Fra Python til Java

Del 2

Uke 2 (IN1010-tidsregning)
Tirsdag 21. januar 2020

Beskjeder

- Obliger
  - Husk innlevering 0
  - Meld dere på samretting for senere obliger
- Hvor kan man jobbe på Ifi? Se https://www.mn.uio.no/ifi/studier/ressurser/
- Varierer hvor fulle seminargruppene er, mulig å finne en annen hvis det er fullt

Oversikt

- Bruk av klasser fra Java API til å
  - lese fra terminal (Scanner)
  - lese fra fil (Scanner og File)
  - skrive til fil (PrintWriter)
- Tekststrenger (String), manipulasjon og konvertering
- Arrayer
- 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
  - Java dokumentasjonen er bedre egnet for oppslag
  - Bruk den til spørsmål om konkrete konstruksjoner/ mekanismer

Java dokumentasjon (NB: Java 8)

Java SE 8 API (Application Programmer Interface)

- Søk etter f eks "Java 8 api string" eller ".. scanner"
- Hvilken pakke en klasse ligger i (for import)
- Dokumentasjon på konstruktører og metoder:
  - hvordan virker de
  - hva returnerer de
  - hvilke argumenter tar de

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

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

Pakker og klasser

- All Java-kode hører til en klasse, som igjen er deklartert i pakker (package)
- 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
- (programmer uten package legges i en default *unnamed* pakke – og er tilgjengelige i samme fil, eller når filnavn = klassenavn.java og de ligger i samme mappe)

Feilhåndtering – "exceptions"

- Feil som oppstår under kjøring genererer unntak (exceptions) i Java
- Noen typer unntak krever Java at vi håndterer i programmene våre – blant annet ved åpning av filer
- I dag viser vi en "lettvint" måte å gjøre dette på:
  - vi "kaster" ("throws") unntaket fra metoden der det oppstår videre til kallstedet
  - må gjøres for hver metode opp til og med main
  - krever import av det aktuelle unntaket
- Mer grundig håndtering:
  - teste og fange (try – catch) unntaket der vi vet mest mulig om hva som gikk galt
  - gi en tilpasset feilmelding
- Eksempler på slik håndtering i notatet "Enkel lesing og skriving i Java" på semestersiden.
- "Alt" om unntakshåndtering i Big Java, 7.4.

"Wrapper" klasser: Integer, Float, Boolean, Character, ++

- Tilbyr 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;
```

- Objekter av en slik klasse kan lagre en verdi av "sin" type
- Representere en verdi i et objekt i stedet for i en primitiv variabel
  - Eks: Skal lagre en samling heltall i ArrayList eller HashMap (kommer)

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 trim()

- Alternativt: Lage et Scanner-objekt for strengen og bruke metodene der som vist på senere lysark

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, toUpperCase`	
`trim`	

- Du finner alt om String i Java dokumentasjonen (søk etter "Java 8 API <..>")

- 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 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

NB: Klasse-metode!

Å lese data 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:
  - ett og ett "token" (vanligvis atskilt av blanke)
  - en hel linje som tekst (som så skal lagres eller prosesseres videre)

Å lese ett og ett "token" (ord)

- Et *token* er en sammenhengende tegnsekvens som avsluttes med whitespace* eller slutt på filen/ strengen
- Kalles gjerne *ord* på norsk (kan være tall)
- Viktige metoder for å behandle ord i klassen Scanner :

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

- Scanner leser input sekvensielt fra starten og fremover, hopper over innledende whitespace. Har en innebygget posisjonspeker.

* 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());

    }
}
```

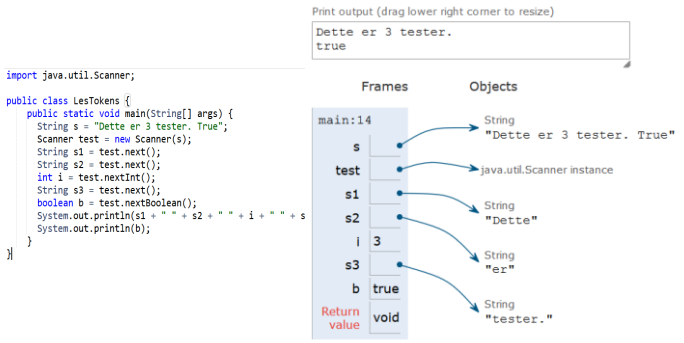
Utskrift:
1kjdfs
834756
false
2.3

En test: Lese ord

```
import java.util.Scanner;

public class LesTokens {
    public static void main(String[] args) {
      String s = "Dette er 3 tester. True";
      Scanner test = new Scanner(s);
      String s1 = test.next();
      String s2 = test.next();
      int i = test.nextInt();
      String s3 = test.next();
      boolean b = test.nextBoolean();
      System.out.println(s1 + " " + s2 + " " + i + " " + s3);
      System.out.println(b);
    }
}
```

Variable og utskrift etter kjøring (pythontutor.com)



Lese 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);
}
```


Lese fra fil: Hele programmet

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

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

            File minFil = new File("Handleliste.txt");
            Scanner lesfil = new Scanner(minFil);
            while (lesfil.hasNext()) {
                  String vare = lesfil.next();
                  System.out.println(vare);
            }
      }
```


Lese fra fil: Enkelt eksempel 2

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

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

            File minFil = new File("Handleliste.txt");
            Scanner lesfil = new Scanner(minFil);
            while (lesfil.hasNext()) {
                  String vare = lesfil.next();
                  Double pris = lesfil.nextDouble();
                  System.out.println(vare + ": " + pris);
            }
      }
```

Å lese en linje av gangen som tekst

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

* "Kaster" linjeskiftet

Lek med statistikk (Python-eksempel fra IN1000)

- Hva er navnet på eldste person i en fil?

```
odlaug 76
oluf 65
gunda 74
malfrid 80
godtfred 68
```

```
eldste_navn = "ingen"
maks_alder = 0
for linje in open("alder.txt"):
    biter = linje.split()
    navn = biter[0]
    alder = int(biter[1])
    if alder > maks_alder:
        maks_alder = alder
        eldste_navn = navn

print(eldste_navn)
```


Lek med statistikk (Java – linjevis med .split)

```
File innfil = new File("alder.txt");
Scanner lesfil = new Scanner(innfil);
while (lesfil.hasNextLine()) {
    String[] biter = lesfil.nextLine().split(" ");
    String navn = biter[0];
    int alder = Integer.parseInt(biter[1]);
    if (alder > maksAlder) {
        maksAlder = alder;
        eldsteNavn = navn;
    }
}
System.out.println(eldsteNavn);
```


Lek med statistikk (Java – linjevis med String-metoder)

```
// programlinjer som før utelatt
File innfil = new File("alder.txt");
Scanner lesfil = new Scanner(innfil);
while (lesfil.hasNextLine()) {
    String linje = lesfil.nextLine();
    int pos = 0;
    while (pos < linje.length()  && linje.charAt(pos) != ' ')
        pos++;
    String navn = linje.substring(0,pos);
    int alder = Integer.parseInt(linje.substring(pos).trim());
    if (alder > maksAlder) {
        maksAlder = alder;
        eldsteNavn = navn;
    }
}
System.out.println(eldsteNavn);
```

Å skrive til fil

- Bruker klassen PrintWriter, må importeres fra java.io
- Oppretter PrintWriter-objekt med filnavn som argument (PrintWriter trenger ikke et File-objekt)
- Skriver med samme metoder som til terminal:

```
println
print
printf          (med formattering)
```

- Må lukke filen for å sikre at alt lagres
- Hvis filen finnes fra før blir den overskrevet
- Må fange eller kaste unntak som ved lesing

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();
    }
}
```


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" );
    }
}
```

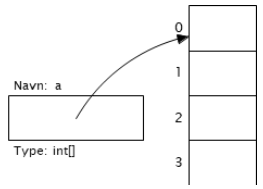
- NB: nextint stopper ved linjeskift, nextLine leser bare til og med linjeskift – ville ikke fungert om vi byttet rekkefølge!

Array

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

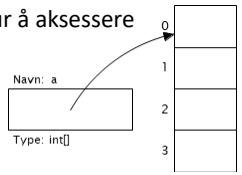
En array deklareres med:

```
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

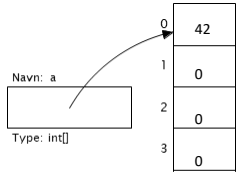


- Ikke en klasse - tilbyr ingen metoder!
- Ønsker vi en "smart" array må vi bruke en ArrayList (fra Java-biblioteket)

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
Person[] deltakere;
```

- Elementer kan være array-er (flerdimensjonal)

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

Verdien 1 skrives ut

Hva skjer her?

IndexErrorOutOfBounds
kun to arrayer i arrayet

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) {
        int sum = 0;
        for (int i=0; i<args.length; i++)
            sum += Integer.parseInt(args[i]);
        System.out.println(sum);
    }
}
```

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

- Som vi kjenner fra Python
- Går gjennom alle verdier i en samling (her array)
- Hvis vi trenger indeks bruker vi "vanlig" for-løkke

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


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
- 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