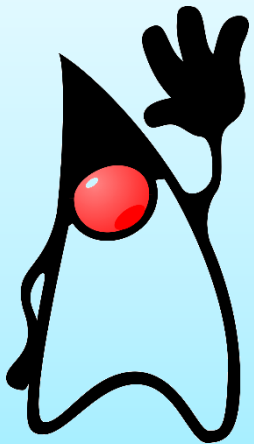


TDDD78 + TDDE30

Objektorienterad programmering och Java



Kursinformation



Examinator, kursledare: Jonas Kvarnström



```
**** COMMODORE 64 BASIC V2 ****
64K RAM SYSTEM 38911 BASIC BYTES FREE
READY.
```



Fråga – kommentera – avbryt!



Inte inspelat!



Hämmer interaktivitet / diskussion



Kursinfo

Rast

Java del 1

Evaluuate TDDE24?

Webreg?

Mål

**Objektorienterad
programmering**

ERT MÅL

*Kunskap och färdigheter
inom
programutveckling*

Översikt – ni behöver inte förstå ännu!

Utan OO: datastrukturer för lagring,
som "manipuleras utifrån" av funktioner

```
daysSinceToday(date)  
isLeapYear(date)  
add(date, days)
```

Date-struktur

```
year: 2027  
month: 12  
day: 31
```

Med OO har objekt både datalagring
och egen funktionalitet, egna metoder

- ➔ **Fundamental** princip:
"Objektet bestämmer över sig själv"
(ingen manipulerar utifrån)
- ➔ Varför? **Grunden** för begrepp som
ärvning, overriding, polymorfism, inkapsling,
...

Date-objekt

```
year: 2027  
month: 12  
day: 31
```

```
daysSinceToday()  
isLeapYear()  
add(days)
```

I TDDE23-24 används klasser och objekt i Python!

Definierat "vanliga" funktioner...

```
def factorial(n):  
    ...  
x = factorial(10)
```

...men använt många objekt/metoder

```
mylist.append(item)
```

Listan är ett objekt

Vi ber listan
att göra något åt oss:
Lägga till ett värde i slutet

Dags att lära sig mer om detta!

**Objektorienterad
programmering**

**Konkret
OO-språk:
Java**



**Generella
begrepp:**

**Stark typning,
pekare,
kontrakt,
...**

MÅL

*Kunskap och färdigheter
inom
programutveckling*

**Generella
färdigheter:**

**Verktyg för
utveckling**

Programmeringsvana!

Viktigt i sig själv och som en bas att bygga på

Medel för att uppnå målen

Vad kommer först?

Ibland föreläsning först
➔ läs + experimentera

Ibland labba först
➔ sätt i ett sammanhang
på föreläsningar

Lyssna och fråga

Föreläsningar

*Ramar för förståelsen,
övergripande tankar och idéer*

Inga seminarier?

Kurser har olika upplägg!

Använda resurserna bäst
givet kursens mål,
studenters erfarenhet,
...



Läsa

Kursbok, instruktioner,
material på nätet, ...

*Mer detaljer – viktiga,
och läses bäst i egen takt*

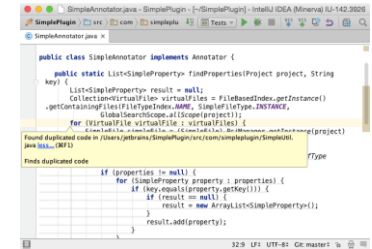
MEDEL:

Hur ska ni uppnå era mål?

Utföra

Labbar och projekt,
med *integrerat studiematerial*

*Erfarenhet,
förståelse för vad man kan –
och vad man behöver öva på*



Föreläsningar

Introduktion

Kursintro, Java för Python-programmerare, typning, ...

Utan objektorientering: Ändå mycket nytt...

Objektorienteringens grunder

Principer och begrepp: Klasser, metoder, fält, konstruktorer, ...

Konkret användning av OO i Java

Viktiga begrepp, med och utan OO

Kontrakt, åtkomst och inkapsling, lagring och livstid,
pekare och komposition, identitet och likhet, ...

Hierarkier och arv

Typhierarkier med gränssnitt (*interface*), polymorfism,
statisk/dynamisk bindning, abstrakta klasser, *overriding*, ...

Grafiska gränssnitt

med *komponenter* och
händelsehantering

...

Projektinfo

Att välja
projekt, ...

- Föreläsningar ska alltid utgå från **förkunskapskraven**

Vad ska ni redan kunna?

Grundläggande begrepp inom programmering

*Konkreta programmeringskunskaper i Python, **motsvarande**
t.ex. kurserna Funktionell och imperativ programmering del 1 och del 2.*



Vad bygger (närmast) på den här kursen?

OOP

Begrepp

Java

Mer programmeringsvana!

U:
TDDD80 (11hp)
Flutter, Dart
Mobila och sociala applikationer

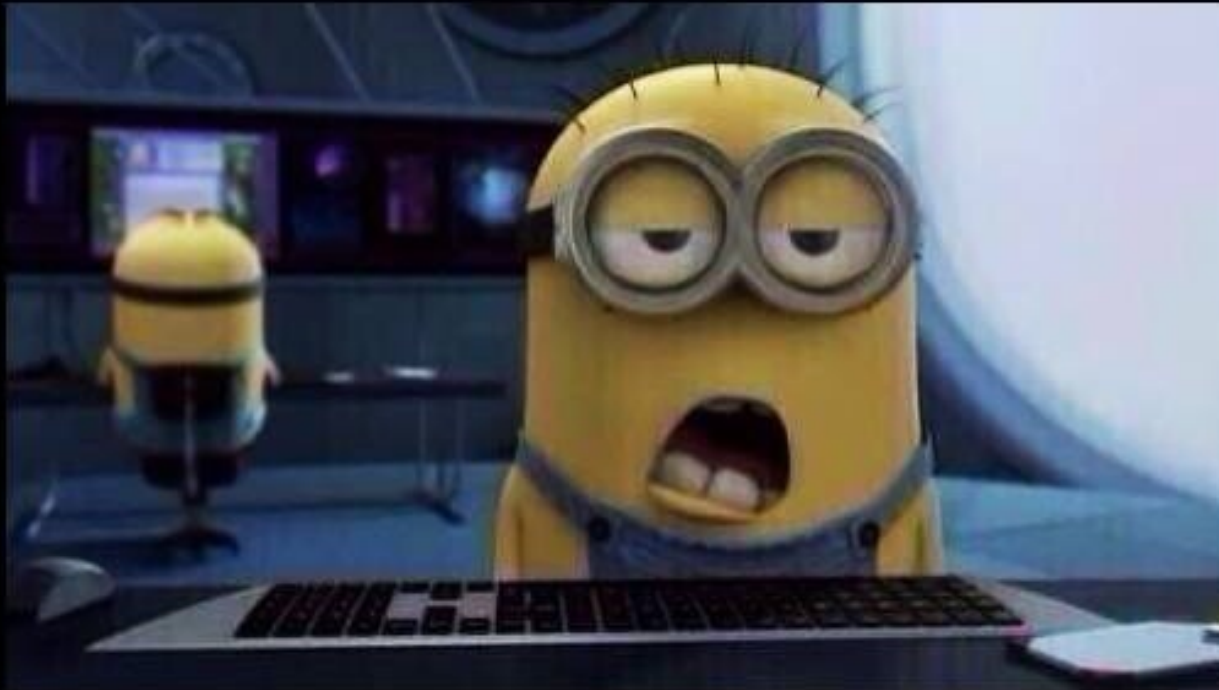
D, U: TDDD86 (11 hp)

Datastrukturer, algoritmer och programmeringsparadigm / C++

Vad kan ni egentligen?



Kan du redan det mesta?



**Då är föreläsningarna inte *helt* anpassade till dig...
Men det kan ändå finnas en del matnyttigt**

Förutsättningar för föreläsningar (4)

**Gå genom bilderna i förväg
Se vad vi ska diskutera**

**Jag vill hellre läsa
om det här**

**Det här lär jag mig bättre
genom egna experiment**

**Här vill jag friska upp minnet
och lyssna lite**

Här fanns det en hel del nytt!

- Två alternativ:

**Färre föreläsningsbilder,
prata fritt från korta punkter**

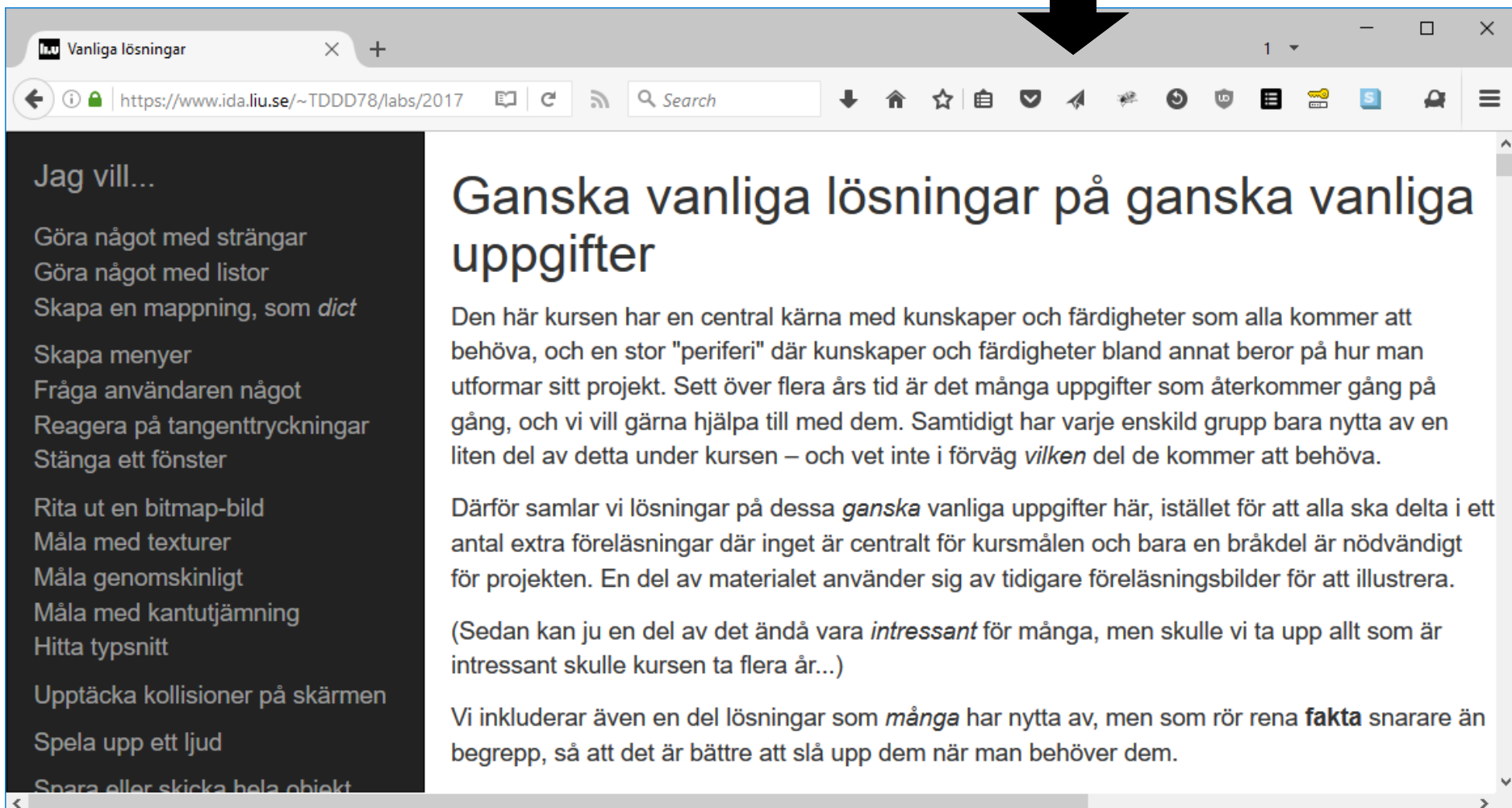
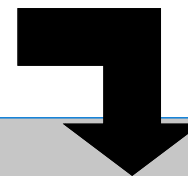
- (+) Känns ”spontant”
- (+) Ingen går i förväg
- (-) Lätt att missa vad som sägs
- (-) Kan glömma: Anteckna hela tiden

**Många föreläsningsbilder,
alla relevanta fakta nedskrivna**

- (+) Bra att både höra och läsa
- (+) Referensmaterial senare
- (-) Kan kännas som att läsa innantill



- Kan vara intressant ibland, men inte centralt för målen?
- Rena **fakta** – tillgängliga metoder, ...?



Vanliga lösningar

https://www.ida.liu.se/~TDDD78/labs/2017

Jag vill...

- Göra något med strängar
- Göra något med listor
- Skapa en mappning, som *dict*
- Skapa menyer
- Fråga användaren något
- Reagera på tangenttryckningar
- Stänga ett fönster
- Rita ut en bitmap-bild
- Måla med texturer
- Måla genomskinligt
- Måla med kantutjämning
- Hitta typsnitt
- Upptäcka kollisioner på skärmen
- Spela upp ett ljud
- Spara eller skicka hela objekt

Ganska vanliga lösningar på ganska vanliga uppgifter

Den här kursen har en central kärna med kunskaper och färdigheter som alla kommer att behöva, och en stor "periferi" där kunskaper och färdigheter bland annat beror på hur man utformar sitt projekt. Sett över flera års tid är det många uppgifter som återkommer gång på gång, och vi vill gärna hjälpa till med dem. Samtidigt har varje enskild grupp bara nytta av en liten del av detta under kursen – och vet inte i förväg *vilken* del de kommer att behöva.

Därför samlar vi lösningar på dessa *ganska* vanliga uppgifter här, istället för att alla ska delta i ett antal extra föreläsningar där inget är centralt för kursmålen och bara en bråkdel är nödvändigt för projekten. En del av materialet använder sig av tidigare föreläsningsbilder för att illustrera.

(Sedan kan ju en del av det ändå vara *intressant* för många, men skulle vi ta upp allt som är intressant skulle kursen ta flera år...)

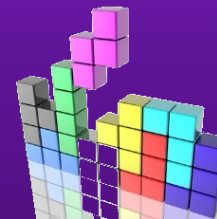
Vi inkluderar även en del lösningar som *många* har nytta av, men som rör rena **fakta** snarare än begrepp, så att det är bättre att slå upp dem när man behöver dem.

Labbar och projekt

Steg 1: Era första(?) Java-program; *individuellt*

Fokus: Lär in rätt sätt, inte fel (svårt att bli av med)

Både mindre uppgifter
och ett större program



Labb 1

Labb 2

Labb 3

Labb 4

Moment
LABx
i LADOK

3 hp,
betyg G

Steg 2a: Breddning och fördjupning; *individuellt*

Tetris-4, Tetris-5

Behövs för betyg 4/5 på kursen

Moment
PRAx
i LADOK

Steg 2b: Eget projekt + projektrapport

Enskilt *eller* i par (hitta egen partner)
Fria händer
Föreläsning + anmälan senare!

3 hp,
betyg
3-4-5

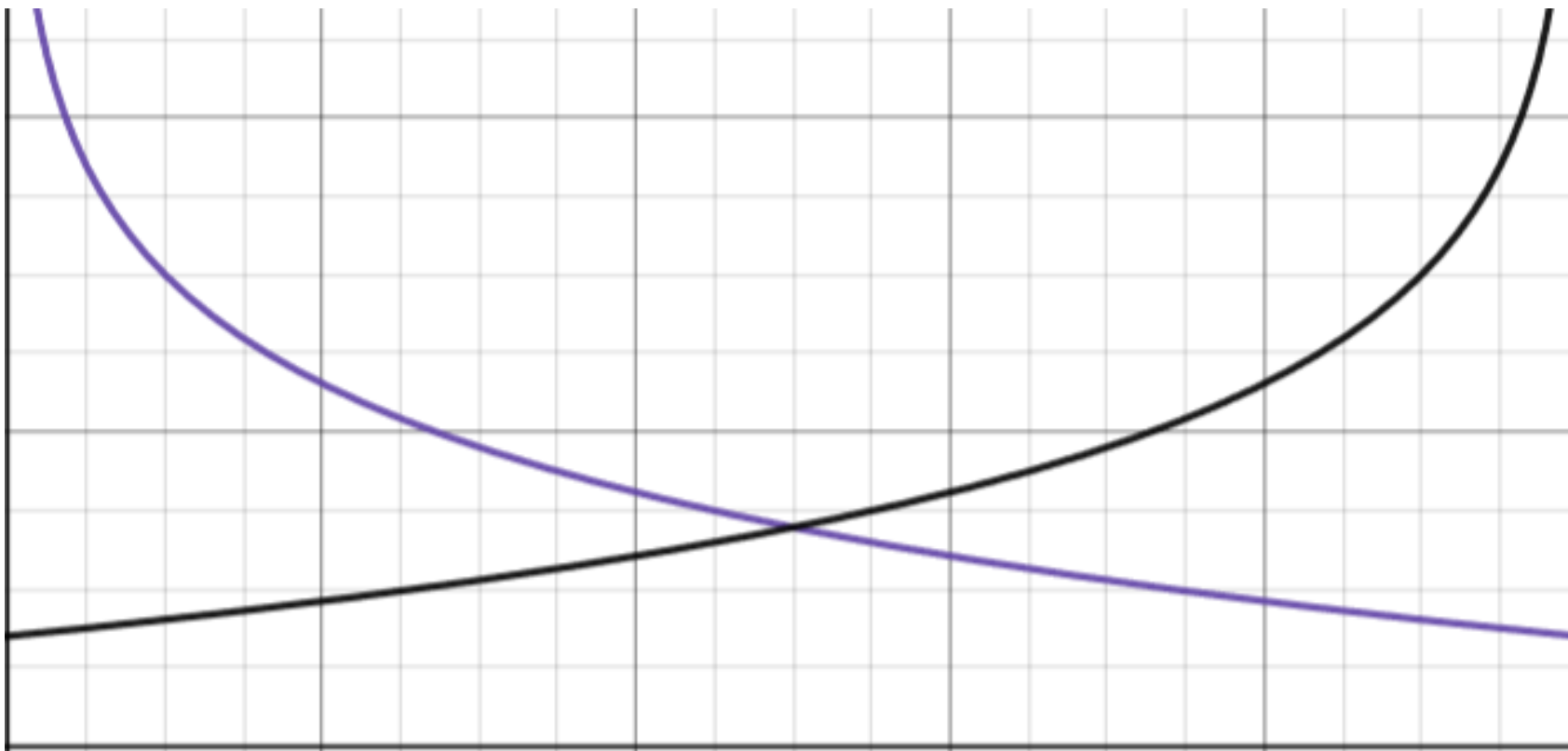
TDDE30: Rapportens språk och struktur (1 hp)

UPG1, G

Vecka		TDDE30		TDDD78		TDDD78 alt. 2		Vecka
4								4
5		Labb 1-2		Labb 1-2		Labb 1-2		5
6				Labb 3-5		Labb 3-5		6
7		Labb 3-5				Labb 3-5		7
8								8
9				Projekt				9
10		Projekt				Projekt		10
11								11
12	Tentaperioder							12
13								13
14								14
15				Klar tidigt?		Se projekt som tenta, jobba i tenta-p		15
16								16
17								17
18		Projekt		Många fler detaljer på https://www.ida.liu.se/~TDDD78/labs/2025/timeline.shtml				18
19				Läs även om deadlines där!				19
20								20
21								21
22	Tentaperioder							22
23								23

■ Avvägning:

- Ju hårdare deadlines, desto färre klarar av dem
- Ju lösare deadlines, desto fler blir inte klara inom kurstiden
 - Inte ovanligt att studenter ber om **fler** deadlines!



Deadlines (2)

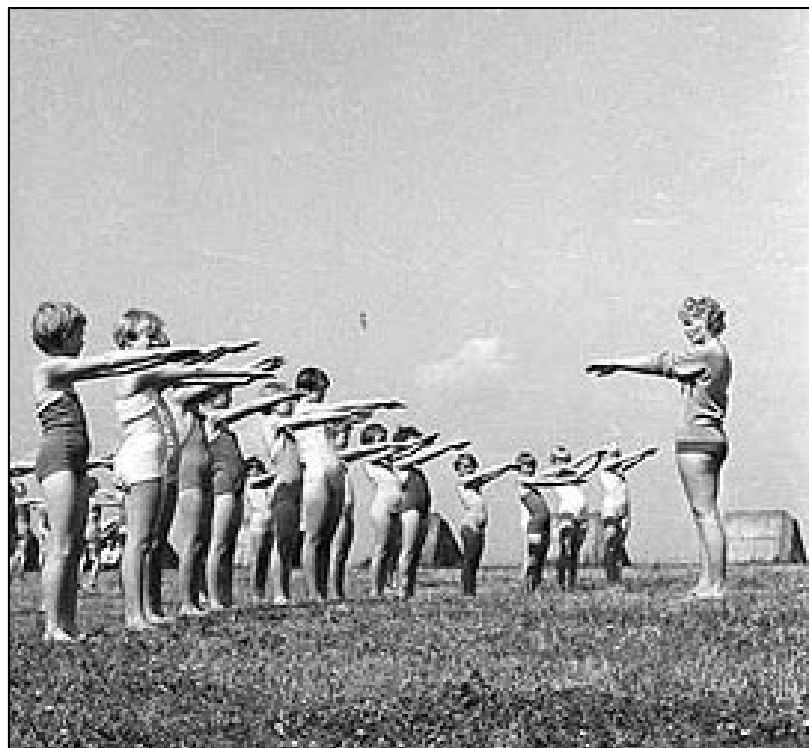
- Principer (se även webben):
 - För varje labb, ett sista *garanterat* redovisningstillfälle
 - Demonstrera gärna på vanliga labbtillfällen, *om/när det finns tid över*
 - Demonstrera gärna på andra redovisningstider, *om/när det finns tid över*

Datum	Alla	TDDD78 (U1)			TDDE30 (D1)	
		Snabb	Mer tid	Info	Jobba med	Info
2024-01-15						
2024-01-16		Labb 1	Labb 1		Labb 1	
2024-01-17		Labb 1	Labb 1		Labb 1	
2024-01-18		Labb 1	Labb 1		Labb 2	
2024-01-19		Labb 2	Labb 2	Demo labb 1-2.	Labb 2	Demo labb 1-2.
2024-01-20						
2024-01-21						
2024-01-22		Labb 2	Labb 2		Labb 2	
2024-01-23		Labb 2	Labb 2		Labb 2	
2024-01-24		Labb 2	Labb 2		Labb 2	
2024-01-25		Labb 3	Labb 2	Demo labb 1-2. Sista <i>garanterade</i> chansen för labb 1 före mars.	Labb 2	
2024-01-26		Labb 3	Labb 3		Labb 3	Demo labb 1-2. Sista <i>garanterade</i> chansen för labb 1 före maj/juni.
2024-01-27						
2024-01-28						
2024-01-29		Labb 3	Labb 3		Labb 3	
2024-01-30		Labb 3	Labb 3		Labb 3	
2024-01-31		Labb 3	Labb 3		Labb 3	
2024-02-01		Labb 4	Labb 4		Labb 4	

Labbarna



"Varsågod, lär dig simma!"



*...eller ska man öva
"handgreppen" först?*

Fokus: Lär in rätt sätt, inte fel (svårt att bli av med)

Medel: Led till bra lösningar
Tutorial-form: Steg-för-steg-instruktioner

Bara ett förberedelsesteg!

Mindre erfarenhet → effektiv metod för "rätt programmeringsvanor"

Mer erfarenhet → snabbt avklarat + kan få nya insikter / idéer...

Resultatet är inte kod, utan kunskap!

Tänk, reflektera, förstå varför vi programmerar på ett visst sätt

Feedback och hjälp för labbar och projekt

**Utvecklingsmiljön
(IntelliJ IDEA,
eller fritt val
på eget ansvar)**

Feedback!

3000+ "inspektioner" som ger tips till programmeraren

```
8  ▶ public class JavaWarnings
9  {
10     public void openFile() {
11         try {
12             new FileInputStream( name: "/tmp/foo.txt").read();
13         } catch (IOException e) {
14             e.printStackTrace();
15         }
16     }

```

'FileInputStream' used without 'try'-with-resources statement

```
18  ▶ public static void main(String[] args) {
19     String name = null;
20     final int size = name.length();
21
22     List names = new ArrayList();
23     name
24 }
25

```

Raw use of parameterized class 'List'

Community-version (open source) alt. gratis studentlicens

Utvecklingsmiljön...
och ett eget verktyg
med fler tips + mer info

Feedback!

Inspection Results

file:///home/jonas/teach/javaoo-test/newinspection.html

Inspekterar teach

JavaWarnings.java: 0 fält, 2 metod(er)

22 List names = new ArrayList();

oo-**RawUseOfParameterizedType** 2 instanser ✕

- A parameterized (generic) type such as `List<ElementType>` is used without providing an actual element type (only `List`). This is supported in Java for backward compatibility but should never be used in new code, as type safety is reduced.
- This is considered a serious issue and must definitely be fixed before handing in a project or lab.
- Raw use of parameterized class `List`

oo-**RawUseOfParameterizedType**

- Raw use of parameterized class `ArrayList`

programming-**MismatchedCollectionQueryUpdate** 1 instans ✕

- A collection is only updated without being queried, or vice versa. Can indicate a bug (why would we add objects without using them?), or incomplete code, or preparation for future features. In either case, it is typically unknown whether the correct objects are added.
- Contents of collection `names` are updated, but never queried

23 names.add(name);

oo-**UNCHECKED_WARNING** 1 instans ✕

- This warning can occur in several situations.
- *Unchecked assignment* can for example mean that you are assigning a value of a 'raw' generic type, such as `List`, to a variable that specifies an element type, such as `List<String>`. Make sure that you specify element types everywhere (or in some cases the 'diamond' `<>`, meaning 'the same element types that the variable already specifies').
- Another situation is where a dynamic cast executed at runtime cannot actually check whether the object has the

**Körs när ni pushar till Gitlab;
ger en issue med bifogad rapport**

- Våra erfarenheter:
 - Ger **snabbare återkoppling** än att fråga handledaren
 - Ger **ständig återkoppling**, även mellan labbtillfällen
 - Ger **tips** så man lär sig mer om Java och objektorientering
 - Ger **färre rutinkompletteringar** och kanske snabbare CSN-pengar...

- Tänk på:
 - Ett påpekande betyder **inte** att **du borde ha kunnat det här redan**
 - Det är ett **inlärningsstillfälle**:
Det är **så här du får lära dig mera** i ett ämne med **tusentals detaljer**

- Man måste inte alltid åtgärda alla varningar
 - Men ett verktyg kan inte avgöra var gränsen går (godkänt, betyg 3-4-5)...
 - Frustrerande: Hur mycket borde jag göra egentligen?
- Utan verktyget:
 - Lämna in, få tillbaka komplettering
 - Frustrerande: "...*Hade jag bara fått en vink hade jag kunnat fixa det direkt!*"
- Med verktyget:
 - Checka in, kolla resultatet, *få en vink*
 - En del varningar **borde uppenbart fixas** – nu får du veta det tidigare
 - En del varningar kanske du fixar **för att vara säkrare på godkänt**

Till slut alltid en helhetsbedömning – hur bra är koden totalt?

Handledare på labbar – Fråga!

Be om feedback även *när allt verkar fungera:*

Största problemen är de
man inte har sett att man har...

Utvecklingsmiljön:

Analyserar koden, visar
problem / möjliga förbättringar,
ger *inlärningsmöjligheter*

Feedback!

En hel kurs labbar på en gång:

**TDDD78,
TDDE30**

Handuppräckning i Teams
(ett team per grupp),
handledning på plats

Jobba utanför schemalagd tid

**Ta med "färdiga frågor"
till handledningen**

**Fråga uppstår i slutet
eller "överbelastat" tillfälle
→ kanske måste vänta
till nästa gång**

Handledare på labbar – Fråga!

Be om feedback även när allt verkar fungera:
Största problemen är de man inte vet om...

Utvecklingsmiljön:

Analyserar koden, visar problem / möjliga förbättringar, ger inlärningsmöjligheter

Inlämningar:

Kompletteringar ger också en chans att lära sig

Feedback!

Kursdeltagare:

Fritt fram att diskutera, men inte skriva av:
Lösningen ska vara er egen

Ni ska också bli självständiga ingenjörer!

Tveka inte att fråga,
men handledarna ska ge ledtrådar och tips, inte färdiga svar.

Ändringar sedan tidigare år

Diskuteras under en annan föreläsning

Skickar ut info i förväg

Nästa steg...

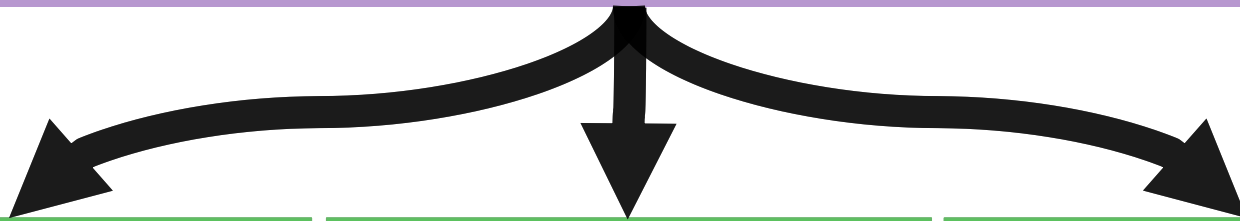
Java för Python-programmerare

Varifrån kommer Java? Rötter sedan 60-talet...

Varför använda det i kursen?

Hur snabbt är det? En ovetenskaplig jämförelse...

Jämförelser med Python: Att starta ett program, syntax, uttryck, operatorer, villkorssatser, loopar (iteration), ...



Föreläsning

**Önskemål från
tidigare studenter!**

Läs / skumma själv

**Om det passar dig
bättre!**

Hoppa över...

**Om du vet
att du kan allt!**