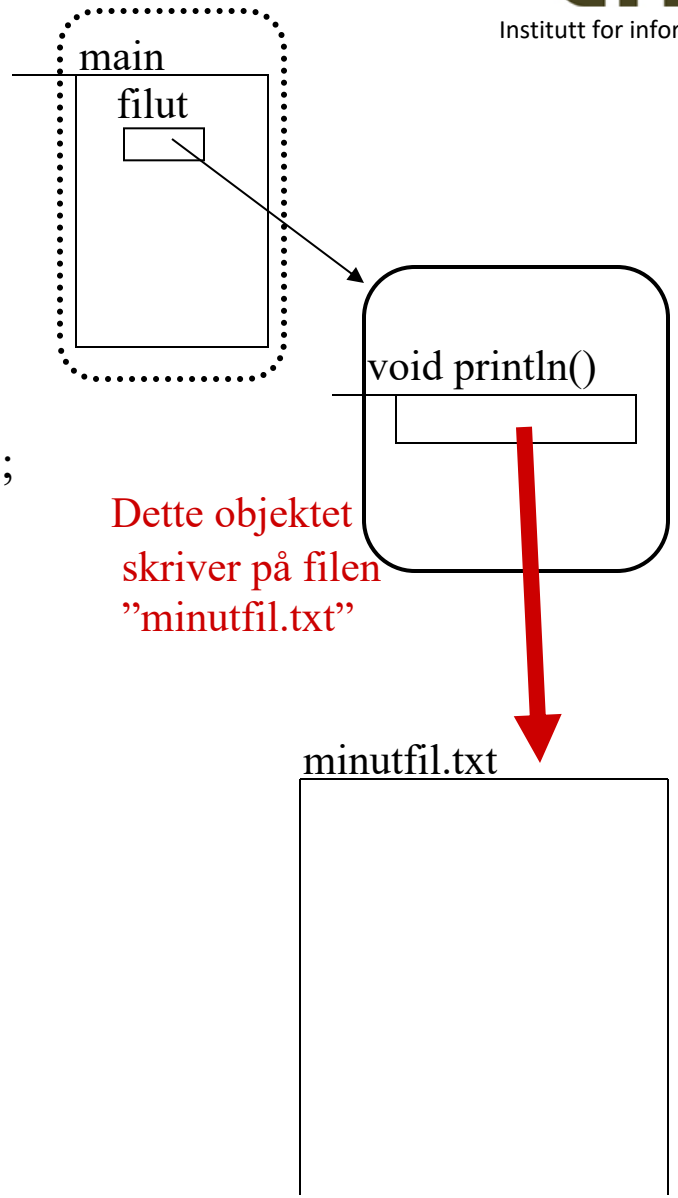
throws
er et Java
nøkkelord

Metoder som gjør filbehandling kan kaste unntaket til main

```
import java.io.*;

class FilTestMedMetode {

    public static void main (String [ ] args)   throws FileNotFoundException {
            skrivTilFil();
    }

    private static void skrivTilFil( ) throws FileNotFoundException  {
            PrintWriter  filut = new PrintWriter (”minutfil.txt”);
             filut.println( ”Meget enkel testutskrift nr. ” + 17 );
             filut.close( );
    }
}
```

Eller behandle unntaket selv:

```
import java.io.*;

try {
  PrintWriter  filut = new PrintWriter (”minutfil.txt”);

  // Utskrift skjer som til skjerm:

  filut.println( ”utskrift” + 17 ):

  // For at innholdet på den nye filen skal
  // bevares må vi til slutt si:

  filut.close( );

}
catch (FileNotFoundException f) { . . . }
```



Når du behandle unntaket selv (til høyre)

```
import java.io.*;

class FilTest {

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

     PrintWriter  filut =
              new PrintWriter (”minutfil.txt”);
     filut.println( ”Utskrift” + 17 );
     filut.close( );



   }
}
```

```
import java.io.*;

class FilTest {

  public static void main (String [ ] args) {

      try {
          PrintWriter  filut =
              new PrintWriter (”minutfil.txt”);
          filut.println( ”Utskrift” + 17 );
          filut.close( );
      }
       catch (FileNotFoundException f) {
           System.out.println (”Fant ikke filen”);
       }
   }
}
```

Innlesing fra terminal


```
import java.util.*;

class LesFraTermNavnAlder {
      public static void main (String [ ] args) {
            int alder;
            String navn;

            Scanner  minInn =  new Scanner  (System.in);

            System.out.print(" Skriv navn: ");
            navn = minInn.nextLine();
            System.out.print(" Skriv alder: ");
            alder = minInn.nextInt();

            System.out.println(" Du heter " + navn +
                        " og er " + alder + " år" );
    }
}
```

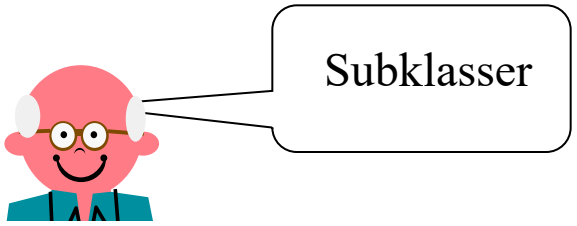
Innlesing fra fil


```
import java.util.*;
import java.io.*;

class LesFraTermNavnAlder throws FileNotFoundException {
      public static void main (String [ ] args) {
            int alder;
            String navn;
            Scanner  minInn =  new Scanner  (new File ("minfil.txt") );

            navn = minInn.nextLine();
            alder = minInn.nextInt();

            System.out.println(" Personen heter " + navn +
                        " og er " + alder + " aar" );
    }
}
```

minfil.txt

```
Per Pettersen
21
```

FileNotFoundException er dette spesielle unntaket
IOException er et mer generelt unntak
Exception er det mest generelle unntaket



Innlesing fra fil - feil

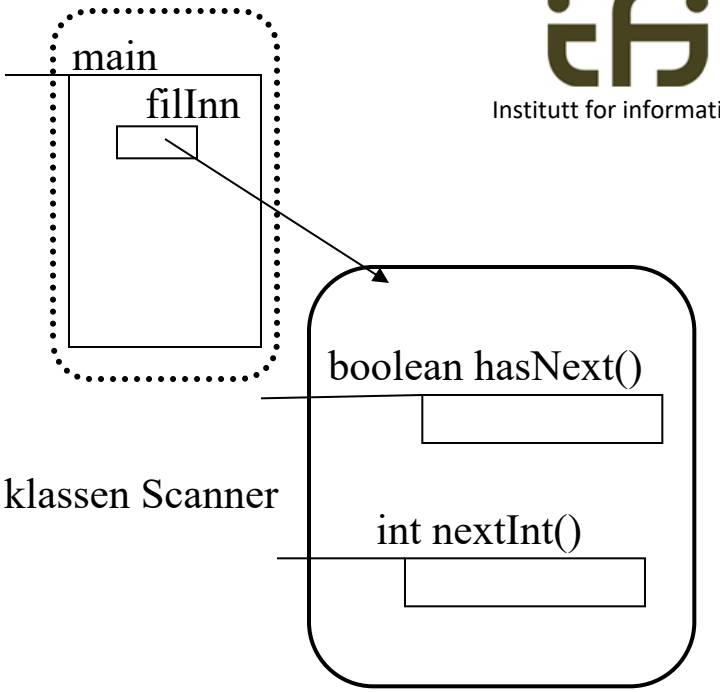

```
import java.util.*;
import java.io.*;

class LesFraTermNavnAlder {
      public static void main (String [ ] args) {
            int alder;
            String navn;
            Scanner  minInn =  new Scanner  (new File ("minfil.txt") );

            navn = minInn.nextLine();
            alder = minInn.nextInt();

            System.out.println(" Personen heter " + navn +
                        " og er " + alder + " aar" );
    }
}
```

minfil.txt

```
Per Pettersen
21
```

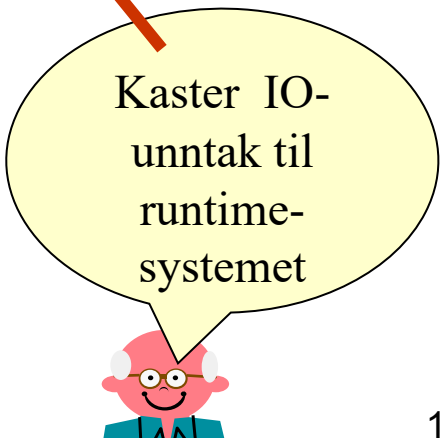
```
java:8: error: unreported exception FileNotFoundException; must be
caught or declared to be thrown
        Scanner  minInn =  new Scanner  (new File ("minfil.txt"));
                           ^
1 error
```

Innlesing av heltall fra fil – utskrift på skjerm



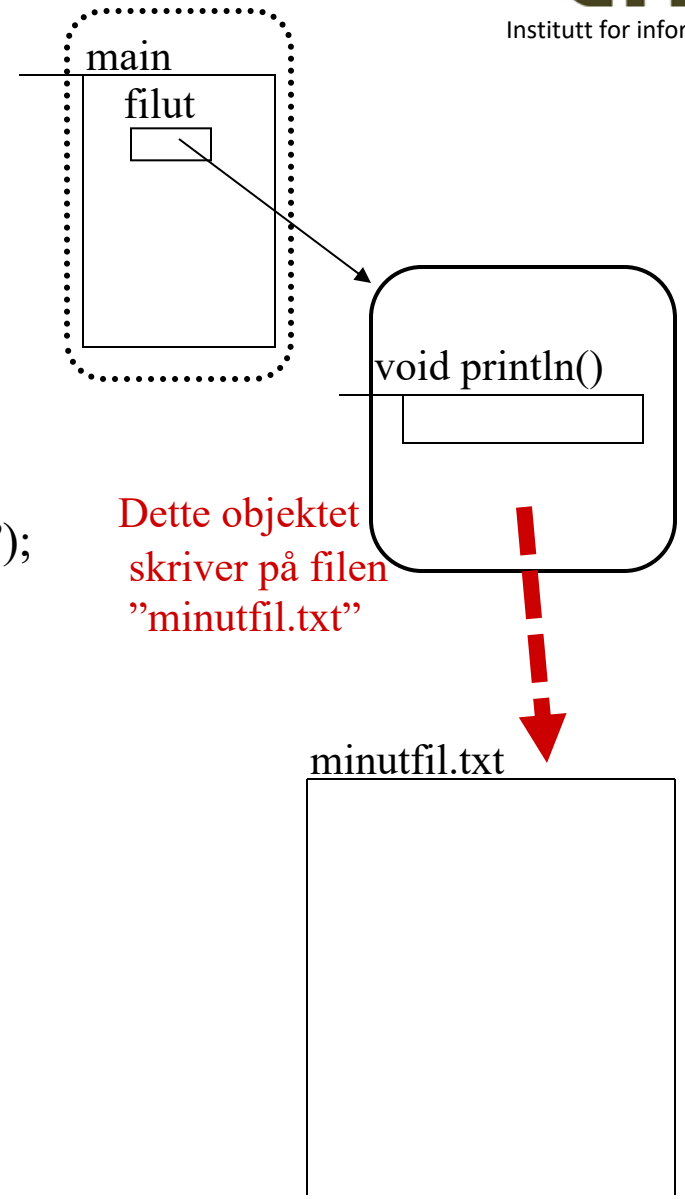

```
import java.io.*;
import java.util.*;

class LesFraFil {

   public static void main (String [ ] args) throws FileNotFoundException {
        Scanner  fillnn =   new Scanner  (new File ("minfil.txt") );
        int tall;
        while(fillnn.hasNext())  {
                tall = terminalInn.nextInt();
                System.out.println(tall);
        }
    }

}
```

DETTE GÅR BRA HVIS FILEN BARE INNE-HOLDER HELTALL !!



Utskrift til fil


```
import java.io.*;

class FilTest {

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

     PrintWriter  filut = new PrintWriter (”minutfil.txt”);

     // Utskrift skjer som til skjerm:

     filut.println( ”Utskrift” + 17 );

     // For at innholdet på den nye filen skal
     // bevares må vi til slutt si:

     filut.close( );
   }
}
```



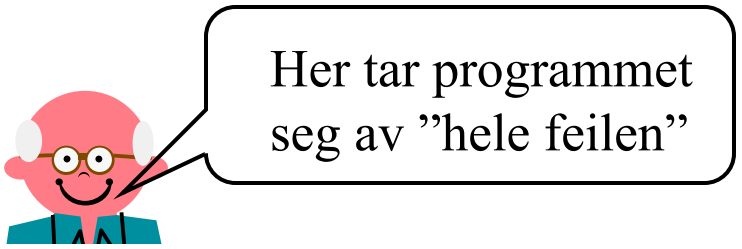
Array indeks utenfor sine grenser


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

```
Seacobra:programmer steing$ javac Test.java
Seacobra:programmer steing$ java Test
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 101
at Test(Test.java:8)
Seacobra:programmer steing$
```

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( ));
            }

    }
}
```




```
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>
```

Generelt om 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:

```
try {
        <... Kode som kan feile …>
 }
 catch  (Exception e)  {
        < .... Gjør noe med feilen ,
              prøv å rett opp ...>
 }
```

Feiler koden blir denne blokken utført med feilobjektet som e peker på som parameter

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 behandler
  et unntak.
  Har en referanse til et unntaksobjekt som parameter
- **finally** - blir alltid utført (mer senere)

- **throw** - Starter å kaste et unntak
  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

- **Viktigst bruk:**

```
try {
        <kode som kan feile>
}
catch (Unntaksklasse u) {
                <behandle  unntaket,  u peker på et objekt som beskriver  unntaket>
 }
```

Unntak kan **oppstå** i egen kode


```
try {
   <KODE SOM KAN FEILE>
         <Når det skjer noe galt
          f.eks. at en referanse er null:>
         throw new Unntaksklasse( );
 . . . .
 . . . .
}
catch (Unntaksklasse  unt) {
  < Unntaksbehandling.
     Dette hoppes over  når intet
     unormalt/galt/feil har hendt >
}

< her fortsetter programmet
  både etter normal utføring og etter
  behandling av eventuelle unntak >
```

På forhånd har vi deklarert:

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

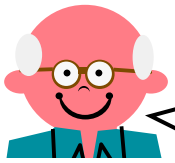
Når unntak oppstår i en annen metode (og ikke behandles der)

inne i metoden a:

```
try { . . . .
      x = b ( );
 . . . .
}
catch (Unntaksklasse  unt) {
  < Unntaksbehandling.
     Dette hoppes over
     når intet unormalt
     har hendt >
}
< her fortsetter programmet
  både etter normal utføring og
  etter behandling av
  eventuelle unntak>
```

kall på metoden b

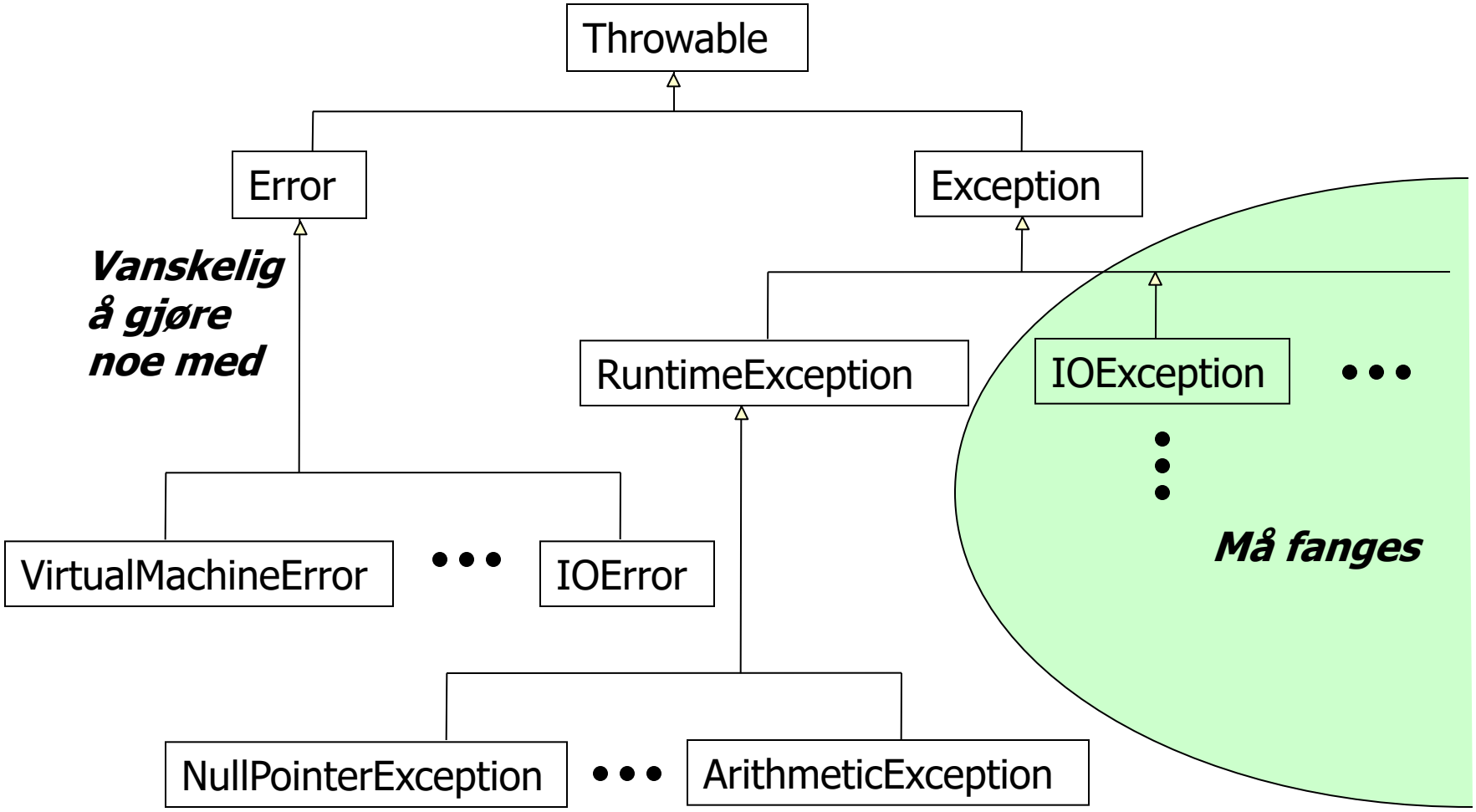
```
int b( ) throws Unntaksklasse {
   inne i metoden b:

   En feil oppdages:
   throw new Unntaksklasse ( ) ;

   Normal retur fra b til a:
   return 17;

}
```



Metoden b feiler kanskje fordi kontrakten for kall på metoden ikke ble oppfylt.

Unntaksklasse er en klasse som på forhånd er deklarert (egendefinert eller definert i Java-biblioteket) som en subklasse av klassen Exception (se forrige side).

Java-bibliotekets klassehierarki for unntak



Unntak i dette subtreet bør fanges