UNIVERSITETET I OSLO

Institutt for informatikk

# INF1010 våren 2018

## tirsdag 23. januar

# I/O og litt om bruk av unntak i Java

Stein Gjessing

Institutt for informatikk

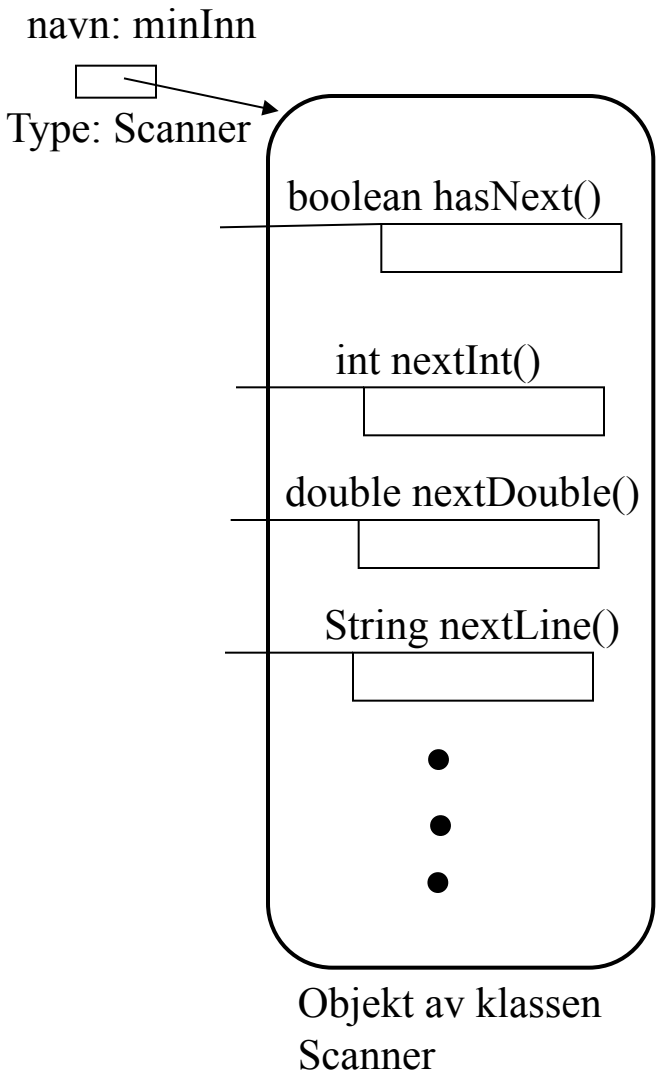
# Lesing fra terminal og fil

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

# 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 - 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 fra fil - riktig

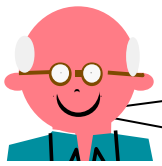


```
import java.util.*;
import java.io.*;

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
```

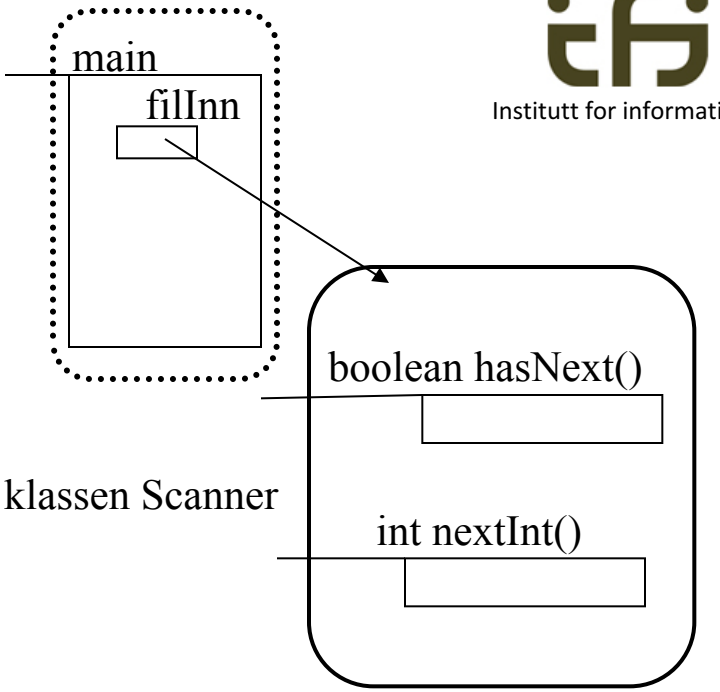
FileNotFoundException er dette spesielle unntaket
IOException er et mer generelt unntak
Exception er det mest generelle unntaket



Mer når vi lærer om subklasser

# Innlesing av heltall fra fil – utskrift på skjerm

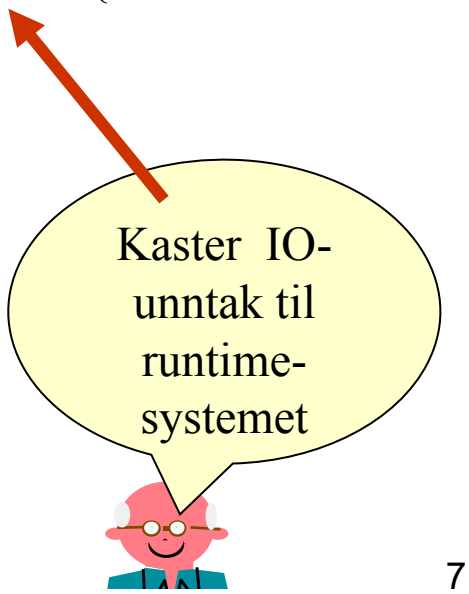





Objekt av klassen Scanner

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

class LesFraFil {

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

}
```

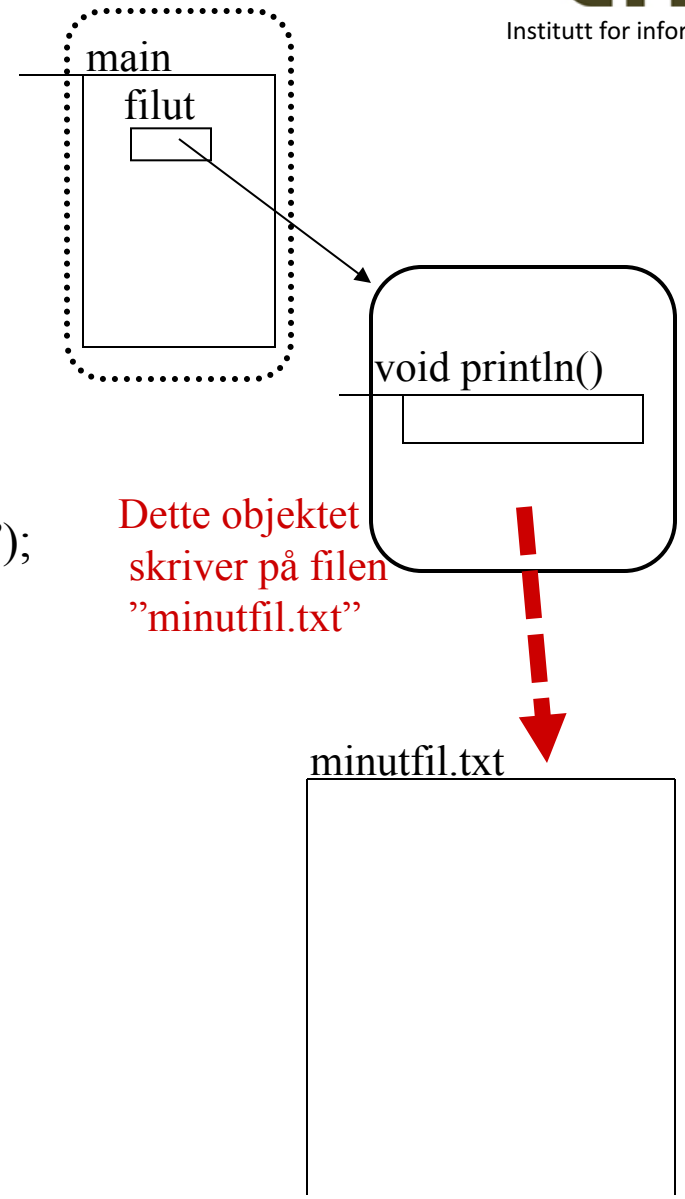
DETTE GÅR BRA HVIS FILEN BARE INNE-HOLDER HELTALL !!



# 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( );
   }
}
```



# **Metoder** som gjør filbehandling må kaste unntaket til main (etc.)



```
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( );
    }
}
```

# Du kan behandle unntaket selv



```
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”);
      }
   }
}
```

# 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 legger 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

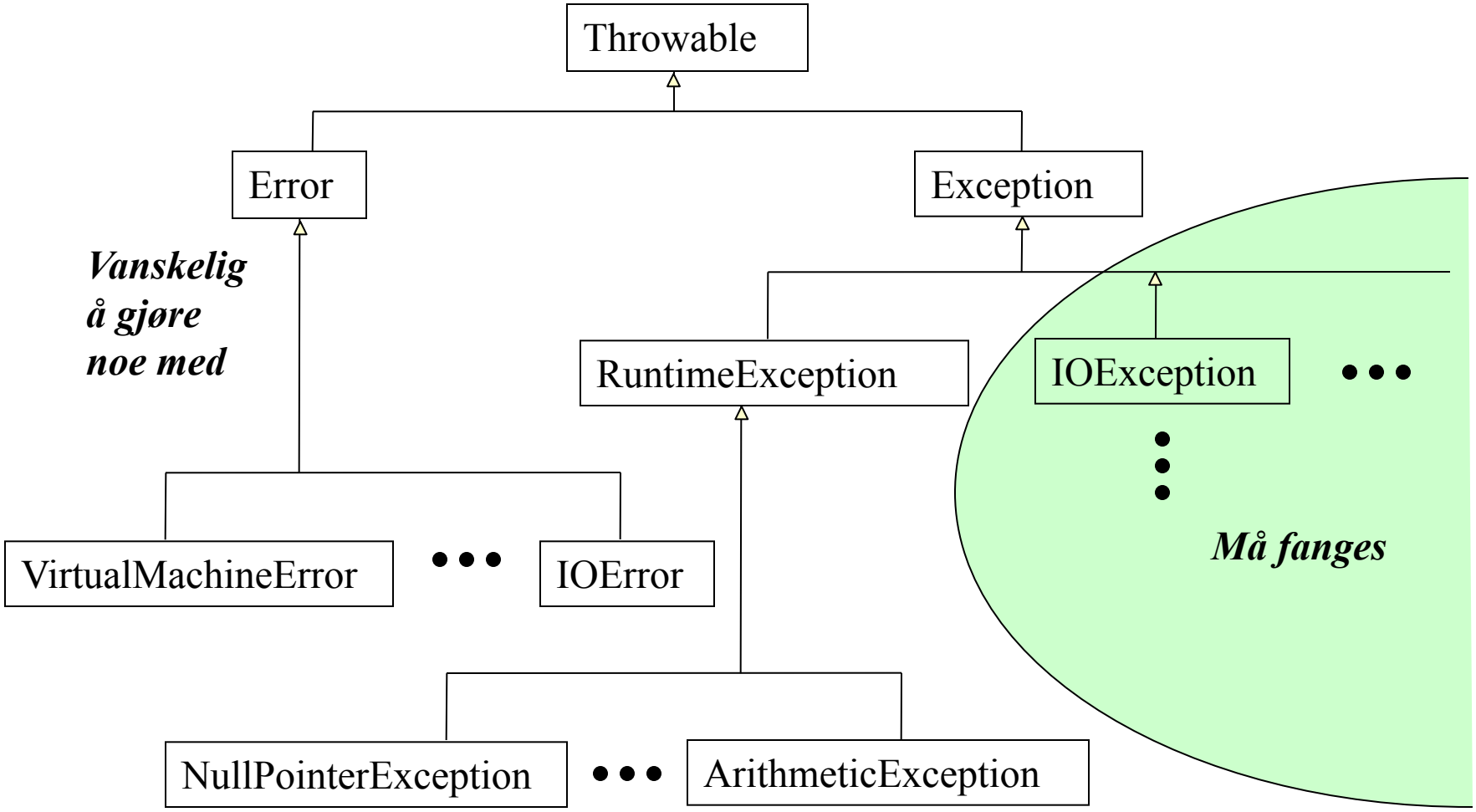
# Fem reserverte Java ord



- **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 peker til et unntaksobjekt som parameter
- **finally** - blir alltid utført (mer senere)
- **throw** - Starter å kaste et unntak (mer senere)
- **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>
}
```

# Java-bibliotekets klassehierarki for unntak




***Unntak i dette subtreet bør fanges***