Velkommen til IN1010 Objektorientert programmering

IN1010

20.1.2026

Siri Moe Jensen

IN1010 forelesere våren 2026



Eyvind



Magnus



Dag



Siri

Plan for denne og neste uke

- I dag
  - 1. time: Praktisk informasjon og tips
  - 2. time: Java for deg som kan Python I
- Torsdag
  (Fellesøvelsen er forelesning denne uken)
  - Variabler, klasser og objekter i Java
- Neste tirsdag
  - Java for deg som kan Python II
- Neste torsdag: Livekoding

I løpet av denne uken bør du:

- Bli godt kjent med semestersiden
- Få god trening i enkel Java-programmering, kompilering og kjøring
- Løse og levere innlevering 0, åpne oblig 1
- Lage en plan for studiene dine

Etter de to første ukene bør du ha et godt grunnlag i Java:

- Kunne Java tilsvarende kap. 1-8 i pensumboken
- Kunne skrive programmer tilsvarende IN1000 pensum i Java

Quiz (alle hjelpemidler & samarbeid tillatt!)

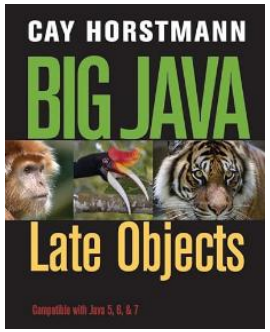
- Er Java nevnt i læringsmålene for IN1010 (“Etter å ha tatt IN1000…”)?
- Når er fristen for 1. obligatoriske innlevering?
- Hva kan gjøre at en besvarelse ikke blir rettet?
  - Den inneholder KI-generert kode
  - Besvarelsen er for svak
  - Mangler KI-deklarasjon
  - Leveres ikke innen 3 dager etter opprinnelig frist
- Hvilke(n) dato(er) er det eksamen i IN1010?
  - 21. januar
  - 20. mars
  - 1. april
  - 29. mai
  - 4. juni
- Hvor mange forsøk har man på en frivillig innlevering?
- Når er første gruppetime?
  - I morgen
  - Torsdag
  - Fredag
  - Mandag
- Hvem må møte på Pragmatisk programmering?
- Hva er Leganto?
  - En spesielt effektiv implementasjon av Java
  - Navnet på den som laget språket Java
  - En tjeneste som gir tilgang til pensum-ressurser
  - Et rom i Ole-Johan Dals hus
  - En PC-produsent

Fra emnesiden: Etter å ha tatt IN1010

- har du god oversikt over programmeringsspråket Java og du kan bruke det til å løse reelle problemer av middels størrelse
- behersker du avanserte objektorienterte mekanismer som subklasser, polymorfi og interface
- har du oversikt over noen grunnleggende datastrukturer (spesielt lenkete lister) og du kan programmere de viktigste operasjonene på dem
- kan du utvikle robuste og pålitelige programmer med godt objektorientert design, og du kan finne alternative løsninger for et gitt problem og vurdere fordeler og ulemper ved de ulike løsningene
- har du kunnskap om parallelle programmer med tråder og du kan benytte dette i enkle programmer
- kjenner du til hendelseshåndtering og kan skrive enkle programmer som håndterer hendelser

Pensum

- Pensum/litteratur i Leganto
  - Big Java, Late Objects (digital tilgang)
  - Notater



- I Leganto finner du stoff som dekker pensum grundig og kan hjelpe til med forståelse og helhet

- Prioritert stoff
  - Alt som er forelest
  - Alt som brukes i innleveringer (frivillige og obligatoriske)

  - Ressurser legges ut på semestersiden
    - Lysark
    - Opptak
    - Kode
    - Innleveringsoppgaver (løsningene må du skrive selv ☺)

Midtveiseksamen og eksamen

Midtveiseksamen:

- Tid: 20. mars (2 timer)
- Sted: Silurveien
- Hjelpemidler: **Ingen**
- Teller 25% av endelig karakter
- **Må bestås** for å kunne gå opp til eksamen

Avsluttende eksamen:

- Tid 29. mai (4 timer)
- Sted: Silurveien
- Hjelpemidler: Alle trykte og skrevne
- Teller 75% av endelig karakter

Frivillige og obligatoriske innleveringer

- Totalt 7 innleveringsoppgaver med tilbakemelding (0-6)
- Leveres i Devilry
- 2 av disse er obligatoriske(!):
  - Innlevering 1 må bestås for å kunne ta midtveiseksamen
  - Innlevering 5 (og midtveiseksamen) må bestås for å kunne ta avsluttende eksamen

- Alle innleveringer må ha en KI-deklarasjon i Devilry-kommentar
  - Blir ellers ikke rettet
  - Bruker opp et forsøk på obliger

Beskjeder, informasjon, spørsmål



i ?

- Viktige beskjeder gis på semestersiden
- Der finner du også
  - timeplan med ukens tema (under forelesninger)
  - lenke til alle ressurser fra blant annet forelesninger og fellesøvelser
  - innleveringsoppgaver med frister, krav og praktisk informasjon
  - Informasjon om undervisningstilbud i emnet
  - ++
- Bruk Discourse for faglige og praktiske spørsmål
  (beskjed med lenke kommer på semestersiden når den er oppe)
- Bruk gruppelæreren din – men husk at utenom timene har de egne studier!

Undervisningen i IN1010

- Forelesning (tirsdag): Nytt stoff – bør beherskes i løpet av uken
- Gruppetimer: Utdype og øke forståelsen
  - Pragmatisk programmering (fredag): Ekstra gruppetime om du synes vi går litt raskt frem
- Plenumsregning (torsdag): Repetisjon, eksempler, livekoding
- Selvstendig arbeid: Innleveringer, Trix, bøker, notater, foiler, forum...
  - Fredagsjava: Felles lab for individuell oppgaveløsing, gruppelærer tilstede

- Husk: *Normert* studietid er 13,5 timer hver uke med IN1010
- Programmering læres ved å programmere: Skriv masse kode - selv!
- Mer om tilbudene på semestersiden under Undervisningstilbud

Et mikrogram læringsteori

- Self-determination theory peker spesielt på noen psykologiske behov som grunnlag for *indre motivasjon, tilfredshet og velvære (!)*:
  - Autonomi
  - Kompetanse
  - Tilhørighet

- Hvordan kan vi – og du – bruke dette til å gjøre studietiden best mulig (og oppnå de resultatene du ønsker ☺)?

Så hva gjør du?

- Bruk undervisningstilbudene – ikke undervurder faste tider, sosiale forpliktelser og læring i fellesskap

- Følg med på egen progresjon, finn gode arbeidsformer
- Sett av den tiden *du* trenger for å følge undervisningen
- Legg til rette for å oppleve mestring – også når ting blir vanskelig

- Husk at du er her for å lære – ikke for å produsere kode (i seg selv)
- Studietiden er en enestående anledning til å fordype seg faglig og fokusere på egen læring

Kunstig intelligens og læring

- Kunstig intelligens er nyttig til mye – også i læringsprosesser. MEN
  - Læring kan ikke delegeres til andre, heller ikke KI
  - KI-tjenester har ikke kunnskap om ditt pensum, læringsmål eller forkunnskaper
  - Læring er – og må være – hardt arbeid!

- Malthe-Sørensen utvalgets* foreløpige notat (desember 2025) peker på en rekke muligheter og utfordringer med bruk av KI i utdanningen (lenke, se spesielt s 16-18). Noen stikkord på minus-siden:
  - Kan gi overfladisk læring og metakognitiv latskap
  - Kan gå på bekostning av god fagforståelse, dybdekunnskap og sosiale fellesskap

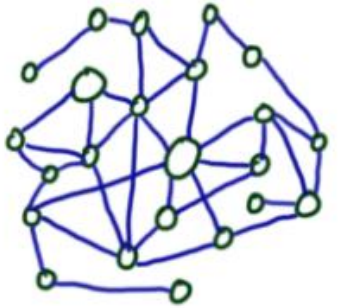
=> Dette utfordrer din evne til å vurdere hva og hvordan du «setter bort» av studiearbeid

*Utvalget om kunstig intelligens i høyere utdanning

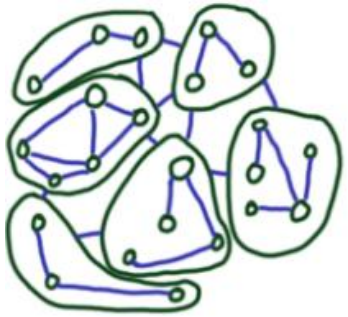
Hvorfor lære objektorientert programmering?

For å lage store og avanserte programmer som løser komplekse problemer må man kunne:

- Dele programmet i mindre komponenter som **løser hver sin del** av problemet
- Komponentene må **dele data/funksjonalitet** viktig for andre komponenter
- Komponentene må **skjule detaljer** som ikke er viktig for andre komponenter
- Komponentene bør være **gjenbrukbare** på ulike variasjoner av problemet



- Objektorientering er én – vanlig - måte å gjøre dette på, hvor komponentene er objekter



Merk: Mye av pensum er først nyttig når man skriver store programmer

Språk lages med ulike utgangspunkt

Java (James Gosling/ Sun Microsystems; 1996)

Mål: It must

- be simple, object-oriented, and familiar
- be robust and secure
- be architecture-neutral and portable
- execute with high performance
- interpreted, threaded, and dynamic

Python (Guido van Rossum; 1989)

Filosofi (utdrag av The Zen of Python, Tim Peters)

- Beautiful is better than ugly
- Explicit is better than implicit
- Simple is better than complex
- Complex is better than complicated
- Readability counts

Sentrale anvendelsesområder

Python

- Data Science & Data Analytics
- Kunstig intelligens (AI), dyp læring
- .. og mye mer

- Stadig vanligere:
  Introduksjonskurs i programmering

Java

- Enterprise Application utvikling
- Android app-utvikling
- ...og mye mer

- Nye versjoner er mer effektive, og støtter bl.a. enkel scripting og mer effektiv prosessering av (numeriske) data

Det finnes ikke ett "beste språk"

Mange faktorer avgjør

- Anvendelse (domene og oppgaver)
- Eksisterende arkitektur, kode og kompetanse
- Personlige preferanser
- Trender i business (vid forstand), samfunn og teknologi
- Teknologiutvikling, utvikling av språkene

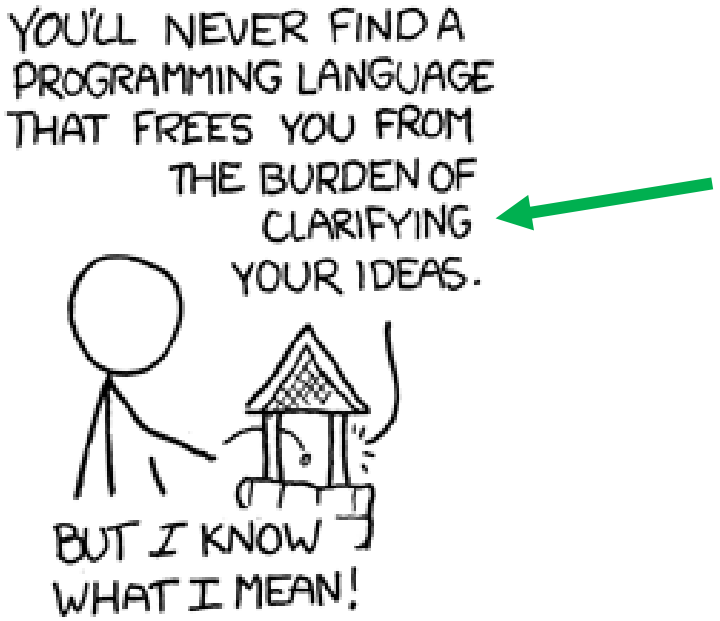
Kontinuerlig utvikling

Men:

- Java og Python er begge blant topp 3 på de fleste rangeringer av programmeringsspråk: Antall søk, antall arbeidsannonser, popularitet blant utviklere, etc.

Hvorfor **lære** flere programmeringsspråk?

- Python er et enkelt språk å komme i gang med, og relativt lett å lese
- Men det betyr også at endel sentrale aspekter – og noen utfordringer - kamufleres eller unngås
- I Java ser du mer direkte "hva som skjer" – og blir kjent med flere sentrale programmeringsmekanismer
- Du vil høyst sannsynlig ha nytte av disse – og andre språk
  - For bruk i videre studier eller jobb
  - For å vite hva som er generelt – og spesielt for ett språk
  - For å bli kjent med flere måter å "se" og løse oppgaver på
  - For å lære språk nummer 3 og 4 enda raskere ☺



Quiz (alle hjelpemidler & samarbeid tillatt!)

- Er Java nevnt i læringsmålene for IN1010 (“Etter å ha tatt IN1000…”)?
- Når er fristen for 1. obligatoriske innlevering?
- Hva kan gjøre at en besvarelse ikke blir rettet?
  - Den inneholder KI-generert kode
  - Besvarelsen er for svak
  - Mangler KI-deklarasjon
  - Leveres ikke innen 3 dager etter opprinnelig frist
- Hvilke(n) dato(er) er det eksamen i IN1010?
  - 21. januar
  - 20. mars
  - 1. april
  - 29. mai
  - 4. juni
- Hvor mange forsøk har man på en frivillig innlevering?
- Når er første gruppetime?
  - I morgen
  - Torsdag
  - Fredag
  - Mandag
- Hvem må møte på Pragmatisk programmering?
- Hva er Leganto?
  - En spesielt effektiv implementasjon av Java
  - Navnet på den som laget språket Java
  - En tjeneste som gir tilgang til pensum-ressurser
  - Et rom i Ole-Johan Dals hus
  - En PC-produsent

Python vs Java: Hovedforskjeller

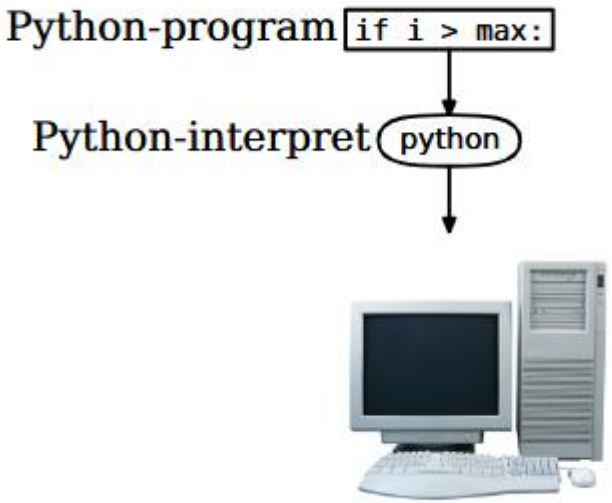
Python 

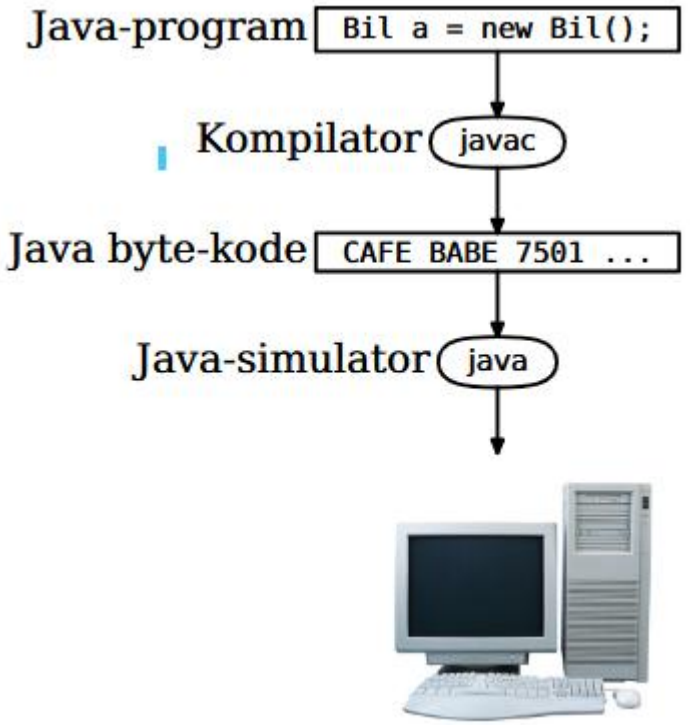
- Dynamisk typet
  - Variabler har ingen spesifisert type
  - En variabel kan inneholde ulike typer under kjøring
  - En funksjon/metode kan ta og returnere verdier av ulike typer
- Tolket
  - Oversetter kode til maskin-instruksjoner under kjøring
  - Type-feil oppdages under kjøring

Java 

- Statisk typet
  - Variable har en spesifisert type
  - Variable kan kun inneholde verdier av den typen
  - Metoder og parametere har spesifiserte typer
- Kompilering
  - Oversetter all kode til maskin-instruksjoner først
  - Kompilering gir et optimalisert, type-sjekket program klar for kjøring

Python vs Java: Fra kildekode til kjørende kode





Fordeler med dynamisk og statisk typing

- Dynamisk typing (Python)
  - Mindre å skrive
  - Mer fleksibelt (som listen `l = [ 2, "?", True ]`)

- Statisk typing (Java)
  - Programmereren kan kode inn hva han/hun tenker (`int i = 5;`)
  - Kommuniserer forventede typer til Java kompilator *og* menneskelige lesere.
  - Kompilatoren vil oppdage mange flere feil.
  - Kjøringen vil oftest gå raskere.

"Java og Python er objektorienterte språk"

- Hva betyr det?
  - ? Vi kan definere klasser og opprette objekter av disse *begge*
  - ? Vi kan bruke en objektorientert tankegang i modellering og design *begge*
  - ? Alle data er (egentlig) representert som objekter *Python*
  - ? All kode ligger i klasser, dvs vi må skrive minst én klasse *Java*

- Et Javaprogram har *all* kode inne i en eller flere klasser
  - Vi skal bruke *statiske* metoder i Java (finnes også i Python, men er ikke pensum i IN1000)
  - En statisk metode kan brukes selv om vi ikke har noen objekter av klassen – den hører ikke til ett objekt, men til klassen
  - Mer om dette på torsdag

Kjøring av Java-programmer

- Du må installere Java Development Kit (JDK)
- Du trenger to programmer for å kjøre et Java-program
  - **javac** oversetter (kompilerer) .java-filen(e) til en eller flere .class-filer
  - **java** utfører .class-filen(e)

```
class Hei {
public static void main(String[] arg) {
    System.out.println("Hei!");
}
```

```
$ javac Hei.java
$ java Hei
Hei!
$
```

Installasjon av Java og oppsett av VSCode: Se Teknisk info på semestersiden

Fra Python til Java

Gitt følgende Python program – hvordan skriver og kjører vi (nesten) samme program i Java?

```
ant_programmer = 100
print("Dette er mitt " + str(ant_programmer) + ". Python-program")
```

Eksempel i Java (kommentarer i senere slides)

```java
class Program {

    public static void main(String[] args) {

        int antallProgrammer;
        antallProgrammer = 1;

        System.out.println("Dette er mitt" + antallProgrammer + ". Java-program!");

        System.out.print("Denne linjen venter pa .println() kall eller programslutt"

    }
}
```

Innpakning

- I Java må alt ligge i klasser

- Utføring av programmet starter alltid i en metode som heter main – deklareres slik:

```
class Program {

    public static void main(String[] args) {

    }
}
```

- Filen må ha samme navn som klassen, her **Program.java**

Blokker og programsetninger

- I Java brukes krøllparenteser ({}) i stedet for innrykk.
- Alle Java-setninger avsluttes med semikolon (;).

```
if (antallProgrammer > 0) {
    System.out.println("Dette er mitt " + antallProgrammer + ". Java-program!");
}
```

- Linjeskift og innrykk betyr altså ikke noe i et Java-program:

```
        if (antallProgrammer
    > 0) {
            System.out.println
                    ("Dette er mitt " + antallProgrammer + ". Java-program!");
}
```

Angivelse av variabeltype

- Python har *dynamisk typing* der typen er knyttet til verdiene ; derfor kan en variabel tilordnes verdier av en hvilken som helst type.

- Java har *statisk typing* der både variablene og verdiene har type; variabler kan bare få verdier av den riktige typen.

- Derfor: *I Java må alle variabler deklareres*.

```
int antallProgrammer = 1;
double pris;
pris = 3.25;
boolean ferdig = true, ok;

System.out.println("Pris = " + pris + " og ferdig = " + ferdig);
```

```
>java Program
Pris = 3.25 og ferdig = true
```

Javas viktigste typer

int	heltall (dvs 0, 255, -3, ...)
double	flyttall (dvs 0.0, 1.5, -22.873, ..)
boolean	logisk verdi (dvs **true**, **false**)
char	enkelttegn (dvs **'1'**, **'a'**, **'*'**, ...)

NB: enkel apostrof (fnutt)

Dessuten finnes klassen

String	tekst (dvs **"abcd"**, **""**, **"$50"**, **"146"**, ..)

NB: dobbel apostrof

Utskrift

I java finnes to metoder for utskrift:

- System.out.print skriver ut parameteren (kun en parameter!)
  - Hvis det er flere elementer skjøter du dem sammen med +
  - Eller man kan kalle System.out.print flere ganger
- System.out.println gjør det samme, men legger til linjeskift.




```
System.out.print("Navn: " + "Donald Duck");
System.out.println();
```

```
System.out.println("Navn: " + "Donald Duck");
```

if-tester kan ha flere alternativer

```
if (antallProgrammer== 0) {
    System.out.println("Dette er mitt null'te Java-program!");
} else if (antallProgrammer == 1) {
    System.out.println("Dette er mitt forste Java-program!");
} else {
    System.out.println("Dette er mitt " + antallProgrammer + "Java-program!");
}
```

Legg merke til:

- Java har parenteser rundt if-testen (i stedet for kolon etter)
- Java har ingen **elif** – i stedet skriver vi **else if**

Løkker

While-løkker

```
while (antallProgrammer < 10000) {
    // Fortsett å programmere i Java
    antallProgrammer += 1;
}
```

For-løkker med teller (erstatter ... in range(10000) i Python)

```
for (int i = 0; i<10000; i++) {
    // Skriv ett program til i Java
}
```

Hvor mange feil finner du?

Feil.java

```
class Feil {
    public static void main (String[] args) {
        pris = 25;
        pris = 2*pris
        if pris > 200 {
            pris = 200;
        }
        else:
            pris = pris + 50;
        print("Pris: " + pris);
    }
}
```

Et komplett lite eksempel

FeilUtenFeil.java

```
1  class FeilUtenFeil {
2      public static void main (String[] args) {
3          int pris = 25;
4          pris = 2*pris;
5          if (pris > 200) {
6              pris = 200;
7          }
8          else {
9              pris = pris + 50;
10         }
11         System.out.println("Pris: " + pris);
12     }
13 }
```

$ javac FeilUtenFeil.java

$ java FeilUtenFeil

Lesbarhet teller - også i Java

```
public class KaffedrikkendeStudentMedDeltidsjobb {
    private int kopperDrukket = 0;
    public void drikkKoppMedKaffe() {
    kopperDrukket++;
    }
}
```

- Indenter tilsvarende Python selv om det 'bare' er for lesbarhet
  - Vanlig å spandere en egen linje til avsluttende krøllparentes
- Navngiving i Java:
  - Klasser skal ha stor forbokstav (CamelCase)
  - Metoder og variabler skal ha liten forbokstav (mixedCase)
  - Hvert nye ord etter det skal ha stor forbokstav (bruker ikke '_')

  - (Hva med balansen mellom lengde og informasjonsinnhold i navnene?)

Komme i gang

- Bli kjent med semestersiden
- Skriv, kompiler, korriger, kjør og korriger programmer:
  Trix, innlevering 0, gruppetimer
- Mer om grunnleggende Java denne uken og neste
  - Klasser og objektorientert programmering torsdag
  - Arrayer, metoder, input fra terminal og filhåndtering neste mandag

Effektivitet

- Hva er målet i IN1010?
  - Lære å programmere i Java, forstå konseptene og mekanismene som brukes
  - Solid grunnlag for videreutvikling i flere retninger
  - Ikke å utvikle spesialiserte programmer i et konkret produksjonmiljø

- Bruk din egen tid best mulig
  - Skriv mange programmer
  - Men sørg for at du lærer (evt trener på) noe for hvert program

  $\Rightarrow$ jo mindre og senere hjelp, desto mer læring – som sitter
  finn feil og svar på spørsmål selv før du bruker gruppelærer, løsningsforslag, ..

- Bruk gruppelærers tid best mulig
  - Følg undervisningen, bruk ressursene og gjør deg godt kjent med semestersiden
  - Bruk Pragmatisk Programmering for ekstra gruppetid ved behov
  - Spør i felles fora: Gruppetimer før lab, Discourse-kanaler ikke epost

Oppsummering

- Det finnes ikke noe beste språk for alle formål og situasjoner
- Det er en styrke å kunne flere språk
- Alle informatikere må regne med å lære flere språk
- Flere språk gir dypere forståelse

- Det er egentlig bare én måte å lære et nytt programmeringsspråk på:
  - les dokumentasjonen (lærebok, notater, Java Tutorials og Java 8 API)
  - prøv selv med mange eksempler (evt i pythontutor – velg Java som språk)
  - om ikke alt fungerer som forventet, gjenta

- Etter uke 2 regner vi med at du er "flytende" i Java