TDA 545: Objektorienterad programmering

Föreläsning 1:
Introduktion

Magnus Myréen

Chalmers, läsperiod 1, 2015-2016

Vad handlar kursen om?

Det här är en grundkurs i programmering.

Programmering språket är Java.

Kursen handlar om:

1: hur man skriver program som är
- korrekta (i rimlig grad)
- läsbara och enkla att underhålla (av andra)
- återanvändbara
- rimligt effektiva,
- användarvänliga,
- klara i tid (gäller även labbar) …

2: om problemlösning (viktigt), hur man lär sig själv hitta lösningar, hur man lär sig mera

Niklas livsvisdom!

Vad handlar kursen om?

Det här är en grundkurs i programmering.

Programmering språket är Java.

Kursen handlar om:

3: grunder i Java:

- enkla datatyper, kontrollstrukturer, metoder (beräkningsabstraktioner)
- klasser, objekt, arv, interface (gränssnitt), mm
- exceptions
- grafiska gränssnitt

4: objektorientering: abstraktion, modularisering, composition, programdesign

Varför Java?

Java är ett populärt språk.

Massor med kod skrivs i Java.

Arbetsgivare vill att programmerare kan Java.

Dessutom är Java ett rent högnivå språk!

Denna föreläsning:

Del 1: **hur fungerar kursen**

Del 2: **vad är programmering?**

Del 3: **hur lär man sig programmera?**

Kursens mål

Kursens syfte är att lära ut
principerna för det objektorienterade synsättet och
hur man konstruerar objektorienterade program.

Kursens websida

http://www.cse.chalmers.se/edu/course/TDA545/

(finns också som kort adress: http://tinyurl.com/tda545)

Här hittar ni all information om kursen.

Hemsidan uppdateras kontinuerligt under kursens gång.

T.ex. föreläsningspresentationerna (dessa *slides*) kommer upp *efter* varje föreläsning.

Om någonting saknas, skicka mig en fråga via epost:

myreen@chalmers.se

E-post

Om någonting saknas, skicka mig en fråga via epost:

myreen@chalmers.se

Skicka gärna frågor per e-post till mig.

Kom ihåg att:

- beskriva problemet
- vad hände?
- skicka med felmeddelandet
- klistra in den söndriga Java koden
- ifall koden är längre än tio rader, skicka den som en bilaga till meddelandet

Kursens komponenter

Människor:

- ni: studenterna!
- jag (föreläsaren/kursansvarig): Magnus Myréen
- assistenter: Uno Holmer, Ivar Josefsson, Erik Jungmark, Andreas Wieden

Händelser:

- föreläsningar (teori, exempel)
- övningstillfällen (exempel i små gruppe
- labbar (självverksamhet med handledar
- … och tentamen på slutet

Schemat finns på kursen hemsida.

OBS: Bara tentamen och labbar måste göras, ingen närvaro krävs normalt.

Studenterna (den viktigaste komponenten!)

Hur många har programmerat förr?

Hur många har själv skrivit ett program på 20 rader eller mera?

Hur många har själv skrivit ett program på 200 rader eller mera?

Hur många tycker att de kan programmera ganska bra?

Hur många har själv skrivit Java program?

Föreläsaren (ansvarig för kursen)

Jag (dvs Magnus Myréen) har börjat jobba på Chalmers förra året (2014-09-01).

Från Finland.
Finlandssvensk.

Studerade i Oxford till BA i Computer Science (2002-2005).

Doktorerade från Cambridge i CS år 2008.

Var forskare (postdoc & research fellow) till augusti 2014.

Nu är jag forskare här. (Jag forskar i formella metoder och FP.)

Gruppindelning

Det är cirka hundra studerande i den här kursen.

För att få ner siffran till lämpliga små grupper, så ska ni dela upp er i fyra grupper.

A, B, C, D

Samma grupper som i *Grundläggande datorteknik EDA433*

Varför denna indelning? Svar: se schemat på hemsidan.

Ni bör också hitta en labbkompis för labbarna.

Labbarna

Ni bör också hitta en labbkompis för labbarna.

Labbarna görs i par.

Det finns fyra labbuppgifter (labbar), med deadlines:

Lab 1: Registrering i Fire
(deadline: 18:00, tisdag 8.9)

Lab 2: Metoder och fält; en klass för rationella tal
(deadline: 18:00, tisdag 22.9)

Lab 3: Counter och ~~Secant~~ Labyrint
(deadline: 18:00, tisdag 6.10)

Lab 4: Memory
(deadline: 18:00, onsdag 21.10)

Labbarna

Man lämnar in labbar genom Fire systemet.

Ni får feedback på er labb senast tre arbetsdagar efter deadlinen.

Man får poäng för labbar, som är godkända.

Om labben inte är godkänd, får man en vecka på sig att lämna in en ny version.

För att komma igenom kursen, måste man ha:

- minst *ett poäng* för varje del av labuppgifterna, och
- minst hälften av alla poängen totalt (dvs *9 poäng totalt*)

Bedömning / vitsord

För att komma igenom kursen, måste man ha:

- minst *ett poäng* för varje del av labuppgifterna, och
- minst hälften av alla poängen totalt (dvs *9 poäng totalt*)

då kommer får man godkänt för kursens labbar.

För tentamen får man vitsord: U, 3, 4 eller 5.

Feedback

Skicka gärna frågor och kommentarer till mig!

myreen@chalmers.se

Apropå feeback och labbar:

"Dåligt formulerade uppgifter. Lägg ner lite mer tid på att förklara vad du vill att vi skall göra. Det skulle innebära att vi får mer tid till att koda på övning för att vi inte behöver sitta och fundera på vad vi skall göra för något."

Det är normalt att man måste ägna tid till att fundera ut ***vad*** som skall göras. Ni må tycka illa om det men så är det. Sen måste man också fundera ut ***hur*** det skall göras.

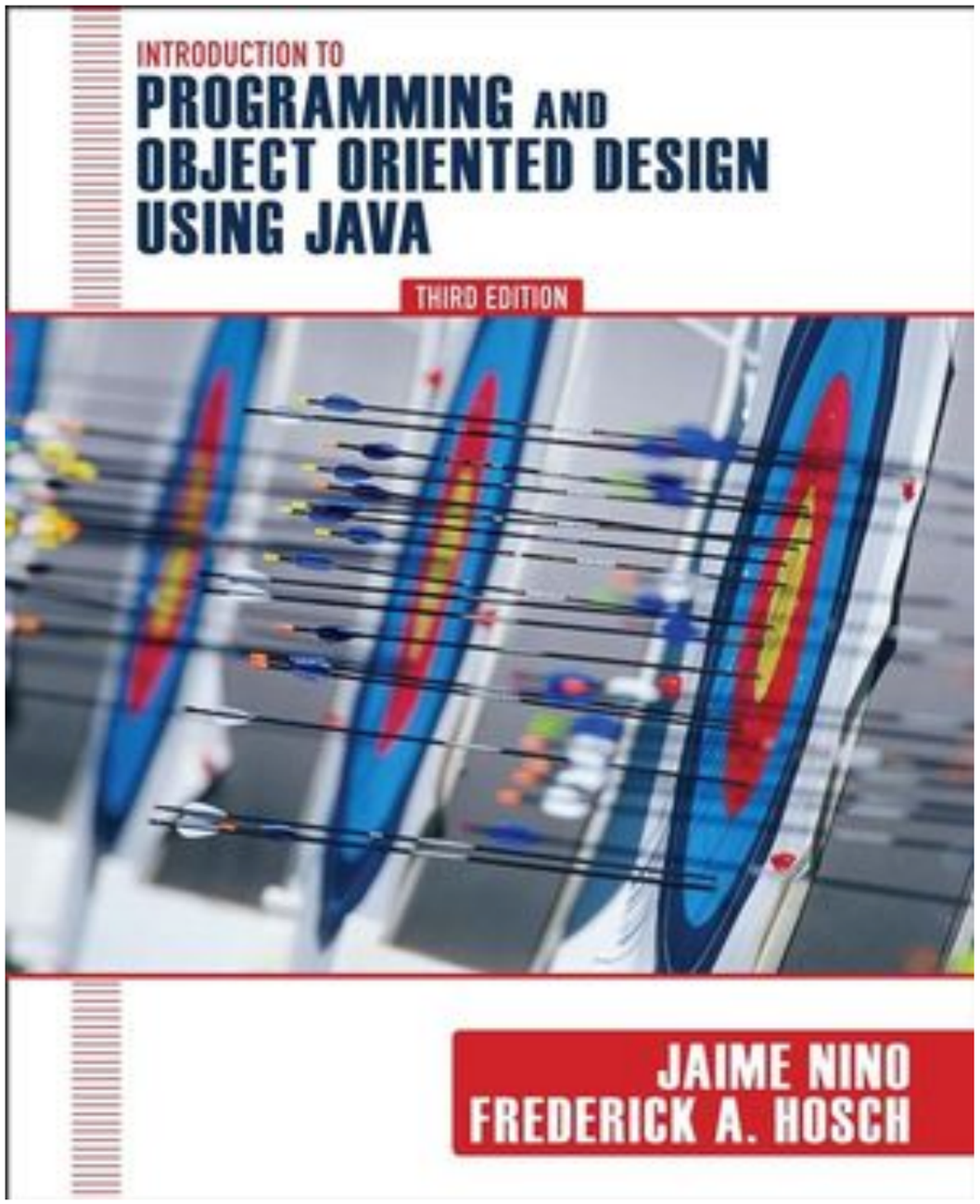
Mera feedback

Kursen behöver 3 eller 4 **student representanter**.

- ett möte i studie vecka 2 eller 3
- ett annan möte mot slutet av studieperioden
- ett sista möte efter slutet av kursen

Idén är att representanten berättar vad studenterna tycker om kursen. Vad kunde förbättras?

Kursens bok



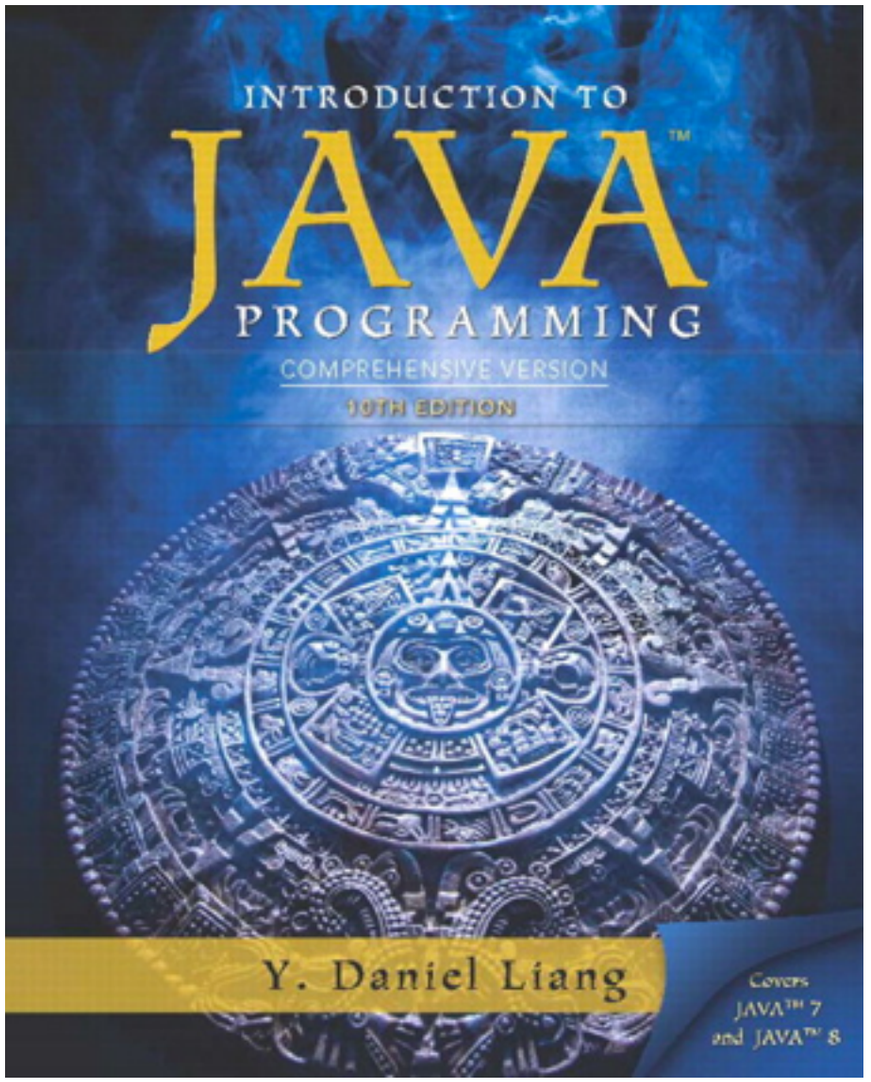
ISBN: 978-0-470-12871-8

Boken är lättläst och rätt välskriven om än lite pratig. I grova drag så ingår kapitlen 1-4, delar av 5 och 6, 7-11, delar av 12, 13, delar av 14, 15, 17, 18, delar av 19, supplement b, appendix i, se hemsidan för mer detaljer.

Cirka 700 sidor att läsa.
Försök läsa 100 sidor per vecka.

Men glöm inte att programmera!
Gör t.ex. så många av övningarna som möjligt.

Andra böcker om Java (förslag av Maxim Goretskyy)



http://www.cse.chalmers.se/edu/course/TDA545/index.html#bok

Vad ni ska göra nästa dagarna

Från hemsidan:

Läsvecka 1:
måndag 10:00 – 11:45: föreläsning, rum HA4
tisdag 10:00 – 11:45: föreläsning, rum HC4
tisdag 15:15 – 17:00: laboration, rum 5355, ED-3507
torsdag 10:00 – 11:45: föreläsning, rum HC4
torsdag 13:15 – 15:00: övning, rum EL43, grupp A, B
fredag 10:00 – 11:45: övning, rum EL41, grupp C, D

Men också:

- funder: vilken grupp vill du vara i
- hitta en labbkompis
- vill du vara student representation för denna kurs?

Lite mera gammal feedback

"I början av kursen (första 2-3 veckorna) skippade jag Sl- passen och några övningspass för att fokusera på labbarna, då jag kände att jag behärskade momenten sedan förut. Kände dock senare att jag skulle ha gått på allt, eftersom det hade varit bra repetition.”

"Jag har nog lagt ner mer tid än de flesta. Men jag tycker inte att det varit någon börda, för det är så kul”

"Jag läste för lite hemma vilket medförde att föreläsningarna i för hög utsträckning rörde material jag inte stiftat bekantskap med tidigare. Då blir det svårt att ta till sig den kunskap som delgavs."

Ännu mera gammal feedback

"Latade mig istället för att gå på övningar, vilket jag inte borde ha gjort”

"programmering tycker jag man lärde sig som bäst av att testa själv”

"Jag känner nu efteråt att anledningen till att jag klarat mig så bra som jag faktiskt har är att jag lagt ned mycket tid själv, utan min partner på labbarna. Vårt samarbete har fungerat väl, men jag har ändå varit noga med att skriva 100% av all kod själv (min partner har givetvis skrivit sin egen kod också), vilket jag nu känner har varit nödvändigt för att få nog praktisk erfarenhet av programmering för att klara mig."

Denna föreläsning:

Del 1: **hur fungerar kursen**

Del 2: **vad är programmering?**

Del 3: **hur lär man sig programmera?**

Vad är programmering?

Programmering = att berätta för datorn hur
den ska utföra en uppgift.

Kod — detaljerade anvisningar, som är maskinläsbara

Algoritm — liknande anvisningar, men avsedda för människor

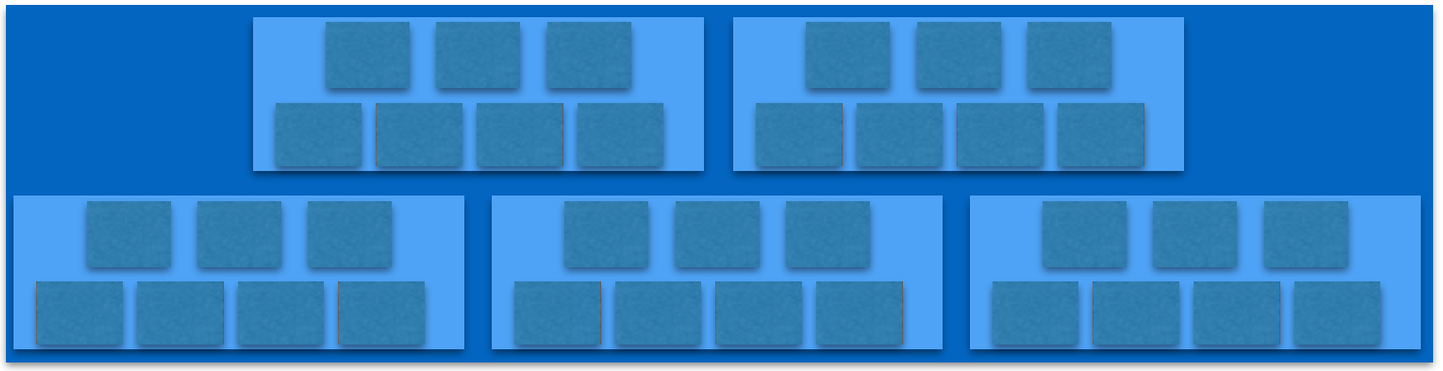
Var ser man algoritmer?

Vad är (objekto.) programmering?

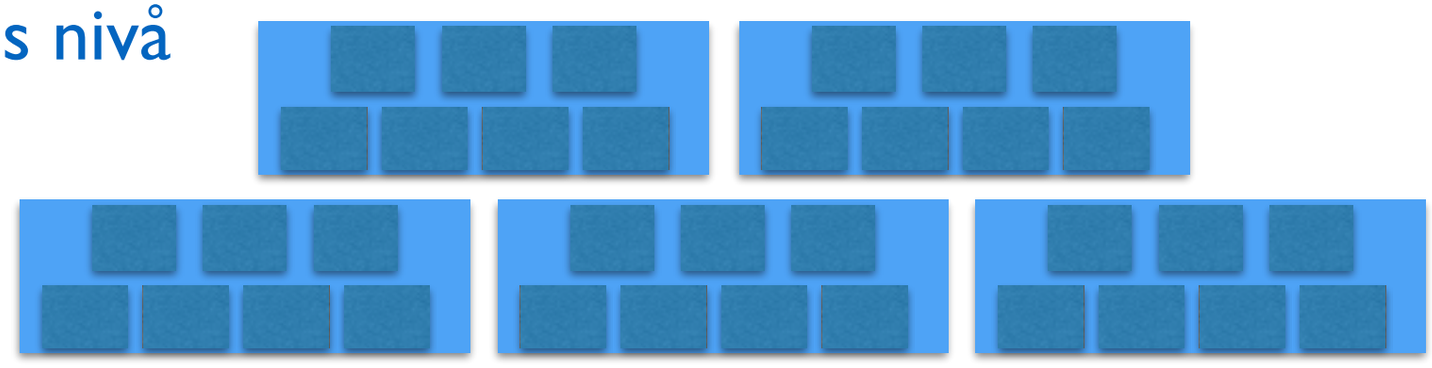
Det är som att bygga med Lego…

… men bitarna kan tillsammans bli större ***bättre*** bitar.

högre abstraktions nivå



högre abstraktions nivå



Något om Abstraktion (= subtraktion?)

Det är svårt att tänka på flera nivåer på samma gång.

Abstraktion är ett verktyg.

Abstrahera = Att införa ett begrepp

Ett begrepp fångar vad som är gemensamt

"Bortse från eller framhäva gemensamma
och viktiga egenskaper hos en samling objekt
och bortse från detaljer och konkret realisation."

Abstraktioner hjälper oss att fokusera på de för tillfället viktiga egenskaperna.

Ex skottår — det är en viktig egenskap hos ett årtal men vi behöver inte veta exakt hur man avgör om ett årtal är ett skottår för att kunna använda oss av egenskapen.

Abstraktioner

Vi abstraherar varje dag på olika nivåer:

atomer — primitiv syn på världen
organ — biologisk samling atomer
Sam (katten) — namn på en specifik samling organ
katt — en större klass där Sam ingår
djur — en ännu större klass levande organism

de flesta detaljer utelämnade

- varje steg representerar olika synsätt på samma objekt — olika abstraktionsnivåer
- vilken nivå vi väljer att använda beror på våra behov för tillfället
- varje nivå "implementeras" på en lägre nivå

Abstractions

Computer Scientists build abstractions to

- extract common features
- hide irrelevant details
- control complexity
- protect integrity

Abstraction is used to present a simple consistent model.

Teachers do the same thing!

We abstract away from the complexities of material

- present simple consistent model
- slowly add complexity
- filling in exceptional cases later

"Black box" — tänkande

En insida och en utsida:

Gränssnittet (interfacet) är det som binder samman delarna.
Interfacet har såväl syntax som semantik (specifikationen)

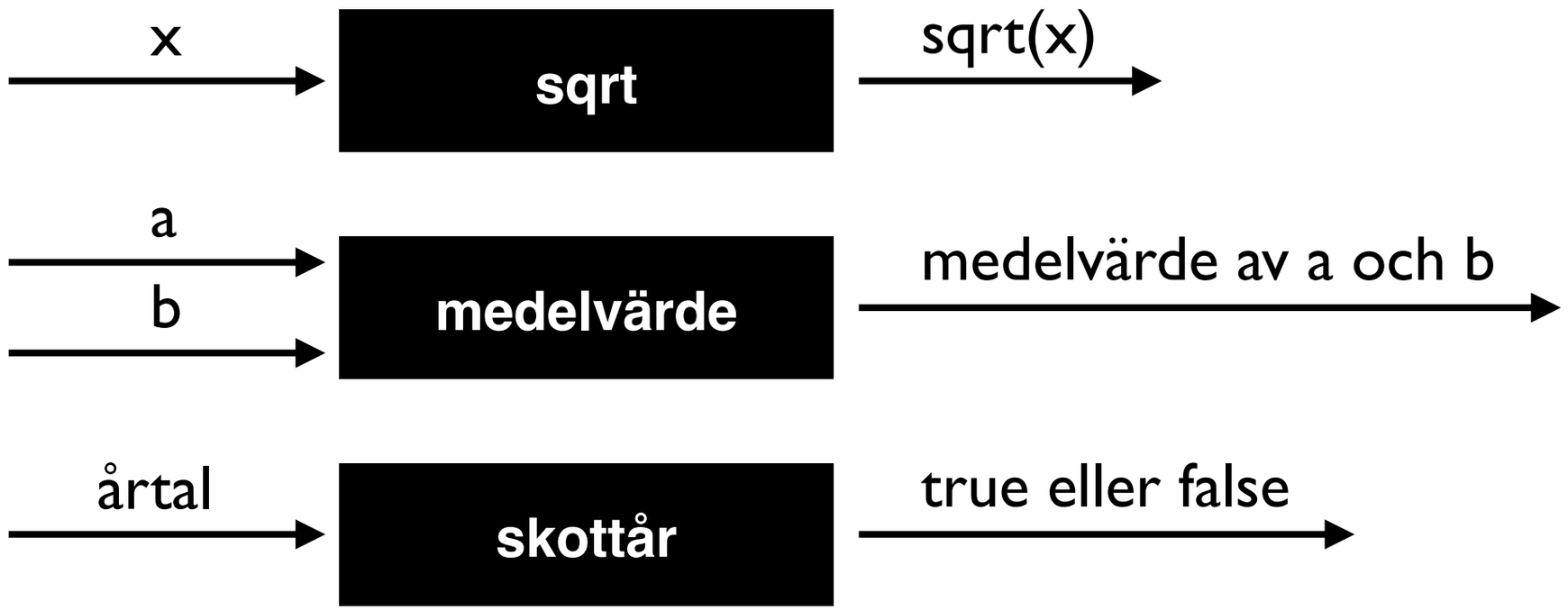
> "*The interface of a black box should be fairly straightforward, well defined, and easy to understand.*"
>
> "*To use a black box, you shouldn't need to know anything about its implementation; all you need to know is its interface.*"
>
> "*The implementor of a black box should not need to know anything about the larger systems in which the box will be used.*"

En black box är ett sätt att uttrycka vissa steg i ett program på en högre abstraktionsnivå.

Vi kan abstrahera — gömma detaljer. ***vad*** inte ***hur***

Metoder som Svarta boxar

metod ≈ underprogram ≈ funktion ≈ procedur



Ett sätt att uttrycka vissa steg i ett program på en *högre* abstraktionsnivå.
Ger direkt stöd för stegvis förbättring.

Vi kan abstrahera — gömma detaljer. ***vad*** inte ***hur***

Varje box skall endast göra **en** sak bra.

Metoder är begränsade =>
vi behöver "större" svarta boxar

Antag tex att vi behöver representera ett datum.
Här räcker inte en metod (som ju skall göra en sak bra) för vi vill kunna göra flera saker med ett datum:

- skottår?
- addera två datum
- undersöka vilken veckodag ett viss datum inträffar
- fråga efter nästa datum (nästa dag)
- osv

Vi behöver *något kraftfullare*. Något som i en enhet kan representera ett datum och dessutom alla dessa egenskaper hos ett datum.

Klasser/Objekt modellerar hur vi tänker om den verkliga världen genom att stödja nedbrytningen av världen till hanterbara saker.

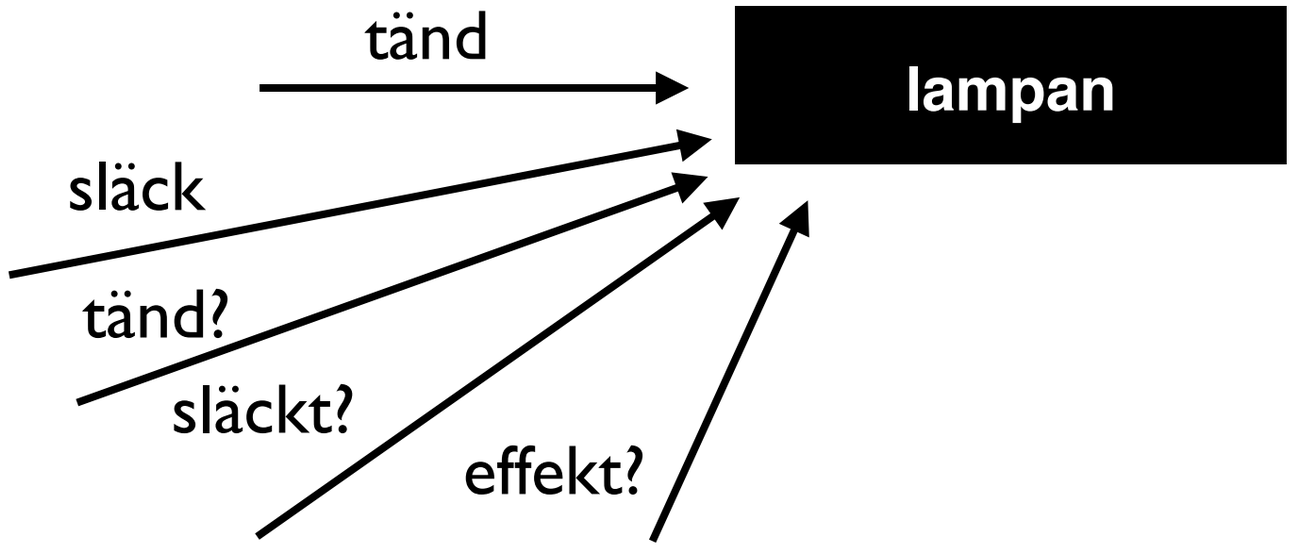
Objekt som Svarta boxar

Identitet: hall-lampan

Tillstånd (state):

- tänd, släkt, sönder
- effekt: (40 / 60 …W?)

Beteende (methods):

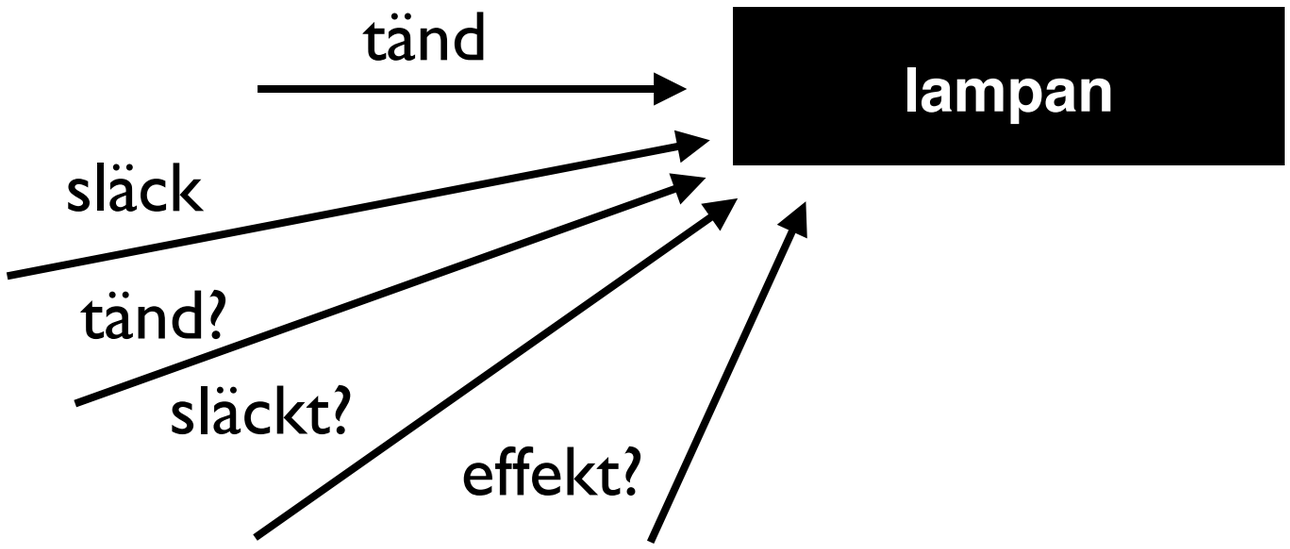


Vi kan abstrahera — gömma detaljer. *vad* inte *hur*

Varje box skall endast göra en sak bra.

Objektorienterad programmering

Objektorienterade system är strukturerade runt sina objekt
dvs funktionaliteten är distribuerad till systemets objekt.



Syntax och Semantik

Språkets syntax och semantik måste beskrivas exakt.

formen —
hur man får skriva

innebörden i det
man skriver

10/9 2007 eller
2007 10/9 eller
071009 eller
September 10, 2007 eller
9/10/07 eller 10/9/07 samma innebörd eller?

Abstraktionens syntax och semantik måste beskrivas noggrant:
Ett datum på formen ååååmmdd med operationerna:

- år, månad, dag, skottår: alla med ett datum som input
- +: med två datum som parameter (dvs addera)
- datum: med år, månad och dag input

Syntax och semantik är viktigt

Man måste känna syntaxen/semantiken för att kunna programmera.

```
public class Problem {
   public static void main
                        (String[] args) {
   // ----------------------
      int antal = 0;
      for ( int i=1; i<10; i++ ) {
         antal = antal++;
         System.out.println(antal);
      } // end for
   // ----------------------
   } // end main
} // end Problem
/*
   Vad skrivs ut och
   hur skrivs det ut?
*/
```

Program = Satser (syntax)

Ett program byggs huvudsakligen upp av deklarationer och satser

Enkla satser (enkla satser avslutas med ";")

- tilldelning (assignment)
- metodanrop (av void metoder)
- break ("kortslutning" av loop/switch sats)
- return <uttryck>

Sammansatta satser (avslutas ***inte*** med ";" utom do-while)

- villkorsats (if)
- flervalsats (switch)
- iterationssatser (for, while och do-while)
- blocksats ({satser})

Gör dig en ordlista

- programmeringsspråk,
- syntax,
- semantik,
- abstraktion,
- modularisering,
- composition,
- objektorientering,
- datatyper,
- kontrollstrukturer,
- metoder,
- procedurer,
- funktioner,
- black box,
- interface (gränssnitt),
- …

Läsa boken.
Försök läsa 100 sidor per vecka.

Denna föreläsning:

Del 1: **hur fungerar kursen**

Del 2: **vad är programmering?**

Del 3: **hur lär man sig programmera?**

Hur lär man sig cykla?

Det räcker inte med att läsa en bok om cykling…

Man måste försöka själv!
… och så lär man sig genom sina misstag.

experience

=

sum of all previous mistakes

Hur lär man sig programmera?

Hur blir man en erfaren programmerare?

Det räcker inte med att läsa en bok om programmering…

Man måste försöka själv!
… och så lär man sig genom sina misstag.

Börja skriva Java program!

Hur gör man det?

Du är inte den första som vill lära dig Java… Google-a!

Det gäller att installera **javac** och **java**, dvs **JDK**.

JDK = Java (Standard Edition) Development Kit

Bläddra igenom ‘tutorials’ och annan dokumentation.

Ha gärna sidoprojekt!

Hacka lite Java på sidan om.

Lär dig skriva program som t.ex.

skriver ut ett mönster med ASCII,
skriver en fil, ändrar en bildfil,
fungerar på din Android telefon!
spelar lite musik, eller
berättar om det blir regn idag, mm.

Lär dig hitta svar!

Vad intresserar dig?

Det kommer att vara ***mycket trögt*** i början, men det blir bättre.

Allt du lär dig om Java kommer att vara nyttigt i längden.

CV tips!

En trist CV har bara lista på all kurser som du har gott.

En fin CV har en lista på tuffa projekt som du har lyckats med!

experience

=

sum of all previous mistakes

Sammanfattning

Del 1: **hur fungerar kursen?**

Föreläsningar, labbar, övningar, bok och tentamen.

Del 2: **vad är (objeckto.) programmering?**

Att bygga med objekt som klossar.
Black-box tänkande. Abstraktion.

Del 3: **hur lär man sig programmera?**

Programmera, programmera, programmera…
Programmera något som intresserar dig!
Försök hitta svar!