UNIVERSITETET I OSLO

Institutt for informatikk

INF1010 våren 2017

Onsdag 25. januar

Litt om unntak i Java

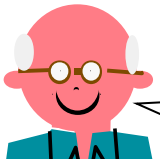
Stein Gjessing

Institutt for informatikk

Nytt tema: Feilhåndtering (IO: Innlesing/Utskrift)

- 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>
}
```



Vi skal lære mer om unntak og feilbehandling etter hvert, **bare litt her i dag**

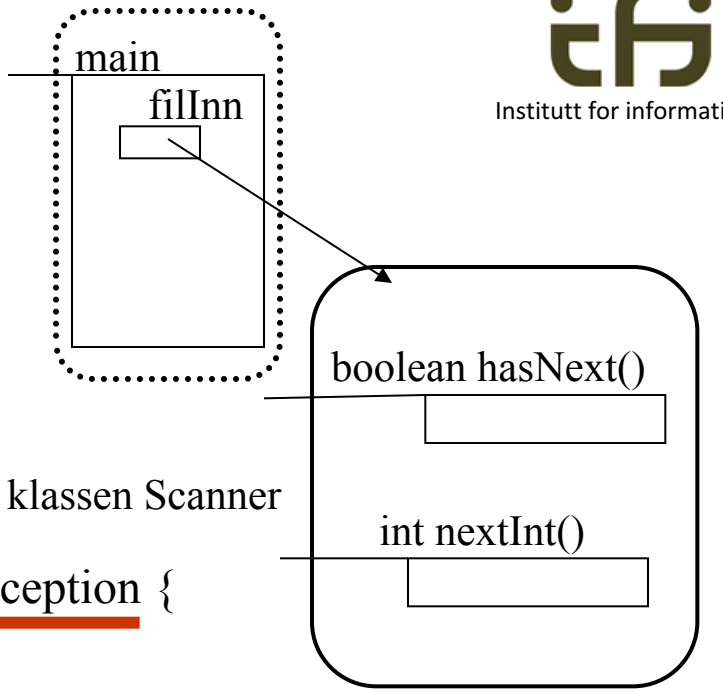
I Java-biblioteket:

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

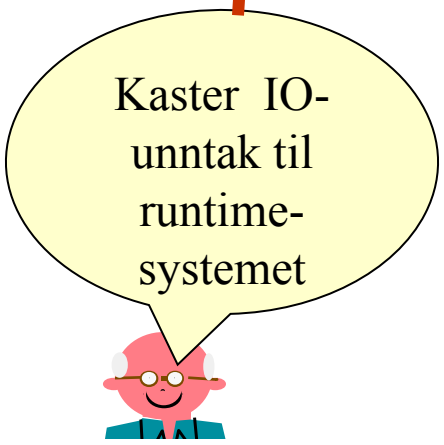
(IO: Innlesing/Utskrift)

throws er et Java nøkkelord

Innlesing av heltall fra fil – utskrift på skjerm




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

class LesFraFil {

   public static void main (String [ ] args) throws IOException {
        Scanner  filInn =
            new Scanner  (new File ("minfil.txt"));
        int tall;

        while(filInn.hasNext())  {
                    tall = filInn.nextInt();
                    System.out.println(tall);
        }
    }

}
```



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

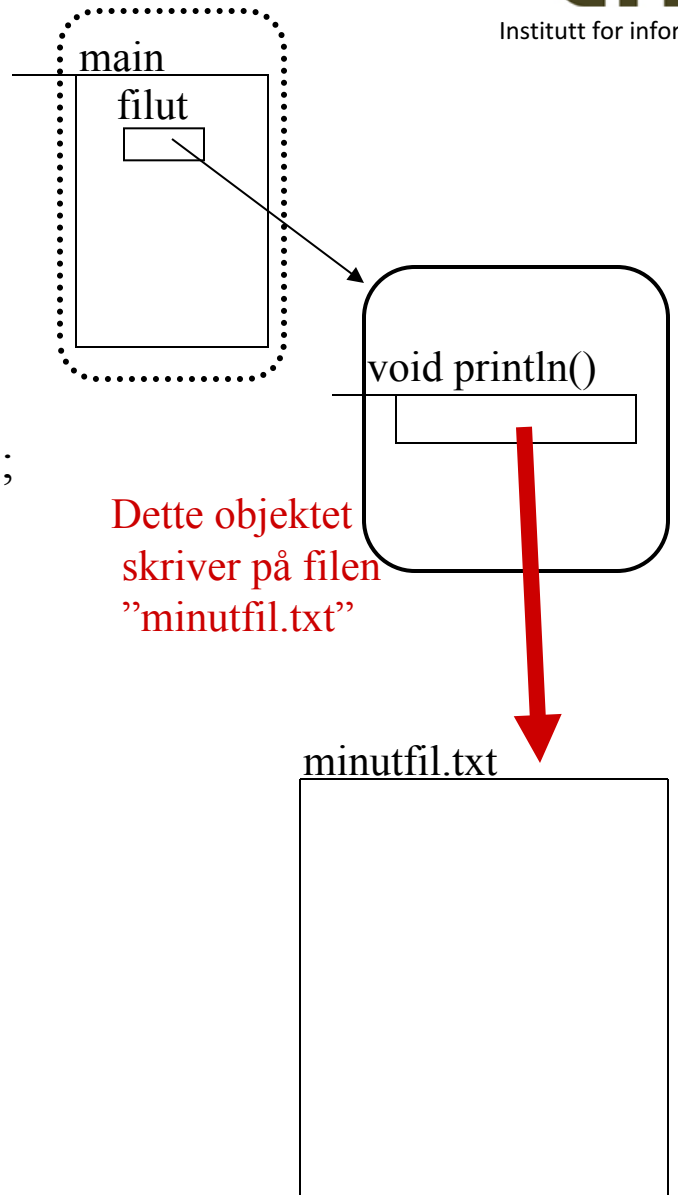
Array indeks utenfor sine grenser


```
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$
```

Du kan 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) { . . . }
```



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)

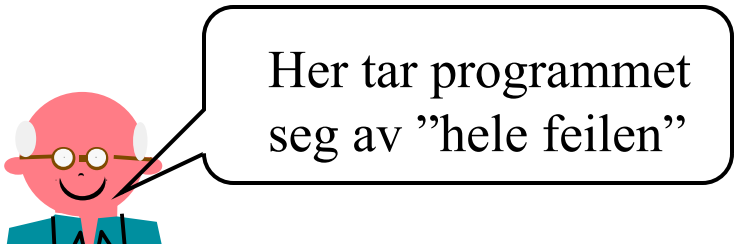
- **throw** - Starter å kaste et unntak
  throw <en peker til et unntaksobjekt>
  f.eks throw new Unntak();
- **throws** - Kaster et unntak videre
  Brukes i overskriften på en metode som
  ikke selv vil behandle et unntak

- **Viktigst bruk:**

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

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>
```

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:

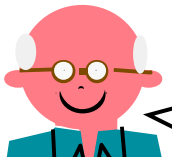
```
try { . . . .
      x = b ( );
 . . . .
}
catch (Unntaksklasse  unt) {
  < Unntaksbehandling.
     Dette hoppes over
     når intet unormalt
     har hendt >
}
< her fortsetter programmet
  både etter normal utføring og
  etter behandling av
  eventuelle unntak>
```

kall på metoden b

```
int b( ) throws Unntaksklassen {
```

inne i metoden b:

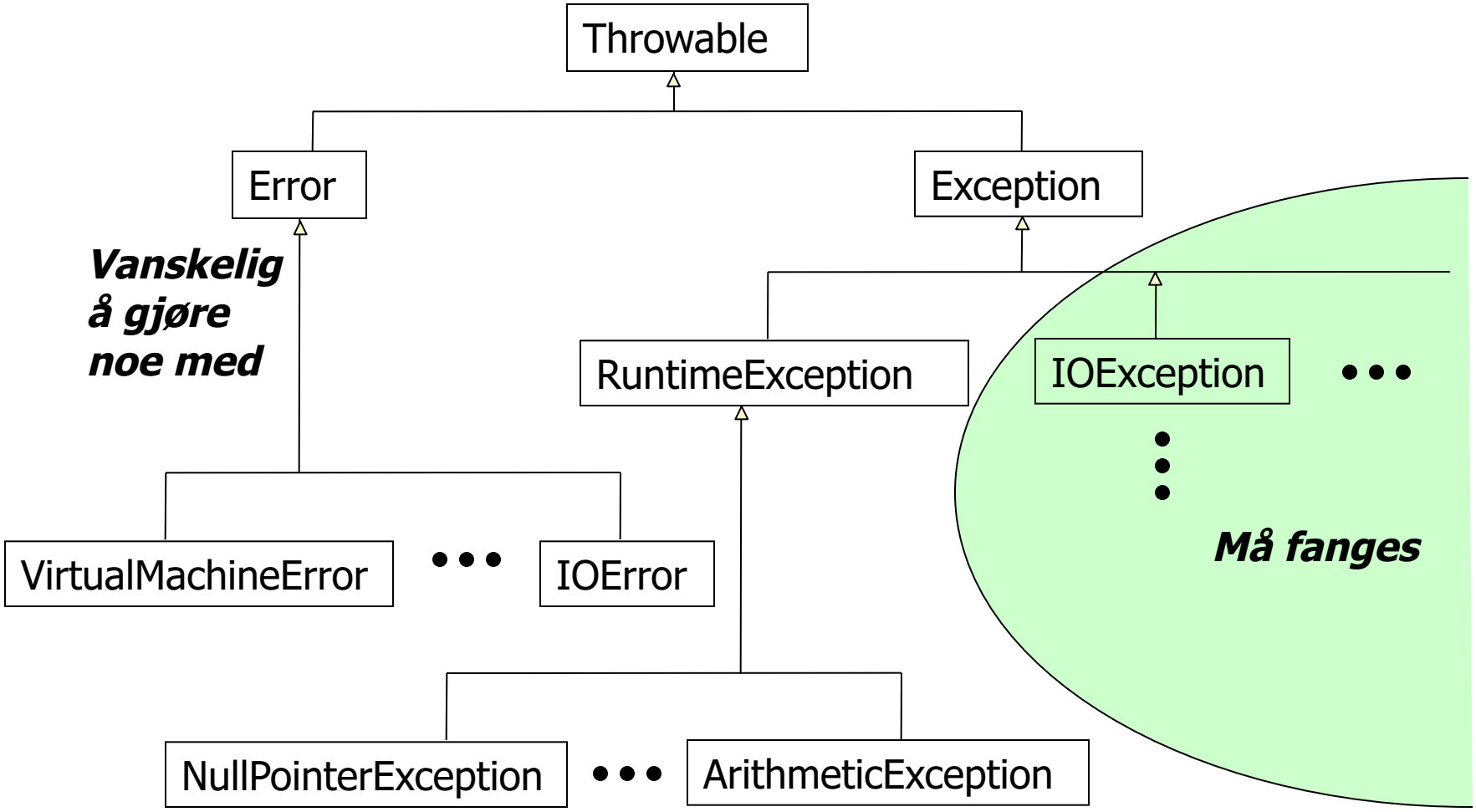
```
  b oppdager en feil:
  throw new Unntaksklasse ( ) ;

        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