Fra Python til Java

Lese fra og skrive til terminal og fil

Uke 1 – dag 3 (IN1010-tidsregning)

Mandag 18. januar 2021

Beskjeder

- Obliger
  - Husk innlevering 0 (frist i kveld, nyttig å løse uansett)

- Grupper og gruppetimer (seminar):
  - Du kan fritt møte opp på digitale grupper (de som har en d etter navnet i timeplanen
  - Det avgjøres i dag om vi kan åpne fysisk undervisning på de fysiske gruppene
  - Timeplanen oppdateres for hver gruppe senest ila morgendagen (tirsdag)

Oversikt

- 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

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 nødvendige konsepter
  - ville gjort overgangen til Java tyngre

- Men:
  - IN1010 pensumet gir grunnlag for, og krever, dypere forståelse
  - Dere skal bruke (og skrive!) klasser i Java biblioteket
  - Begynn gjerne å bruke Java 8 API dokumentasjon allerede nå

Java versjoner

- IN1010 versjon = subsett av Java 8 (=1.8)
  - Alt vi trenger og bruker i IN1010
  - Fokus på grunnleggende mekanismer for videre påbygging
- Kjører i Java 8 (7) og nyere
  - Java 8 er fortsatt (eneste) i bruk mange steder, spesielt i eldre og store systemer
  - Stabilt og grundig testet/ rettet
  - Kjører i alle nyere versjoner (bakover-kompatibilitet)
  - Ifi servere har installert versjon 8
- Hvorfor bruke nyere versjoner (f eks 15)
  - Flere mekanismer tilgjengelig, inkl noen snarveier
  - Noen nye mekanismer ligner på Python
  - Noen verktøy (IDE'er) vil foreslå/ anta at du ønsker nyere versjoner
  - Du lærer å kjenne igjen og bruke konstruksjoner som er mulige/ vanlige i nyere systemer
- Vi anbefaler å holde seg til "IN1010 Java"
  - Forstå (tankesettet i) Java og hvordan språket skiller seg fra Python
  - Lettere å gå til nyere versjoner enn motsatt vei
  - Både du og retter kan alltid fullføre/ teste programmet ditt uansett maskin – inkl ifis servere ved behov.
  - Får vist det du skal kunne i IN1010

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)
  - ++

- Søk etter f eks "Java 8 API String" eller ".. Scanner"
  - Hvilken pakke en klasse ligger i (for import)
  - Dokumentasjon på konstruktører og metoder:
    - hva gjør de, argumenter og retur

- Java Tutorials om du ønsker å gå "til kilden" for de store (og små) sammenhengene i språket Java

- Big Java inneholder alt dere trenger å vite om dagens temaer (og endel mer) – og gir stort sett gode forklaringer på riktig nivå

Gå til menti.com og oppgi kode: 67 38 39 1

Oppgaver: Programmet **BrukCounter.java**

fra tirsdagens forelesning

```
class Counter {
     private static int nosCounters =0;
     private int value;
     public Counter() {
          value = 0;
          nosCounters ++;
      }
       public static int getNosCounters( ) {
            return nosCounters;
      }
      public void count(int ant) {
            value = value + ant;
       }
      public int getValue( ) {
            return value;
      }
}
```

```
class BrukCounter {
     public static void main (String [ ] arg) {
          Counter cntr1 = new Counter( );
          Counter cntr2 = new Counter( );
          cntr2.count(3);
      }
}
```

Lese fra fil

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 = 0
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()
```

finn_eldste_navn.py

```
fil = open("alder.txt", "r")
eldste_navn = "ingen"
maks_alder = 0
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

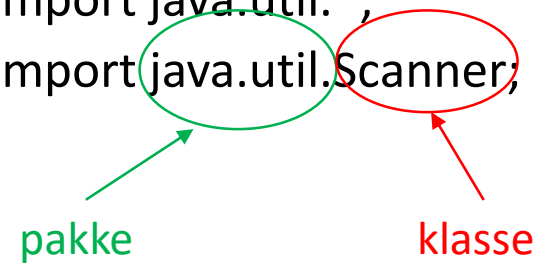
```
import java.util.Scanner;
import java.io.File;
import java.io.FileNotFoundException;

public 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 = 0;
        while (sc.hasNextLine()) {
            String[] biter = sc.nextLine().split(" ");
            String navn = biter[0];     // Virker dette for alle navn?
            int alder = Integer.parseInt(biter[1]);
            if (alder > maksAlder) {
                maksAlder = alder;
                eldsteNavn = navn;
            }
        }
        System.out.println(eldsteNavn);
        sc.close();
    }
}
```

NB: Kaller .close() på Scanner-objektet ved lesing fra fil

Pakker og klasser, import

- All Java-kode hører til en klasse, som igjen hører til i pakker (packages)
- Pakken java.lang importeres alltid automatisk (her ligger for eksempel String-klassen)
- Andre pakker (eller enkeltklasser i pakker) må importeres eksplisitt
  - import java.util.*;
  - import java.util.Scanner;



(klasser uten pakke legges i en default *unnamed* pakke – og er tilgjengelige i samme fil, eller når filnavn = Klassenavn.java og de ligger i samme mappe)

Tilgangskontroll

Så lenge alle klasser i programmet ditt ligger i samme mappe (=samme pakke):

- Gjør hovedklassen (med main) **public**
- Trenger ikke aksessmodifikator på andre klasser
- *Innholdet* i klassen er enten
  - **public** (kan aksesseres utenfra klassen)
  - **private** (kan bare aksesseres i klassen)

- Gjelder data (instansvariabler) og metoder
- Gjelder også static (klasse-)innhold

```
public class BilSalg {
    public static void main (String[] args) {
        int antallStein;

        // Operasjoner i main
        System.out.println("Bilsalg main");
    }
}
```

```
class Bil {
    private static int total = 0;
    private String eier;
    private int antForesporsler = 0;

    public Bil (String navn) {
        eier = navn;
    }

    public static int finnTotal ( ) {
        return total;
    }

    public void foresporsel ( ) {
        // innhold og andre metoder i klassen Bil
    }
}
```

Feilhåndtering – "exceptions"

- 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
  - .. eller "kastes" (med `throws`) fra metoden tilbake til kallstedet
  - .. hver metode opp til og med main må da kaste videre *eller* fange unntaket

- 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.

"Wrapper" klasser for primitive typer: Integer, Float, Boolean, Character, ++

- Tilbyr blant annet klassemetoder og konstanter som kan være nyttige (uten at man trenger opprette et objekt av klassen)

Eks:

- Konstant som angir høyeste verdi for en integer

```
int max = Integer.MAX_VALUE;
int min = Integer.MIN_VALUE;
```

Konvertering fra String

- Klassene Integer, Double, Boolean, Char, .. har metoder for å hente verdier av sine respektive 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()

Lek med 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 dokumentasjonen (søk etter "Java 8 API String")

- 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.

Konvertering til String, 3 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

```
double tall = 5.2;
String s = String.valueOf(tall);
```

en metode for hver mulige argument-type: Overload

Å lese data fra fil inn i et program

- Oppretter et objekt av klassen **Scanner**

- Argumentet angir hvor du skal lese fra
  - **System.in** (terminalen) eller
  - et objekt av klassen **File** (om du skal lese fra fil) eller
  - (en tekststreng (**String**) (om du skal lese fra en streng))

- **Scanner** ligger i pakken **java.util**, som må importeres

- To hovedmåter å lese inn data vha Scanner:
  - en hel linje som tekst (som så skal lagres eller prosesseres videre)
  - ett og ett "token" (vanligvis atskilt av blanke)

Å lese en linje av gangen som tekst

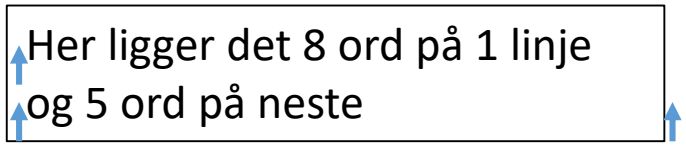
- I prinsippet det vi gjorde i Python i IN1000
- Leser en og en linje inn i en streng uten å bry oss om typer, eller om det er ett eller flere (eller ingen) ord
- Scanner-metoder for å lese linjevis:
  - `public boolean hasNextLine()`
  - `public String nextLine()`
- Leser forbi linjeskift, returner alt før linjeskift som en String*
- Kan siden bruke linjen hel, eller dele opp

Ulikt Python
(som tok med linjeskift)

* "Kaster" linjeskiftet

Scanner

- Leser et *buffer* – f eks med innholdet fra en fil, eller fra terminalen
- Har en *posisjonspeker* som husker hvor vi er nå
- *Leser* alltid fremover (og flytter da posisjonspekeren)
  - nextLine() // returnerer neste linje
- Men kan også *sjekke hva som kommer* uten å flytte pekeren
  - hasNextLine() // returnerer `true` eller `false`: Er det flere linjer?



FinnEldsteNavn – utdrag av main-metoden

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

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

NB: Kaller .close() på Scanner-objektet ved lesing fra fil

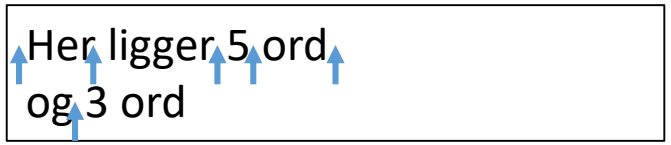
Å lese ett og ett "token" (ord)

- 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)

Test av Scanner metoder – hva skjer?

```
import java.util.Scanner;

public class LesTokens {
    public static void main(String[] args) {
      String test = "     1kjdfs\n834756 2.3";
      Scanner les = new Scanner(test);
      System.out.println(les.next());
      System.out.println(les.nextInt());
      System.out.println(les.hasNextInt());
      System.out.println(les.nextDouble());

    }
}
```

Utskrift:

1kjdfs

834756

false

2.3

Lese fra terminal – med en liten felle

```
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 + " aar" );
    }
}
```

- NB: nextint stopper ved linjeskift, nextLine leser bare til og med linjeskift – ville ikke fungert om vi byttet rekkefølge! Test gjerne.

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

```
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

Å skrive til fil

```
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

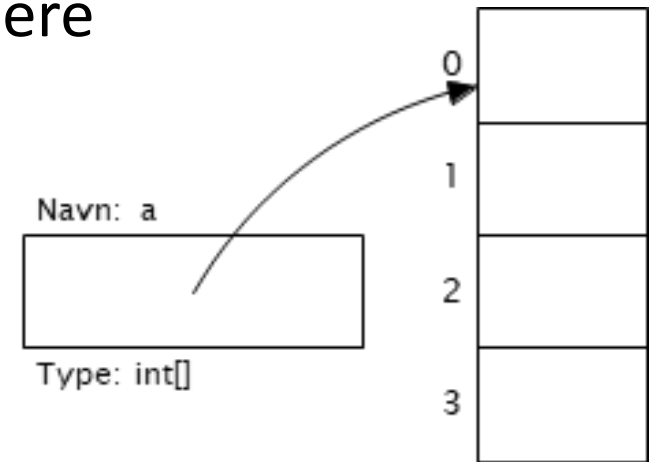
- I Java er array et alternativ til Pythons lister.
- En *array* er en datastruktur med mange elementer.

En array deklareres og 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

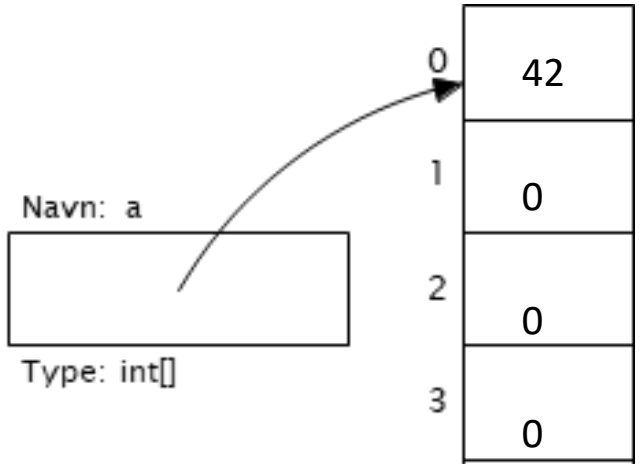


- Ikke en klasse - tilbyr ingen metoder!
- Ønsker vi en "smart" array må vi bruke en ArrayList (fra Java-biblioteket)

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;
```

- Elementer kan være andre array-er (flerdimensjonal)

```
int[][] tabell = { {1,2,3}, {3,6,9}}; // Initialisert
System.out.println(tabell[0][0]);
System.out.println(tabell[2][0]);
```

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 samling (her 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:
  - Hvis vi trenger indeks-verdien inne i løkken
  - Hvis vi skal endre verdier i arrayen

Mer om arrayer, for-løkker, ...

Big Java – elektronisk tilgjengelig fra semestersiden under Pensum

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

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

Oppsummering

- 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

- Primitive typer kan pakkes inn i objekter av tilsvarende klasse Integer, Double, Boolean, ..
- 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