

EDA016 Programmeringsteknik för D

Läsvecka 1: Introduktion

Björn Regnell

Datavetenskap, LTH

Lp1-2, HT 2015

1 Introduktion

- Om denna kurs
- Vad är programmering?
- Vårt första Java-program
- Grundläggande programkonstruktioner i Java
- Sammanfattning
- Meddelande från [Code@LTH](#)

Om denna kurs

Vad och hur?

■ *Vad* ska du lära dig?

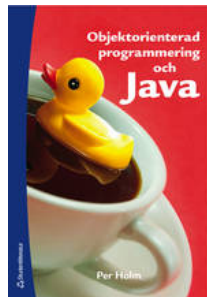
- Grundläggande principer för programmering
- Konstruktion av enkla algoritmer
- Tänka i abstraktioner
- Imperativ och objektorienterad programmering
- Programspråket Java
- Utvecklingsmiljön Eclipse: implementera, testa, felsöka

■ *Hur* ska du lära dig?

- Genom praktiskt eget arbete: **Lära genom att göra!**
- Genom studier av kursens teori: **Skapa förståelse!**
- Genom samarbete med dina kurskamrater: **Gå djupare!**

Kurslitteratur

- "Objektorienterad programmering och Java" av Per Holm
- Kurskompendium med övningar och laborationer
- Bokpaket säljs på KFS
John Ericssons väg 4
<http://www.kfsab.se/>



Personal

Kursansvarig:

Björn Regnell, bjorn.regnell@cs.lth.se

Kurssekreterare:

Lena Ohlsson

Exp.tid 09.30 – 11.30 samt 12.45 – 13.30

Handledare:

Maj Stenmark, Tekn. Lic., Doktorand

Gustav Cedersjö, Doktorand

Anton Klarén, D09

Maria Priisalu, D11

Anders Buhl, D13

Erik Bjäreholt, D13

Fatima Abou Alpha, D13

Cecilia Lindskog, D14

Emma Asklund, D14

Kursmoment — varför?

- **Föreläsningar:** skapa översikt, ge struktur, förklara teori, svara på frågor, motivera varför
- **Övningar:** bearbeta teorin med avgränsade problem som mestadels löses med papper & penna, förberedelse inför laborationerna
- **Laborationer:** lösa programmeringsproblem praktiskt, obligatoriska uppgifter; lösningar redovisas för handledare
- **Resurstider:** få hjälp med övningar och laborationsförberedelser av handledare
- **Samarbetsgrupper:** grupplärande genom samarbete och dialog
- **Kontrollskrivning:** obligatorisk, diagnostisk, kamraträttad; kan ge samarbetsbonuspoäng till tentan
- **Inlämningsuppgift:** du visar att du kan skapa ett större program självständigt; redovisas för handledare
- **Tenta:** Skriftlig tentamen utan hjälpmedel, förutom **snabbreferens**.

Nytt för i år

Årets kurs är i flera avseende väsentligt annorlunda and förra årets upplaga, så lita inte på allt som era äldre kursare säger :)

- Övningar blir resurstider i datorsal
- Inlämningsuppgift utan skriftlig rapport
- Samarbetskultur och grupplärande
- Nya övningar
- Nya laborationer
- Nya föreläsningar

Ändringarna är framtagna i samråd med studierådet. Mer om bakgrunden här:

<http://fileadmin.cs.lth.se/cs/Education/EDA016/2015/update.pdf>

Detta är bara början...

Exempel på efterföljande kurser som bygger vidare på denna:

■ Årskurs 1

- Programmeringsteknik – fördjupningskurs
- Utvärdering av programvarusystem
- Diskreta strukturer

■ Årskurs 2

- Objektorienterad modellering och design
- Programvaruutveckling i grupp
- Algoritmer, datastrukturer och komplexitet
- Funktionsprogrammering

Utveckling av mjukvara i praktiken

- **Inte bara kodning:** kravbeslut, releaseplanering, design, test, versionshantering, kontinuerlig integration, driftsättning, återkoppling från dagens användare, ekonomi & investering, gissa om morgondagens användare, ...
- **Teamwork:** Inte ensamma hjältar utan autonoma team i decentraliserade organisationer med innovationsuppdrag
- **Snabbhet:** Att koda innebär att hela tiden uppfinna nya "byggstenar" som ökar organisationens förmåga att snabbt skapa värde med hjälp av mjukvara. Öppen källkod. Skapa kraftfulla API.
- **Livslångt lärande:** Lär nytt och dela med dig hela tiden. Exempel på pedagogisk utmaning: hjälp andra förstå och använda ditt API \implies *Sammarbetskultur*

Att skapa koden som styr världen

I stort sett alla delar av samhället styrs av mjukvara:

- kommunikation
- transport
- byggsektorn
- statsförvaltning
- finanssektorn
- media
- sjukvård
- övervakning
- integritet
- upphovsrätt
- miljö & energi
- sociala relationer
- utbildning

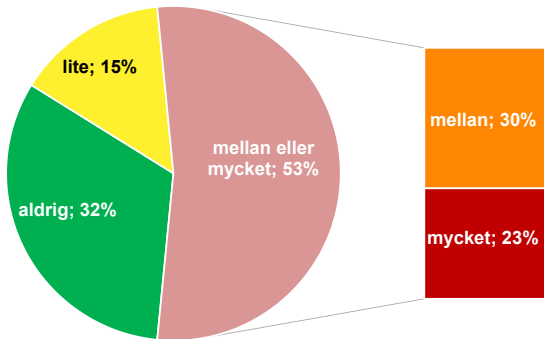
Hur blir ditt framtida yrkesliv som systemutvecklare?

- Redan nu är det en skriande brist på utvecklare och bristen blir bara värre och värre...
CS 2015-08-17
- Global kompetensmarknad
CS 2015-06-14
CS 2015-08-15
- Fokus på innovation, tid-till-marknad
- Autonoma utvecklingsteam i decentraliserade organisationer
- Öppen källkod

Förkunskaper

- Förkunskaper \neq Förmåga
- Varken kompetens eller personliga egenskaper är statiska
- "Programmeringskompetens" är inte *en* enda enkel förmåga utan en komplex sammansättning av flera olika förmågor
- Ett innovativt utvecklarteam behöver många olika kompetenser för att vara framgångsrikt

Stor spridning i programmeringsförkunskaper bland D-are (enl. enkätsvar 2010-2014)



Förkunskapsenkät

Fyll i denna enkät idag:

<http://cs.lth.se/eda016/survey>

Ligger till grund för randomiserad gruppindelning,
så att det blir en spridning av förkunskaper inom gruppen.

Varför samarbetsgrupper?

Huvudsyfte: **Djupinläring!**

- Pedagogisk forskning stödjer tesen om att lärandet fördjupas om det sker i utbyte med andra
- Ett studiesammanhang med höga ambitioner och respektfull gemenskap gör att vi når mycket längre
- Varför ska du som redan kan mycket anstränga dig mer genom att dela med dig av dina kunskaper?
 - Förstå bättre själv genom att förklara för andra
 - Träna din pedagogiska förmåga
 - Förbered dig för inför ditt kommande yrkesliv som mjukvaruutvecklare

Samarbetskontrakt

Gör ett skriftligt samarbetskontrakt med dessa och ev. andra punkter som ni också tycker bör ingå:

- 1 Kom i tid till gruppmöten
- 2 Var väl förberedd genom självstudier inför gruppmöten
- 3 Hjälp varandra att förstå, men ta inte över och lös allt
- 4 Ha ett respektfullt bemötande även om ni har olika åsikter
- 5 Inkludera alla i gemenskapen

Diskutera hur ni ska uppfylla dessa innan alla skriver på.

Ta med samarbetskontraktet och visa för handledare på labb 1.

Om arbetet i samarbetsgruppen inte fungerar ska ni mejla kursansvarig och boka mötestid!

Plagiatregler

Samarbetsbonus

En typisk kursvecka

- Gå på föreläsningar måndag–tisdag
- Jobba själv med boken, övningar, labbförberedelser
- Träffas i samlingsgruppen och hjälp varandra att förstå mer och fördjupa lärandet
- Gå till resurstider och få hjälp och tips av handledare, onsdag–torsdag
- Genomför obligatorisk laboration på fredagen

Se detaljerna och undantagen i [schemat i TimeEdit](#)

Resurstider

Laborationer

Att göra i Vecka 1: Rivstarta

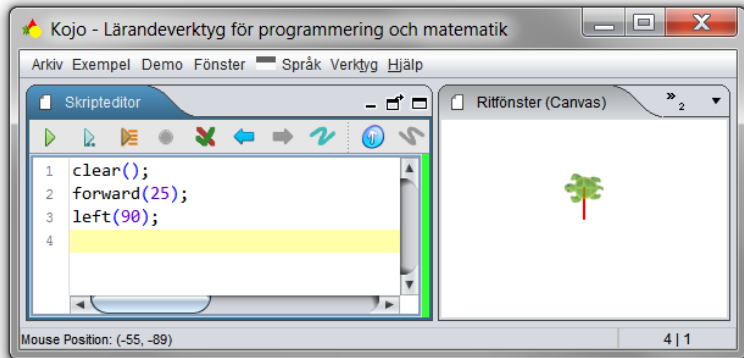
- 1 Läs följande kapitel i kursboken:
1, 3.1-3.3, 5.1-5.3, 6.1-6.2, 7.1-7.3, 7.5-7.6, 7.8-7.9
- 2 Gör övning 1: Hello World, Hello Args, javac, editera–kompilera–exekvera–felsök, värden, uttryck variabler och tilldelning
- 3 Gör förberedelserna till laboration 1: skapa textfil, etc.
- 4 Träffas i samlingsgrupper och hjälp varandra att förstå
- 5 Kom till **resurstiderna** och få hjälp och tips
- 6 Genomför laboration 1: Quiz — träna på att editera, läsa, ändra och felsöka i färdig programkod
while, **for**, Scanner

Vad är programmering?

Programming unplugged: Två frivilliga?



Editara och exekvera ett program



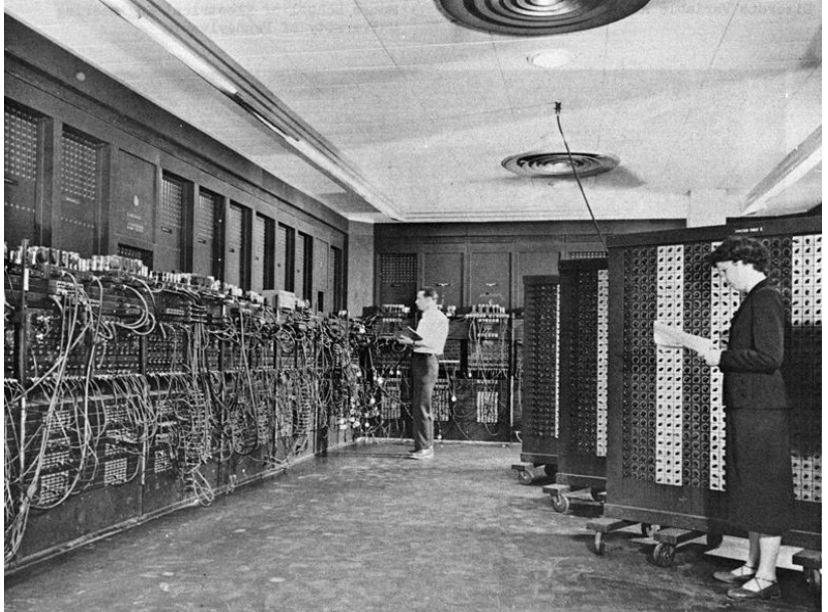
Världens första programmerare



Ada Lovelace skrev världens första datorprogram på 1800-talet.

Programmet skulle köra på en kugghjulsdator som hennes kompis Charles Babbage försökte bygga.

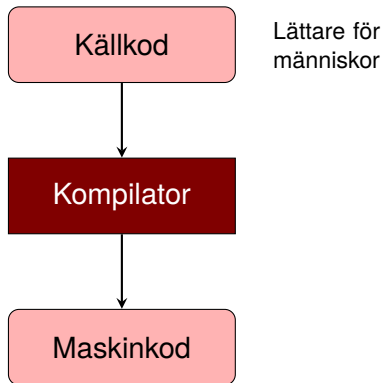
Vad är en dator?



Vad är en kompilator?



Grace Hopper uppfann första kompilatorn 1952.



Exempel på vanliga programspråk

- 1 Java
- 2 C
- 3 C++
- 4 C#
- 5 Python
- 6 Objective-C
- 7 PHP
- 8 Visual Basic .NET
- 9 JavaScript
- 10 Perl

TIOBE Programming
Community Index
Augusti 2015

De 10 "vanligaste"?

Vad är Java?

Utvecklingsverktyg

Vad är objektorientering?

- Det finns många olika **programmeringsparadigm** (sätt att programmera på), till exempel:
 - **imperativ programmering**: programmet är uppbyggt av sekvenser av olika satser som påverkar systemets tillstånd
 - **objektorienterad programmering**: en sorts imperativ programmering där programmet består av objekt som sammanför data och operationer på dessa data
 - **funktionsprogrammering**: programmet är uppbyggt av samverkande (matematiska) funktioner som undviker föränderlig data och tillståndssändringar
 - **logikprogrammering**: programmet är uppbyggt av logiska uttryck som beskriver olika fakta eller villkor och exekveringen utgörs av en bevisprocedur som söker efter värden som uppfyller fakta och villkor

Grundläggande principer i imperativ programmering

- **Sekvens**: Ett program innehåller sekvenser av *satser*. Ordningen mellan dessa har helt avgörande betydelse.
- **Alternativ**: Systemet reagerar på vad som händer och kan välja olika vägar genom programmet beroende på *variablers* värde
Java: **if**-sats, **switch**-sats
- **Repetition**: Göra saker om och om igen
Java: **while**-loop, **for**-loop
- **Abstraktion**: Kapsla in (komplexa) programdelar och sätta namn på dessa så att de enkelt går att återanvända utan att vi behöver "rota i innanndömet".
Java: klasser och metoder

Hello World!

Vårt första Java-program i filen HelloWorld.java

```
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hej och välkomna!");  
    }  
}
```

Kompilera och kör:

```
> javac HelloWorld.java  
> java HelloWorld  
Hej och välkomna!  
>
```

Ovan ingår i övning 1.

Hello World! – Vad betyder egentligen allt detta?

```
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hej och välkomna!");  
    }  
}
```

- **public** Denna programdel är synlig "utåt" och kan användas av andra delar.
- **class** Ett slags "kodbyggblock" som samlar olika programdelar. All java-kod måste finnas i en klass. Det finns tusentals färdiga klasser att använda direkt i Java och man kan lätt skapa egna klasser. Klammerpar { } anger början och slut.
- **static** Denna programdel skapas direkt vid programmets start och det finns exakt *en* sådan här per klass.
- **void** Berättar för kompilatorn att inget värde returneras från denna programdel.
- **main** Berättar var exekveringen av programmet börjar.
- **()** Parentespar berättar för kompilatorn att vi här kan ha parametrar.
- **String[] args** Möjliggör indata till programmet i form av flera textsträngar. Parametern args måste finnas i main, men vi använder den inte i detta program.
- **System.out.println** Den färdiga klassen System kan bl.a. skriva ut text. Textsträngar avgränsas av citationstecken. Semikolon avgränsar satser.

Översikt av några grundläggande programkonstruktioner i Java

- värde (value): data som programmet kan använda
42 "hej" 42.0 **true**
- uttryck (expression): data kombineras med operatorer och ger nya värden
41+1 "h"+"ej" 43.5-1.5 **!false**
- deklaration av variabel (variable declaration): skapa plats i minnet för data
int x = 42;
- tilldelningssats (assignment): ändra värdet på variabler
x = 43;
- alternativ (choice): välj väg beroende på variablers värde
if switch
- repetition (loop): upprepa om och om igen
while for

Värden och uttryck

Alternativ

Välj väg genom programmet med **if**-sats.

```
public class Alternativ {  
    public static void main(String[] args){  
  
        if (true) {  
            System.out.println("Sant!");  
        } else {  
            System.out.println("Falskt!");  
        }  
  
    }  
}
```

En if-sats gör så att exekveringen av programmet kan delas upp i olika grenar; vilken gren som görs beror värdet av ett villkorsuttrycket: **true** eller **false**

Alternativ med variabel

Det blir roligare om vi har en variabel:

```
public class AlternativeWithVariable {  
    public static void main(String[] args){  
  
        int x = 42;  
        if (x >= 42) {  
            System.out.println("Sant!");  
        } else {  
            System.out.println("Falskt!");  
        }  
    }  
}
```

Alternativ med variabel som kan ändra sig

Det blir ännu roligare om vi har en variabel som kan anta olika värden beroende på vad som händer under exekveringens gång:

```
import java.util.Scanner;

public class AlternativeWithVariableThatCanChange {

    public static void main(String[] args){

        Scanner scan = new Scanner(System.in);
        System.out.print("Skriv heltal: ");
        int x = scan.nextInt();

        if (x == 42) { // OBS! dubbla likhetstecken
            System.out.println("Sant!");
        } else {
            System.out.println("Falskt!");
        }

    }

}
```


Meddelande från Code@LTH