

# Velkommen til TDT4100 - Objekt-orientert programmering, våren 2024



Dag Olav Kjellemo  
Dag.O.Kjellemo@ntnu.no  
Kontor: 102 IT-bygget sydfløy,  
Rom 102

# Dagens forelesning

- Om faget
- Praktisk informasjon
- Læringsmål
  - Bli kjent med faget
  - Introduksjon til objekt-orientering
- Mens jeg prater... mys litt på <https://www.ntnu.no/wiki/display/tdt4100>  
(søk bare på nett etter *wiki tdt4100*)

# Litt om meg

Dag Olav Kjellemo

- Universitetslektor ved IDI
- Kontor 102 sydfløy, delt med Børge Haugset
- Har jobbet i industri, skole og akademia
- Faglige interesser: Programmering, matematikk
- Emneansvarlig i TDT4110, og noen mattefag.

# Også med i fagstaben:

Børge

- Eirik
- Mathea
- Andreas
- Oskar
- Vetle
- Viljan

# Objektorientering

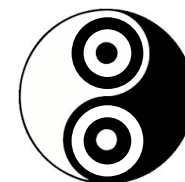
- I ITGK lærte dere programmering med fokus på prosedyrer
  - Der var koden fokusert rundt funksjoner
  - Python støtter også objektorientering, men uten å definere egne typer objekter

# Objektorientering

- OO har en annen tilnærmingssmåte
  - Fokus er å organisere koden rundt ‘objekter’
  - *Objekter* vil ha visse egenskaper avhengig av type
  - datatyper som har ulike egenskaper og oppførsel (som en liste som man kan legge til ting i)
  - Som et konseptuelt verktøy: Objekter representerer “ting”
  - Men det er også noe programvarefilosofi
  - Et objekt inneholder (oftest) data (attributter)
  - En *klasse* beskriver et objekts attributter og oppførsel (metoder)
  - Det passer svært godt for store systemer
- Det er en norsk oppfinnelse (Simula)! (som var det første programmeringsspråket som jeg hadde på universitetet)

# Læringsmål

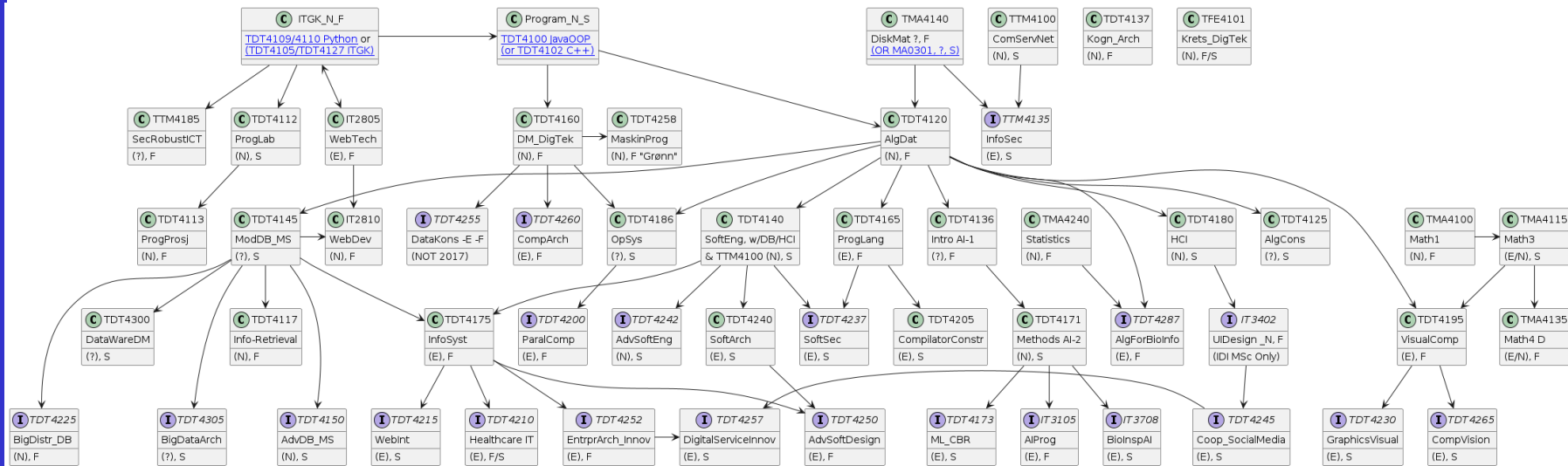
- Kunne **tenke** og **konstruere** objektorientert...  
fordi *objektorientert konstruksjon* er ledende innen moderne systemutvikling. Det er ikke det eneste, og man finner etter hvert ut hva som passer til hvilket problem.
- Kunne **skrive** objektorientert programkode...  
i *Java* fordi det er et fritt tilgjengelig, relativt enkelt og praktisk programmeringsspråk.
- Vi bruker Visual Studio Code
- VS Code er lettvekt sammenliknet med Eclipse og IntelliJ, men har fått en stor tilhengerskare. Mange (de fleste?) av dere kjenner det fra TDT4109.
- VS Code er fritt tilgjengelig for de tre hovedplattformene (linux, mac, win).
- Jeg har virkelig begynt å like VS Code! Masse utvidelser, justeringsmuligheter, og tettere integrasjon mot skall.



# men også...

- Etter dette kurset kommer dere til å ha et vesentlig utvidet syn på hva programmering er.
- Dere går fra å se på todimensjonale lister som en utfordring til å kunne beskrive gode strukturer for større programmer!
- Dere kommer til å skjønne hvordan man kan dele opp et problem, og la ulike deler av koden løse hver sin oppgave. På en mye mer strukturert måte enn bare funksjoner.
- Det kommer til å koste – dette har gått for å være et kurs som krever innsats – men det belønner med å gi en fenomenal modning som programvareutvikler.
- Start **nå**

# TDT4100- relasjon til andre emner!



<https://www.ntnu.no/wiki/display/idiemner/Course+Dependencies>

## Forventede forkunnskaper

[Faginnhold - Objektorientert programmering med Java - NTNU Wiki](#)

Vi forutsetter at dere har grunnleggende kunnskap om *prosedyreorientert* programmering (Python, Matlab e.l.)

- IT Grunnkurs eller tilsvarende (f.eks. IT1103)
- variabler, enkle datatyper og lister/tabeller, uttrykk, kontroll- og løkkekonstruksjoner og funksjoner/funksjonsskall

Erfaringsmessig er det stor variasjon i den enkeltes utgangspunkt

- Selv om vi i år starter med objektorientering med én gang, vil mange elementene i prosedyreorientert programmering i praksis gjennomgås/repeteres underveis i de første ukene
- Vi har laget noe materiale, for å gjøre overgangen enklere
- Dersom du vet at ditt grunnlag er litt svakt, så er det ekstra viktig å komme godt i gang med læringsarbeidet

# Læringsform

Forelesingene benyttes til å gi introduksjon til teorien, men også hvordan jobbe med praktisk programmering i VS Code

- Mest fokus på objektorientert tankegang, mindre på Java selv
- Bruk boka og eksterne ressurser for øving på Java-syntaks
- *En liten ting om GitHub Copilot og slikt og det med læring*

Øvingene brukes for at dere skal lære selv og opparbeide forståelse

- øvingsforelesninger vil presentere og gjennomgå øvingene, med fokus på *praktisk problemløsning* med øvingene som utgangspunkt
- *gjøres tilgjengelig én uke før de starter, offisielt ute xxxdag etter øvingsforelesning, og leveres typisk innen neste. Mer info kommer.*
- dere får velge stud.ass og saltid, men ikke alle kan ha samme stud.ass
- øvinger leveres på Blackboard, men *godkjennes på sal/digitalt med studass*

# Læringsform

- Vi underviser, du lærer?
  - Forelesninger og øvinger er en ramme for og hjelp til din læring, men jobben må du gjøre selv
  - Du bygger en *mental modell* av objektorientering og Java gjennom et samspill mellom refleksjon (tenke) og aksjon (gjøre)
  - Læringsjobben er **din!**
  - Målet mitt er å gi oversikt, forklare, eksemplifisere. Og, håper: inspirere

# Ressurser

- Blackboard er å betrakte som hjemmesiden til faget:
  - informasjon om organisering av faget, inkl. forelesninger og øvinger
  - pekere til ekstramateriale for emner som ikke er (godt) dekket av boka
  - kalenderen vil bli brukt aktivt
  - Husk at du kan slå av meldinger om oppdatering hvis du har appen...
- Fagstoff, oppgaver og andre ressurser finnes på vår wiki: <https://www.ntnu.no/wiki/display/tdt4100/>
- *[Piazza](#) som diskusjonsforum til spørsmål og svar?*  
(Lenken til 2022, ny vil opprettes)
- *Anbefalt lærebok: Big Java, early objects av Cay Horstmann, International Student Version*  
**Til eksamen vil dere få ta med bare denne boka. Det vil *ikke* være lov med notert fagstoff i boka, mens notater og annet som hjelper på navigasjon i boka, er tillatt. Dette kan bli sjekket på eksamen.**

<https://www.ntnu.no/wiki/display/tdt4100>

- Fag-wikien har stoff innen temaer som ikke dekkes godt (nok) av boka. Wikien er vinklet mot fagets læringsmål innen objekt-orientert programmering, boka har mer et Java-fokus.
- Ulike typer materiale:
  - oversikt og teori
  - kode-eksempler
  - oppgaver
- Øvingene bygger på oppgaver som ligger på fag-wikien
- Nettstedet er allerede fylt med stoff basert på hva vi tror dere trenger. Si fra hva som mangler!

# Verktøy vi skal bruke

- Ordentlige snekkere bruker ordentlige verktøy, det samme gjelder for programmerere
- IDE = Integrated Dev. Environment
- VS Code er en god all allround-IDE
  - støtter mange språk inkl. Java, Python, C++
  - har støtte for mange IT-verktøy og –prosesser
  - kan utvides med egne tillegg!
- Sjekk ut [installasjonsbeskrivelsen på wikien](#)
- Greit å beholde det en har når det funker. Ikke fikle med nye versjoner og utvidelser rett før eksamen...
- *Eksamen: Det blir windowsmaskiner eid av NTNU på eksamen. Øv.*
- Mer om VS Code-installasjon og –oppsett i første øvingstime
- NB! Første øvingstime onsdag 8:15-10 (byttet med forelesning)



Visual Studio Code

# Litt mer om VS Code

- Jeg har satt det opp litt mer enn det dere ser. Mys litt i 'extensions'
- Du kan gjøre nesten alt med tastaturet
  - Sjekk keyboard shortcuts
- control-shift-p er din venn
- VS Code starter smått, men den kan gjøre utrolig mye. Og mer med tillegg
  - Husk at for solide støttehjul kan gjøre det vanskeligere å lære å *faktisk* sykle. Ikke gå på med ChatGPT med en gang. Vær så snill.

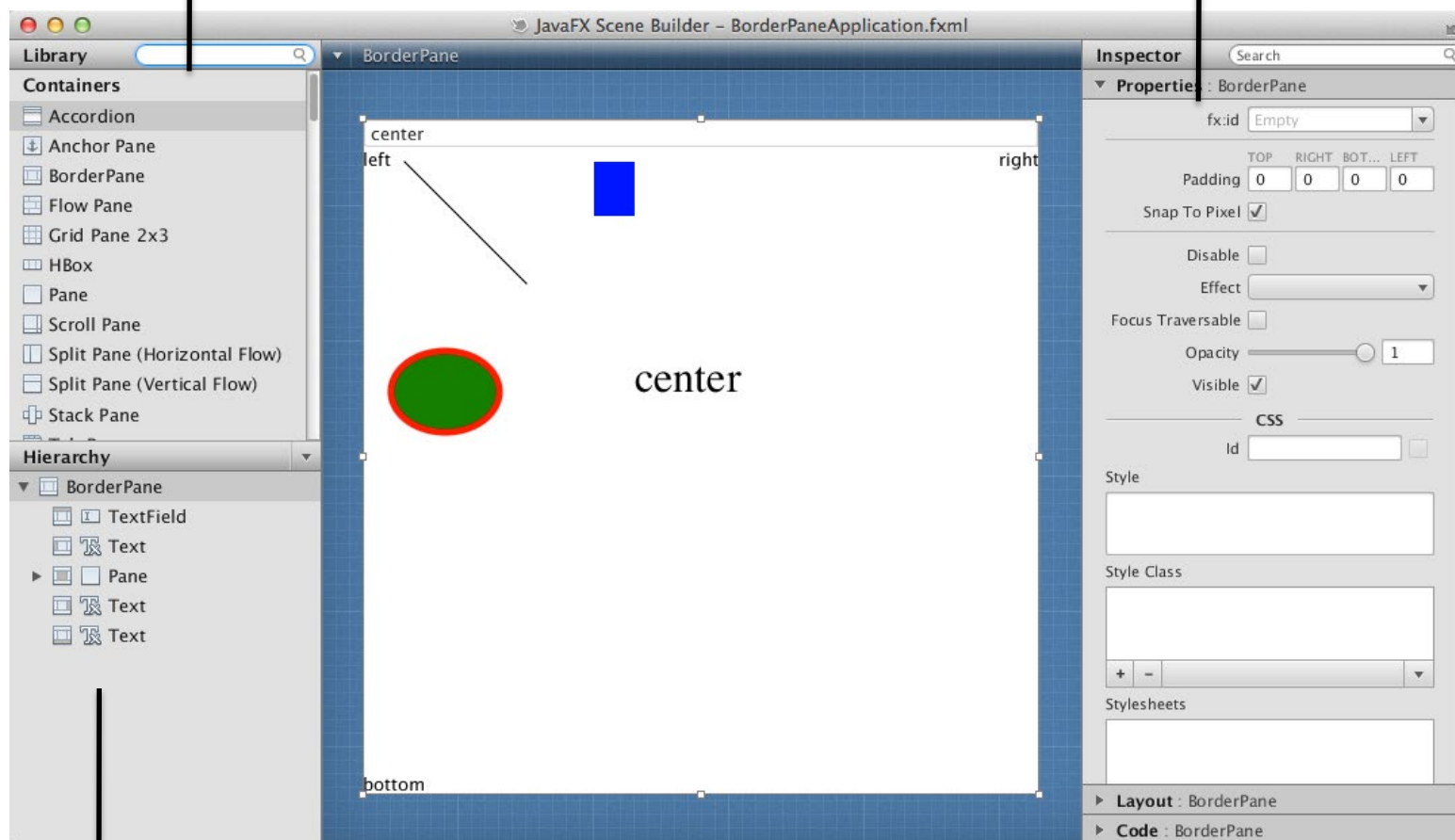


- All kode jeg skriver legges ut på [gitlab](https://gitlab.com)
- *Dere som tok TDT4109 før jul*
  - *Nå får dere igjen for git-øvingen!*
- Dette er også [der](#) stoff fra øvingsforelesninger, øvinger og løsningsforslag legges ut
- Ikke endre kode i *foreksempel*
- Dere finner hjelp på gitlab selv

# SceneBuilder – tegne JavaFX-apps

objektpalett

objektegenskaper



objekthierarki

# Referansegruppe

*Referansegruppa* har ekstra ansvar for å gi tilbakemeldinger underveis i semesteret.

Vi trenger frivillige, (minst) én fra hver studieretning

Vi ønsker primært ”vanlige” studenter som går på/ser forelesning og gjør øvingene.

Er ikke en kjempejobb. Tre møter, skrive en sluttrapport. Men jeg trenger innspill!

Send meg en epost på [dag.o.kjellemo@ntnu.no](mailto:dag.o.kjellemo@ntnu.no)

# Innhold i faget

- Implisitt repetisjon av prosedyreorientert programmering
- Klasser og instanser
- Innkapsling
- Objekt-interaksjon og interaksjonsdiagram
- Grensesnitt
- Testing (JUnit)
- Unntakshåndtering
- Arv
- Abstrakte metoder (og klasser) og polymorfi
- Klassediagram
- Java API og Collection-rammeverket
- Funksjonsobjekter
- Noen tema innen mer avansert objektorientering
- App-programmering med JavaFX/FXML

# Obligatorisk aktivitet

- Øvinger
  - For det meste praktiske oppgaver, men også noen teorispørsmål.
- Prosjekt – individuelt eller i par
  - Denne var tidligere karaktergivende Det vil nå kun være godkjent for å kunne ta eksamen, som en større øving.
  - Viktig for kunnskap. Og moro!

# Vurdering i faget

- Eksamen 100%
  - Her vil en få utdelt et prosjekt på eksamensdagen, og skal løse oppgaver i VS Code
  - På Sluppen, på *lånepc*. Mac-brukere: Lær dere alt-gr.
    - Sluppen og lånepc er en fin måte å sikre like vilkår

Vi planlegger å ha en prøve-eksamen, så dere får testet oppsett på sluppen.

# Foreløpig ukeplan

- Forelesninger med Opptak
  - mandager kl. 14.15-16.00 R7
  - onsdager kl. 08.15-10.00 F1
- Øvingsforelesninger med Opptak
  - Fredag 10:15 – 12:00 EL5
- Oppstart i uke 2 (denne)
  - Mandag: faginfo og litt om objektorientert tankegang
  - **Onsdag uke 2: Øvingsforelesning 8:15 - 10** installasjon og oppsett av VS Code, og repetisjon om prosedyreorientert programmering
  - Fredag, 2. forelesning: Mer om objektorientering, se litt på hvor vi skal
- De følgende lysark er en gjennomgang uke for uke av tema og øvingsoppgaver
  - NB! Forelesningsplanen vil endres – følg med på kalenderen på Blackboard

# Litt mer detalj

- De følgende sidene er ikke helt oppdatert etter rekkefølgen/oppdelingen vi bestemte i år, men det er ikke viktig.
- Det viktige er innholdet, ikke om det kommer i uke 5 eller 6. Se listen på BB.

# Foreløpig plan

Uke	Mandag	Tema Forelesning	Tema ØF
1	01.01.2024	Ingen forelesning	
2	08.01.2024	Introduksjon	Introduksjon
3	15.01.2024	Klasser, objekter	Introduksjon
4	22.01.2024	Innkapsling, tilstand, validering	Objekter, klasser, tilstand og oppførsel
5	29.01.2024	Datamodellering, assosiasjoner og diagrammer	Innkapsling og validering
6	05.02.2024	Assosiasjoner eksempler	Klasser og debugging
7	12.02.2024	Interface, comparator/comparable	Objektstrukturer (assosiasjoner)
8	19.02.2024	Iterator/Iterable, Optional, funksjonelle grensesnitt, lambda	Interface, comparable, iterable
9	26.02.2024	Lambda fortsettelse, fil, unntak	Interface, funksjonelle
10	04.03.2024	Delegering, observatør-observert	Observatør-observert og delegering
11	11.03.2024	Arv og abstrakte klasser	Arv og abstrakte klasser
12	18.03.2024	Testing og prosjekt. FXML!?	Prosjekt
14	01.04.2024	FXML og mer	Prosjekt
15	08.04.2024	Prosjekt	Prosjekt Q&A/Oppsummering
16	15.04.2024		
17	22.04.2024		

## Uke 2 –

# Repetisjon om prosedyreorientert programmering (POP) og intro til objektorientering (OO)

### POP

- Kontrollstrukturer og variabler
- Kjøring av programmer og tilstand
- Ark-modell for programutførelse

### OO – Objekter = tilstand og oppførsel

- hvordan tenke objektorientert
- data + kode = sant: klasser knytter data og kode sammen

### JShell (Java 11)

- Interaktiv utprøving av java-snutter

## Uke 3 – Klasser og instanser

### Mer om bruk av objekter

- eksempel-objekter
- utprøving av objekter med JShell

### Mer om enkle objekter

- dualiteten tilstand/felt og oppførsel/metoder
- instansiering med new og konstruktører
- diagrammer: objekter og objekt-tilstander

## Uke 4 – Innkapsling, tilstand, validering

### Innkapsling

- innkapsling
- synlighetsmodifikatorer og tilgangsmetoder
- konstruktører og instansiering
- overlasting

### Grensesnitt

- metodedeklarasjoner
- invarianter og regler for oppførsel

### Enhetstesting og Junit- rammeverket

- generell testmetodikk
- JUnit-testrammeverket

# Uke 5&6 – Datamodellering, Objektstrukturer

## Objektstrukturer

- enkle objektstrukturer og –referanser
- navigering i objektstrukturer
- bruk av *Collection*, *List* og *ArrayList*

## Objektstrukturer og objektinteraksjon

- sekvensdiagrammer

## Data-orienterte objekter

- klassediagrammer, assosiasjoner og multiplisitet

# Uke 7– Grensesnitt, comparable/comparator

Grensesnitt

- Felles egenskaper for klasser

Comparable/comparator

- Grensesnitt for sammenligning

# Uke 8 – Itererbare/iterasjoner, Optional, Funksjonelle grensesnitt

## Iterable/iterator grensesnitt

- Gå gjennom element for element

## Optional

- Når noe kanskje ikke finnes,
- Alternativ til bruk av **null**

## Funksjonelle grensesnitt

- Objekter som kan brukes som funksjoner

# Uke 9 – Lambda, filer, unntak

## Lambda

- Alternativ måte å definere funksjoner

## Filer

## Unntak

- Når noe ikke går etter plan A

# Uke 10 – Delegering, observatør-observert

## Delegering

- La et annet objekt gjøre jobben

## Observatør/ observert

- Gi og få beskjed om/lytte etter hendelser
- Eksempler fra GUI

# Uke 11 – Arv og abstrakte klasser

## Arv

- Utvide og redefinere klasser

## Abstrakte klasser

- Grunnklasser for utvidelser

# Uke 12 – Uke 12 – Testing, JavaFX og FXML, prosjekt

Testing

JavaFX,  
FXML

Prosjekt

- Grafiske grensesnitt
- Definisjonsspråk

- Lage app med grafisk grensesnitt

# Fra Python til Java

Hva er de største forskjellene?

Hvordan håndtere overgangen best mulig?

[https://www.ntnu.no/wiki/display/tdt4100/Java+  
vs+Python](https://www.ntnu.no/wiki/display/tdt4100/Java+vs+Python)

# Java vs. Python

- Ved kjøring er mye likt, forskjellene er i stor grad overfladiske (syntaktiske)
- Tre viktige forskjeller
  - “kjøring” av filer
  - (data)typer
  - blokker
- Kjøring
  - Python-filer kjøres fra toppen og ned og kan blande deklarasjoner og utførbare setninger og blokker
  - Java-filer inneholder primært deklarasjoner, og kode som kjøres ved opprettelse av objekter (én fil er en mal for objekter)
  - ALT må lages som klasser i Java
  - Java-filer kan ha en funksjon (main-metode) som kan kjøres, typisk for å få testet resten av koden i fila

# Java vs. Python

- Typer

- Java krever deklarasjoner med typer, både for variabler og funksjoner. Variabler blir da låst til å kun ha verdier av angitt *type*.

- variabel-deklarasjon

```
30 String s = "Java er gøy"; // s er nå en String-variabel
31 s = 1; // gir rød krøllstrek i editoren, siden 1 ikke er en String
```

- metode-deklarasjon

```
30 // returnerer en String, og tar inn et String-argument
31 String met(String arg) {
32     return arg.substring(arg.length() / 2);
33 }
```

- Ny java har mulighet til å deklarere med *var a = "streng"*

- Kompilatoren utleder hva slags type variabelen må være
- (Den ser hva slags type som står til høyre for likhetsteket)

# Java vs Python

- Siden alle variabler og metoder har typer, så vet Java og editoren/kompilatoren mer om hva som er lov og ikke
  - Kan si fra om hva som er feil bruk av variabler og metoder
  - Kan foreslå hva en kan gjøre med variabler og metoder
  - Det er faktisk så greit at Python har begynt nå tillater definisjon av parametertype og returtype for funksjoner...

*# Dette er konstruktøren i Python.*

```
def __init__(self, type:str, antall:int):
```

```
    self.type = type
```

```
    self.antall = antall
```

*# Disse metodene hører hjemme i Frukt. Slik ser det omtrent ut i Java også.*

```
def getAntall(self) -> int:
```

```
    return self.antall
```

# Java vs. Python

- Blokker

- Python angir blokker med : og innrykk  
`if a == 1:`  
 `print("a er 1")`
- Java bruker seg ikke om såkalt whitespace, men bruker { ... } for blokker  
`if (a == 1 ) {`  
 `System.out.println("a er 1");`  
`}`
- VS Code-editoren hjelper deg med innrykk iht. hvordan { og } er brukt, akkurat som for eksempel Thonny eller PyCharm gjorde for Python
- Det eksisterer konvensjoner for hvordan kode bør se ut, og editorer følger gjerne disse. Innrykk er ikke nødvendig, men gjør ting mer lesbart!

- Noen mindre forskjeller

- Java krever semikolon på slutten av enkle setninger
- Java krever oftere parenteser, f.eks. rundt testen i if og while  
`if (a == 1) ...`

# Java vs. Python

- TDT4100 er ikke et Java-kurs, men et kurs i objektorientert programmering med Java
- Det legges lite vekt på Java i seg selv, bruk boka og andre kilder for Java-syntaks osv.
- Fag-wikien inneholder egne sider om dette:
  - [Prosedyreorientert programmering](#) - generelt
  - [Java vs Python](#) – mer detaljert, men mye overlapp

# Ja – Java er noe annet enn Python

I starten må dere ‘bare bruke’ en del ting, uten at dere forstår hva det er.  
 Der Python bare er tenk/pseudokode og skriv, så må en ha en del ekstra  
 styr for å få til enkle ting.

Men når en kommer over det, da begynner en å kjenne hvordan dette  
 også hjelper en til å gjøre ting riktig.

# Andy Harris

- (How to begin thinking like a programmer – YT)
- Når du sliter, så kan det være fordi du ikke helt forstår problemet du skal løse ennå. Finn ut *hva* du skal gjøre før du finner ut av *hvordan* du skal gjøre det.
- Du lager ikke kommentarer for å forklare koden din til deg selv om to uker eller andre, du lager kode for å forklare kommentarene til maskina.
- Det er lov å erkjenne at kommentarer er superfine å ha.
- Algoritmedelen er den vanskelige, selve programmeringen kan en slå opp. Når en vet hva en skal slå opp, sant!
- Feiling er topp! Begynn å debugge. Fortalte du hva maskina skulle gjøre på en feil måte, eller fortalte du feil ting til maskina på en riktig måte. Det er oftest et algoritmeproblem. Kanskje du løser det ved å ta oppvasken og tenke!
- Det er ikke så galt å kaste kode. Det er algoritmetenkingen som tar tid.
- Dette er ingen sannheter – men tenkte det var ting å tenke over

• Det som er sant er at denne siden så helt grusom ut. Men det er en annen sak...

# VS Code

- Masse fine utvidelser
  - Java Code Generators: for å raskt lage typiske metoder
- Debuggeren
- Kan åpne to instanser samtidig

# Lysten på noen nettressurser?

- <https://java-programming.mooc.fi/>

Kort forklaring/eksempel på objekt-orientert:

- [https://youtu.be/KyTUN6\\_Z9TM?si=D\\_b2CKoLEWorOD\\_FA](https://youtu.be/KyTUN6_Z9TM?si=D_b2CKoLEWorOD_FA)

# Hva er objektorientering?

- Dette kommer mer neste forelesning
- Sannheten er at du har brukt massevis av objekter allerede
- Nesten alt du har brukt i Python er egentlig objekter. Lister, dictionaries...
- Enn om en kunne laget sine egne slike strukturer, som passet for å løse akkurat din oppgave?

# Andre ting

- Organisering av koden
- «Bygging», dvs. få nødvendige ressurser på plass for å bygge den ferdige koden.
- Versjonskontroll (Git)
- Testing av kode, spesielt enhetstesting