Fra Python til Java

Uke 1 – dag 3 (IN1010-tidsregning)

Mandag 30. januar 2023

Beskjeder

- Seminartimene
  - Ser ut til å være plass på alle grupper, med en del variasjon
  - Du kan bytte om din gruppe ikke passer:
    send e-post ny gruppelærer; om OK, gi beskjed til opprinnelig gruppelærer
  - Oversikt over grupper og gruppelærere under Grupper
    Tid og sted under Timeplan

Læringsmål

Forrige uke og denne:

- Skrive og bruke klasser i Java
- Vite hvordan et Java-program representeres og utføres
  - fra programkode
  - fra øyeblikksbilder av datastrukturen

I dag

- Lære mer Java
- Kunne lese og skrive til/fra fil og terminal
- Manipulere og konvertere strenger
- Bruke arrayer og for-løkker

Oversikt over dagens tema

- Lese inn data i Java – med diverse tilbehør
  - Java API (klassebiblioteket), packages, import og aksessmodifikatorer
  - Exceptions (unntakshåndtering)
  - Klassene File og Scanner
  - Lese ulike typer verdier
- Skrive til fil (PrintWriter)
- Tekststrenger (String), manipulering og konvertering
- Arrayer og for-løkker

Fra IN1000 uke 8: Python
Program med klasse i egen programfil

```
class Eksempel:
   def __init__(self,p1):
        self._instvar1 = p1
```

eksempel.py

Klasser skrives ofte slik at de er nyttige i mange ulike programmer og kan gjenbrukes.

Eks: strenger og lister

```
from eksempel import Eksempel

def hovedprogram ():
    …
    # oppretter og bruker objekter

hovedprogram()
```

hovedprogram.py

2 klasser, den ene med et hovedprogram som kjøres

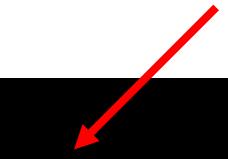
```
class Eksempel {
    int instVar;

    Eksempel (int p1) {
        instVar = p1;
    }
}
```

Eksempel.java

Filnavn = Klassenavn + ".java"

Vi trenger ikke importere klasser som ligger i samme folder.



```
class Hovedprogram {
    public static void main (String[] args) {
        // oppretter og bruker objekter
        Eksempel k = new Eksempel(5);
    }
}
```

Hovedprogram.java

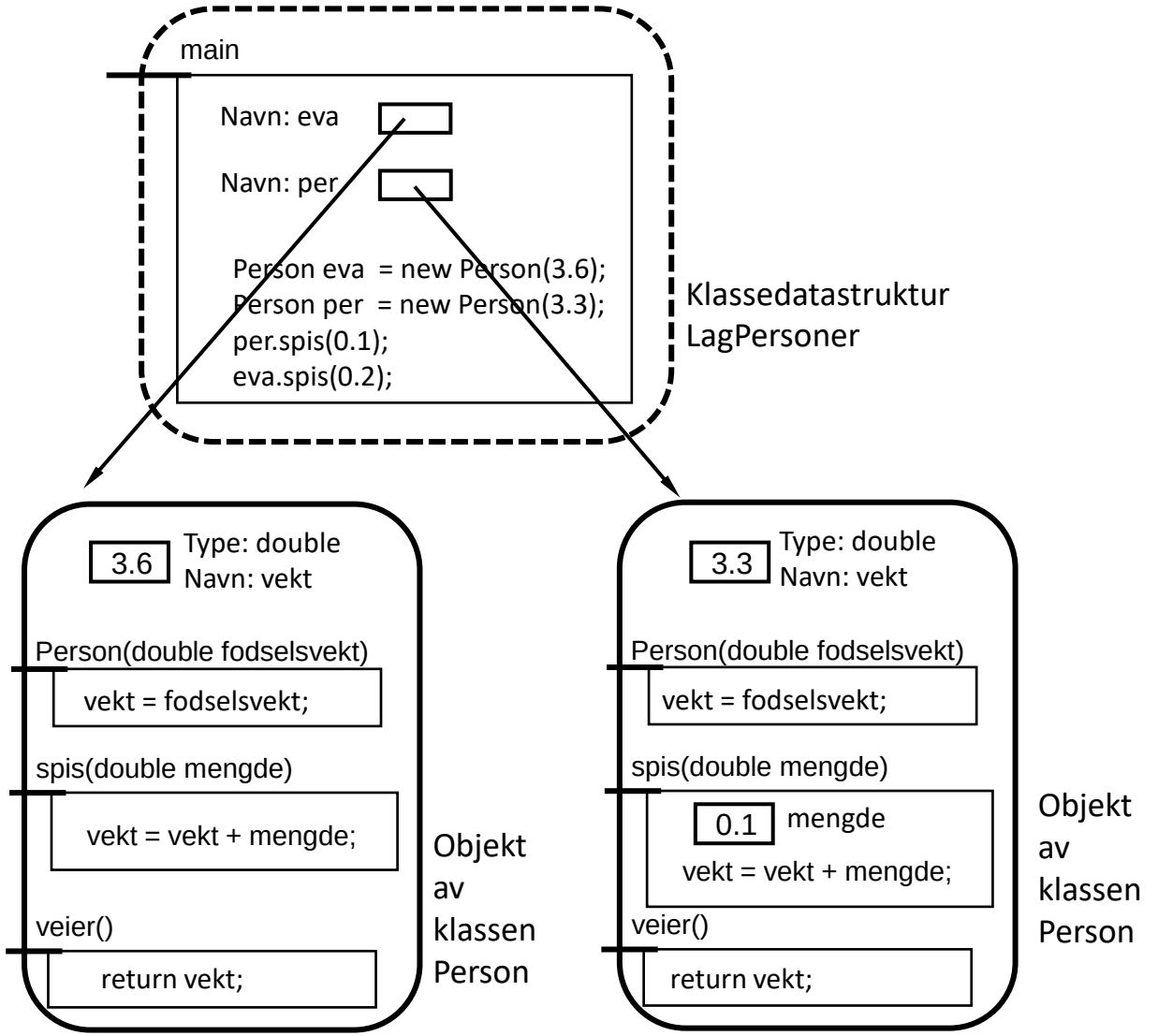
Hovedprogrammet som utføres ligger i main-metoden – og kjøres her med kommandoen:
>java Hovedprogram

Øyeblikksbilde (tilstand) når Per er i ferd med å spise 100g

```java
class Person {
    private double vekt;
    public Person(double fodselsvekt) {
      vekt = fodselsvekt;
    }
    public void spis(double mengde) {
          vekt = vekt + mengde;
    }
    public double veier() {
      return vekt;
    }
}

class LagPersoner {
    public static void main(String[ ] arg) {
       Person eva  = new Person(3.6);
       Person per  = new Person(3.3);
       per.spis(0.1);
       eva.spis(0.2);
    }
}
```

Har Per fått maten inn i munnen?
(Hvilken setning er neste til å bli utført?)



Bli bedre kjent med Java

- Python og Java har ulike underliggende modeller og implementasjon, og noe ulik terminologi

- I IN1000 gikk vi ikke veldig dypt i "hva skjer under overflaten"
  - ikke nødvendig for å bruke mekanismene i IN1000 pensum
  - IN1000 pensum inkluderte ikke alle konseptene (som statiske egenskaper, arv)
  - ville gjort overgangen til Java tyngre

- Men:
  - IN1010-pensumet gir grunnlag for, og krever, dypere forståelse
  - Dere skal bruke mange klasser i Java biblioteket ("Java API")
  - Lær å bruke Java API dokumentasjon allerede nå

Java ~ Java Development Kit (JDK)

- JDK (Standard Edition – SE) inneholder alt du trenger for utvikling og kjøring av Java programmer:
  - Java språket/ kompilator
  - Java API (Application Programming Interface)/ klassebibliotekene
  - Java Runtime Environment (JRE) med Java Virtual Machine (JVM)
  - ++

- Stadig nye versjon av JDK. IN1010 versjon = subsett av Java 8 (=1.8)
  - Java 8 har alt vi trenger og bruker i IN1010
  - Kjører i alle nyere versjoner dvs på alle platformer
  - Fokus på grunnleggende mekanismer for videre påbygging
  - Lettere å gå til nyere versjoner enn motsatt vei

Lese fra fil

Se notatet: "Lesing og skriving i Java" på semestersiden

Navnestatistikk i Python

- Hva er navnet på eldste person i en fil?

alder.txt

```
Petter 92
Kari 92
Emil 101
Katrine 99
```

finn_eldste_navn.py

```
fil = open("alder.txt", "r")
eldste_navn = "ingen"
maks_alder = -1
linje = fil.readline()
while linje != "":
    biter = linje.split()
    navn = biter[0]
    alder = int(biter[1])
    if alder > maks_alder:
        maks_alder = alder
        eldste_navn = navn
    linje = fil.readline()

print(eldste_navn)
fil.close()
```

FinnEldsteNavn. java

```java
import java.util.Scanner;
import java.io.File;
import java.io.FileNotFoundException;

class FinnEldsteNavn {
    public static void main(String[] args)
            throws FileNotFoundException {
        File fil = new File("alder.txt");
        Scanner sc = new Scanner(fil);
        String eldsteNavn = "ingen";
        int maksAlder = -1;
        while (sc.hasNextLine()) {
            String[] biter = sc.nextLine().split(" ");
            String navn = biter[0];     // Virker for alle navn?
            int alder = Integer.parseInt(biter[1]);
            if (alder > maksAlder) {
                maksAlder = alder;
                eldsteNavn = navn;
            }
        }
        System.out.println(eldsteNavn);
        sc.close();
    }
}
```

FinnEldsteNavn. java

```java
import java.util.Scanner;
import java.io.File;
import java.io.FileNotFoundException;

class FinnEldsteNavn {
    public static void main(String[] args) throws FileNotFoundException {
        File fil = new File("alder.txt");
        Scanner sc = new Scanner(fil);
        String eldsteNavn = "ingen";
        int maksAlder = -1;
        while (sc.hasNextLine()) {
            String[] biter = sc.nextLine().split(" ");
            String navn = biter[0];     // Virker for alle navn?
            int alder = Integer.parseInt(biter[1]);
            if (alder > maksAlder) {
                maksAlder = alder;
                eldsteNavn = navn;
            }
        }
        System.out.println(eldsteNavn);
        sc.close();
    }
}
```

Bedre: try – catch, se notat

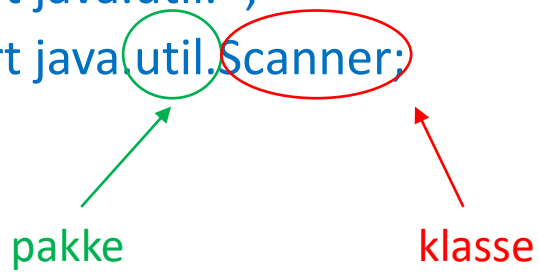
Kaller .close() når Scanner-objektet er en fil

Pakker og klasser, import

- All Java-kode hører til en klasse, som igjen hører til i en pakke (package)
  - Klasser uten pakke legges i en *unnamed* pakke
  - Disse er tilgjengelige for programmer i samme mappe (hvis filnavn = klassenavn)
- Pakken java.lang importeres alltid automatisk (her ligger for eksempel String-klassen)

=> trenger vanligvis ikke tenke på dette

- Andre pakker (eller enkeltklasser i pakker) må importeres eksplisitt
  - import java.util.*;
  - import java.util.Scanner;



Hvor ligger Scanner? Søk "java api 8 Scanner"

Tilgangskontroll: *Hvorfra* kan dette aksesseres?

Tilgangskontroll

- Alle klasser i programmet ditt ligger i samme mappe => de er i samme pakke

- Om du ikke angir tilgangsnivå er alt *package-private* (default tilgang)
  - Alle dine programmer i denne mappen har full tilgang til alt
  - Men: **main** må være **public** (fordi kjøresystemet til Java skal finne den)

- Innhold i klassen som du ønsker å beskytte (data og hjelpemetoder)
  - angis som **private** (kan bare aksesseres i klassen)

Innkapsling

- Mer å ta hensyn til og flere muligheter om programmene dine skal brukes av andre, eller gjenbrukes senere av deg selv – og når vi begynner å bruke arv

Feilhåndtering – "unntak"

- Feil som oppstår under kjøring genererer unntak (exceptions) i Java (og i Python!)
- Noen unntak *krever* Java at vi håndterer i programmene våre – blant annet ved åpning av filer
  - krever import av det aktuelle unntaket
  - unntak kan fanges og håndteres der det oppstår (med **`try..catch`**)
  - .. eller "kastes" (med **`throws`**) fra metoden tilbake til kallstedet

- Oftest beste håndtering:
  - fange unntaket der vi vet mest mulig om hva som gikk galt (med **`try … catch`**)
  - gi en tilpasset feilmelding

- Eksempler på **`try..catch`** i notatet "Enkel lesing og skriving i Java" på semestersiden.

- "Alt" om unntakshåndtering i Big Java, 7.4.

Håndtering av unntak med try .. catch

Eksempel fra Notatet: Lese og skrive i Java

```
class LagFil {
    public static void main(String[] args) {
        PrintWriter f = null;
        try {
            f = new PrintWriter("tall-til-10.txt");
        } catch (Exception e) {
            System.out.println("Kan ikke lage filen tall-til-10.txt.");
            System.exit(1);
        }
```

- Kan få en tilpasset melding/ håndtering
- Trenger ikke kjenne til/ håndtere i andre metoder

Håndtering av unntak med "throws"

Eksempel fra FinnEldsteNavn-programmet

```
.
.
class FinnEldsteNavn {
    public static void main(String[] args) throws FileNotFoundException {
        File fil = new File("alder.txt");
.
.
```

- Om det er greit at programmet stopper med en generell melding
- Må "kastes" av hver metode opp til main (mer senere)

Hjelpe-klasser for primitive typer

Integer, Float, Boolean, Character, ++ (med stor forbokstav!)

+ Noen ganger trenger vi at en verdi er et objekt (eksempler litt senere)

+ Disse klassene kan brukes som verktøykasser - med statiske metoder og konstanter som kan brukes uten å opprette et objekt av klassen. Eksempler fra **class Integer**:

  - Metoden **`parseInt`**

```
int alder = Integer.parseInt(biter[1]);      // konverterer String til int
```

  - Konstanten **`MAX_VALUE`**

```
int max = Integer.MAX_VALUE;                 //  høyeste verdi for integer
```

Konvertering

- Klassene Integer, Double, Boolean, Char, .. har metoder for å hente verdier av sine respektive primitive typer fra en String

```
String minStreng = " 1243 ";
minStreng = minStreng.trim();
int tall = Integer.parseInt(minStreng);
```

- NB: argumentet må *kun* inneholde verdien som skal konverteres. Whitespace kan fjernes med String-metoden trim()

Konvertering til String

- Klassene Integer, Double, Boolean, Char, .. har alle en metode toString som tar et argument av respektiv type og returnerer den som en String

```
int tall = 1234;
String minStreng = Integer.toString(tall);

double flyttall = 12.34;
String minStreng2 = Double.toString(flyttall);

System.out.println("minStreng = " + minStreng);
System.out.println("minStreng2 = " + minStreng2);
```

Konvertering til String på ulike måter

- Legge til en tom streng

```
double tall = 5.2;
String s = "" + tall;
```

- Bruke Double klassens (statiske) metode toString

```
double tall = 5.2;
String s = Double.toString(tall);
```

- Bruke String-klassens (statiske) metode valueOf

```
int heltall = 52;
String s1 = String.valueOf(heltall);
double flyttall = 5.2;
String s2 = String.valueOf(flyttall);
```

en metode for hver mulige argument-type: Overload

Bearbeiding av tekststrenger

- Klassen String inneholder mange nyttige metoder
  - `s.charAt(pos)` returner karakteren på posisjon pos i s
  - `s.equals(s2)` returnerer true hvis s og s2 er like tegn for tegn
  - `s.substring(3,5)` returnerer kopi av innhold i posisjon 3-4 som ny streng
  - `split`
  - `toLowerCase` og `toUpperCase`
  - `trim`

- Du finner alt om String i Java API (og det du trenger i Big Java)

- Java metoder kan "overloades" => finnes i flere varianter:
  - Samme navn og type, men ulike parametere.
  - Den som passer med argumentene i kallet, er den som blir utført.

Klassen Scanner

"scanner" et buffer – fra en fil, fra terminal eller fra en String

Klassen Scanner

- **Scanner** ligger i pakken **java.util**, som må importeres
- Oppretter et objekt av klassen **Scanner**
- Konstruktøren til Scanner tar et argument som angir hvor du skal lese fra
  - `System.in` (terminalen, referanse til et **InputStream**-objekt) eller
  - referanse til et objekt av klassen **File** (om du skal lese fra fil) eller
  - referanse til et **String**-objekt (om du skal lese fra en streng))
- To hovedmåter å lese inn data vha Scanner:
  - en hel linje (som så skal lagres eller prosesseres videre)
  - ett og ett "token" (på norsk "ord")

En linje av gangen som tekst

- I prinsippet det vi gjorde i Python i IN1000: Leser en og en linje inn i en String – trenger ikke vite noe om innholdet på linjen

- Scanner-metoder for å sjekke og deretter lese, en og en linje:
  - `public boolean hasNextLine()`
  - `public String nextLine()`

- Leser forbi linjeskift, returner alt *før* linjeskift som en String*

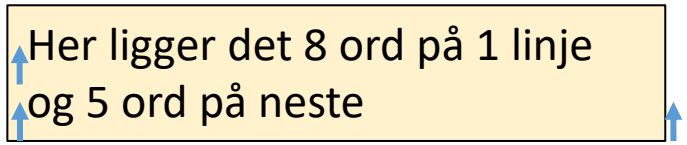
Ulikt Python
(som tok med linjeskift
når vi itererte over
linjene i en fil)

- Kan siden bruke linjen hel, eller dele opp (som i eksempel med alder lengre oppe)

* "Kaster" linjeskiftet

Scanner har en posisjonspeker

- Jobber seg gjennom et buffer (innhold fra terminalen, en fil eller en String)
- Har en *posisjonspeker* (pos) som holder rede på hva som er lest hittil



- *Leser* alltid fremover (og flytter da pos)

```
s = nextLine();       // s == "Her ligger det 8 ord på 1 linje"
t = nextLine();       // t == "og 5 ord på neste"
```

- Men kan også *sjekke* hva som kommer - *uten* å flytte pos

```
mer = hasNextLine()   // returnerer true eller false
                      // etter linjene over er mer == false
```

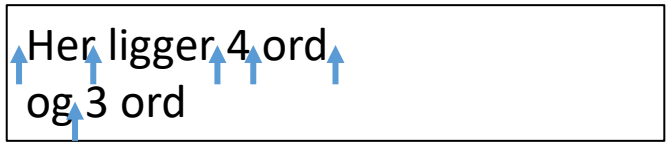
Å lese ett og ett "token" (ord)

- Et token er en sammenhengende tegnsekvens som avsluttes med whitespace* eller slutt på filen
- Kalles gjerne *ord* på norsk (kan være tall eller andre tegn)

- Viktige metoder for å behandle ord i klassen Scanner :

```
public boolean hasNext() // er det noen flere ord?
public String next()     // les og returner neste ord som en String
```

- `next` og `hasNext` hopper over innledende whitespace.



* whitespace: blank, tab, linjeskift

Å lese andre typer enn String

- Kan bruke Scanner-metoder for å teste neste ord:
  - `public boolean hasNextInt()` (sjekk om neste ord er int)
  - `public boolean hasNextDouble()` (sjekk om neste ord er double)

- Kan deretter lese med riktig metode, f eks:
  - `public int nextInt()` (les og returner neste ord som en int)
  - `public double nextDouble()` (les og returner neste ord som en double)

Navn og alder (I)

```
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();   // nextLine leser forbi linjeskift
        System.out.print(" Skriv alder: ");
        alder = minInn.nextInt();   // nextInt leser fra starten av neste linje

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

Alder og navn (II)

```
import java.util.*;
class LesFraTermAlderNavn {
    public static void main (String [ ] args) {
        int alder;
        String navn;
        Scanner minInn = new Scanner (System.in);
        System.out.print(" Skriv alder: ");
        alder = minInn.nextInt();   // nextInt leser ikke forbi linjeskift
        System.out.print(" Skriv navn: ");
        navn = minInn.nextLine();   // nextLine leser forbi 1. linjeskift
        // og finner det linjeskiftet som "lå igjen" etter nextInt .
        // I variabelen navn legges det dermed bare en tom streng

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

Lese ett og ett ord fra fil

- Vi bruker **Scanner** som før
- Må først opprette et objekt av klassen **File** (ligger i **java.io**)
- Dette sendes som argument til nytt **Scanner**-objekt

```
class LesFraFilTokens {
    public static void main(String[] attr)
            throws FileNotFoundException {
        File minFil = new File("Handleliste.txt");
        Scanner lesFil = new Scanner(minFil);
        while (lesFil.hasNext()) {
            String vare = lesFil.next();
            System.out.println(vare);
        }
    }
}
```

Trening i fillesing og Scanner:
Skriv finnEldsteNavn med innlesing ord for ord

Skrive til fil

Se notatet: "Lesing og skriving i Java" på semestersiden

Å skrive til fil

```java
import java.io.PrintWriter;
import java.io.FileNotFoundException;

class SkrivTilFil{
    public static void main (String[] args) throws FileNotFoundException {
        PrintWriter utfil = new PrintWriter("utfil.txt");
        utfil.println("Linje 1");
        utfil.close();
    }
}
```

Array

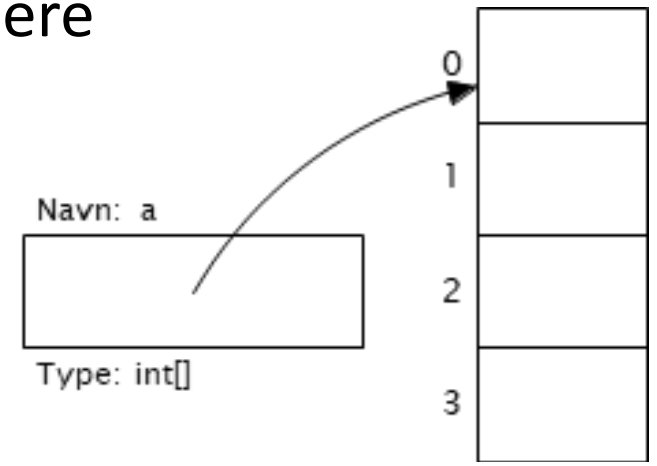
- I Java er array et alternativ til Pythons lister.
- En *array* er en datastruktur med mange elementer av samme type.

En array-variabel deklareres og arrayen opprettes slik:

```
int[] a = new int[4];
```

Array egenskaper

- Elementene er av samme type og lagres i etterfølgende celler i minnet.
- Dette gjør det til en effektiv struktur å aksessere

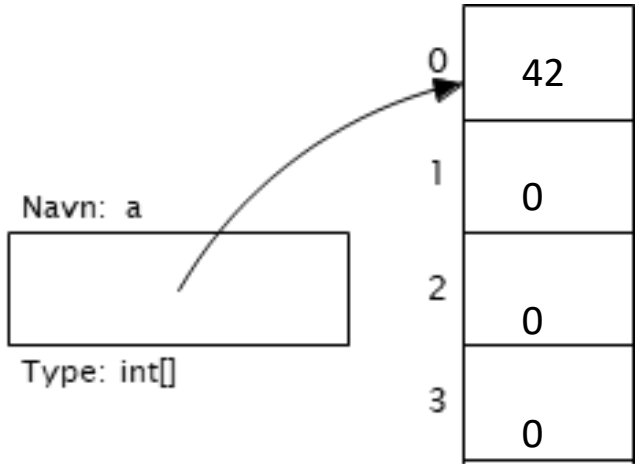


- Ikke en klasse - tilbyr ingen metoder!
- Ønsker vi en "smart" array må vi bruke en **ArrayList** (fra Java-biblioteket)

Array bruk

- Opprette, legge inn, lese

```
int[] a = new int[4];
a[0] = 42;
int sum = 0;
for (int i=0; i<a.length; i++) {
    sum += a[i];
}
```



Innhold i en array

- Alle elementer må ha samme type, bestemmes av deklarasjonen av variabelen

```
double[] priser;    // Ikke opprettet array ennå!
boolean[] resultater;
```

- Kan være referanser til objekter av samme klasse

```
String[] emnekoder = new String[3]; //Array opprettet, med null-verdier
Person[] deltakere;
```

Hva skjer her?

IndexErrorOutOfBounds

Gjennomløp av array: Summere tall

- Merk bruken av argumenter fra kommandolinjen – disse mottar vi som en String array
- Lengden av arrayen finner vi med .length
  (NB: Ingen parenteser/metodekall for arrayer!)

```
class SummerDisse2 {
    public static void main(String[] args) {



    }
}
```

Gjennomløp av array

```
class SummerDisse2 {
    public static void main(String[] args) {
        int sum = 0;
        for (int i=0; i<args.length; i++)
            sum += Integer.parseInt(args[i]);
        System.out.println(sum);
    }
}
```

```
M:\Ifi\Undervisning\IN1010 V2020\Forelesninger\uke2\Kode
>javac SummerDisse2.java

M:\Ifi\Undervisning\IN1010 V2020\Forelesninger\uke2\Kode
>java SummerDisse 10 10 10
30
```

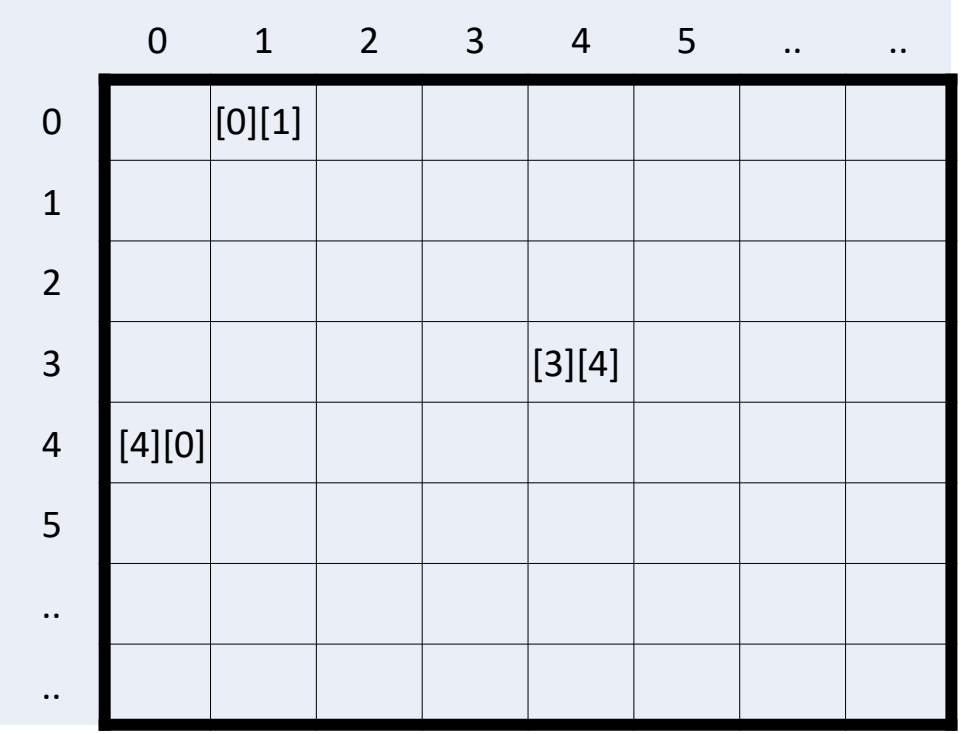
for-each/ enhanced/ forenklet for-løkke

- Ligner for-løkken i Python
- Går gjennom alle verdier i en beholder (her en array)

```
class SummerDisse {
    public static void main(String[] args) {
        int sum = 0;
        for (String str : args)
            sum += Integer.parseInt(str);
        System.out.println(sum);
    }
}
```

- Bruker "vanlig" for-løkke:
  - Hvis vi trenger indeks-verdien inne i løkken
  - Hvis vi skal endre verdier i arrayen

Repetisjon fra IN1000:
En tabell representert som en todimensjonal array

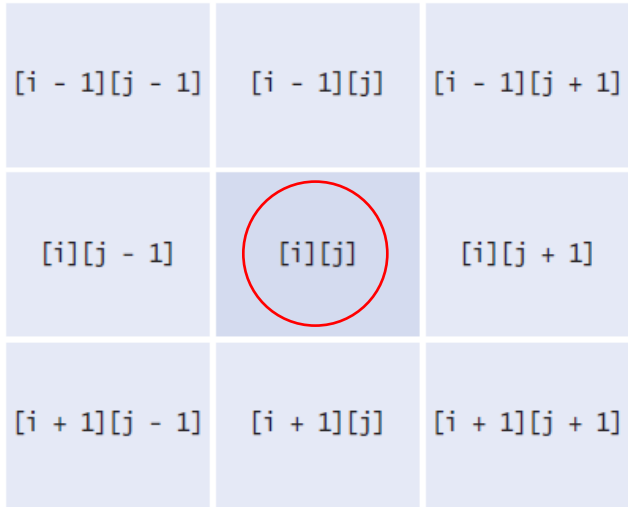


Her er elementene nummerert som om vi leser en bokside – dvs vi lar første indeks angi rad, og andre indeks angi kolonne.

IN1010 indekserer vi alltid todimensjonale strukturer slik.

Liste av lister: Indeksering

- En tabell med generiske indekser:



Flerdimensjonal array

fra Big Java, kapittel 6.7

	Gold	Silver	Bronze
Canada	1	0	1
China	1	1	0
Germany	0	0	1
Korea	1	0	0
Japan	0	1	1
Russia	0	1	1
United States	1	1	0

Figure 12 Figure Skating Medal Counts

Flerdimensjonal array

```
public class ToDArray {
    public static void main(String[] args) {
        final int LAND = 7;
        final int MEDALJER = 3;
        int[][] antall = new int[LAND][MEDALJER];

        antall[0][0]++;          // ny gullmedalje for land nr 0

        for (int landNr=0; landNr<LAND; landNr++) {
            for (int medNr=0; medNr<MEDALJER; medNr++) {
                System.out.print(antall[landNr][medNr] + "   ");
            }
            System.out.println();
        }
    }
}
```

Mer om arrayer, for-løkker, ...

Big Java – elektronisk tilgjengelig fra semestersiden under Pensum

- Forklaringer, tegninger, detaljer
- ... eksempler, tips inkludert "mønstre" for bruk

- Arrayer kapittel 6
  - Forenklet for-løkke i 6.2
- Løkker kapittel 4

Oppsummering

Bruk teorien **mens du programmerer** – og **programmer** mens du leser

- Slå opp klasser og metoder i Java 8 API
- Bruk lærebok eller Java tutorials (Java 8) for mer detaljerte forklaringer
- Oppskrifter på lese fra og skrive til fil og terminal i notat på semestersiden

- Exceptions gir nyttig og noen ganger nødvendig håndtering av feilsituasjoner

- Hjelpeklassene Integer, Double, Boolean, … har nyttige statiske metoder (og konstanter)
- Bli kjent med String-klassen for manipulasjon og konvertering
- Array er en effektiv, nyttig og veldig vanlig konstruksjon i mange språk
- Java har to ulike former for for-løkker