Föreläsning 1-2

innehåll

- Om kursen
  - Kursens mål och innehåll
  - Praktisk information om kursen
- Programmering
  - program, algoritmer
  - variabler, datatyper och tilldelningssatser
  - läsa in värden från tangentbordet, skriva ut på skärmen
  - aritmetiska uttryck
  - alternativ och loopar
  - anropa metoder
  - slumptal

EDAA20 Programmering och databaser

- Läsperiod 1
- 7.5 hp
- `niklas.fors@cs.lth.se`
- `cs.lth.se/edaa20`
- `moodle.cs.lth.se`

Detta ska du lära dig

Mål

- Programmering:
  - Lösa problem med hjälp av dator
  - Grundläggande programmering
  - Grundläggande objektorientering och programspråket Java
- Databaser:
  - Lagra data i relationsdatabaser
  - Använda SQL för att hämta data från databaser

Varför är detta viktigt att kunna

- Programmeringskunskaper efterfrågas mer och mer. Kanske kommer du att
  - samarbeta med programmerare (som kollega, kund, ...).
  - läsa fler kurser inom programmering.
  - jobba med programmering.
- Förkunskaper till andra kurser, t ex Geografisk informationsteknik.
- Problemlösning
  - Du kommer att träna på att lösa problem och formulera algoritmer - kunskaper du kommer att ha nytta av även i andra sammanhang.
- Allmänbildning
  - Vad är egentligen ett datorprogram?
  - Vad menas med algoritm?
  - Demokratiproblem om bara några enstaka känner till detta.

Undervisning

- Programmering:
  - Föreläsningar 14 st
  - Datorlaborationer (obligatoriska) 11 st på 14 tillfällen
  - Övningsuppgifter (finns på Moodle)
  - Resurstider
  - Delmålskontroll (läsvecka 3)
- Databaser:
  - Föreläsningar 2 st
  - Laborationer (obligatoriska) 2 st
  - Seminarieövning (obligatorisk) 1 st

Examination

- Skriftlig tentamen
- Kursen inrapporteras i Ladok i tre delar
  - Programmering, obligatoriska moment 3 hp
  - Databaser, obligatoriska moment 1.5 hp
  - Programmering, tentamen 3 hp

Kursbok och kursmaterial

- Allen B. Downey & Chris Mayfield: Think Java.
  `http://greenteapress.com/wp/think-java/`
  - Kan läsas på nätet (interaktiv version), laddas ner (pdf, gratis) eller köpas (bok).
- Övrigt kursmaterial:
  - Föreläsningsbilderna – finns i veckoschemat på Moodle
  - Laborationsuppgifter
  - PM om programutvecklingsmiljön Eclipse
  - Övningsuppgifter – finns på Moodle
  - Java snabbreferens (tillåtet hjälpmedel på tentan)

Allt material finns på `cs.lth.se/edaa20` eller `moodle.cs.lth.se`

CEQ-synpunkter

- Överlag nöjda studenter
- Några CEQ-kommentarer från förra kursomgången.
  - ”Laborationerna var mycket intressanta och givande!”
  - ”Det är kul att kunna programmera men det är svårt.”
  - ”Labbarna tar alldeles för mycket tid. Det krävdes mycket tid att förbereda, och även om man kom dit mycket förbered så hann man ändå inte klart.”
  - ”Kämpigt att läsa denna kurs på en läsperiod då det tar lång tid att komma in i det annorlunda programmeringstänket.”
- På CEQ-mötet framkom synpunkter på schemat.

Kommunikation

- Webbsida: `http://cs.lth.se/edaa20`
  - Information om kursen (kursplan, schema, extentor ...)
  - Anmälan till laborationsgrupper
- Moodle: `moodle.cs.lth.se`
  - Veckoschema med läsanvisningar
  - Övningsuppgifter
- E-post:
  - Kursansvarig: `niklas.fors@cs.lth.se`
  - Mail kommer vid behov att skickas till din student-mail.
- Frågor, problem?
  - Tveka inte att ställa frågor (på föreläsningarna, på resurstiderna, laborationer, även mellan undervisningstillfällena ...)

Om att lära sig programmera

- Programmering är kul!
- Programmering tar tid (fundera ut lösningar, hitta fel ...).
- Man lär sig programmera genom att träna. Skriv program, testa, hitta på egna program, experimentera ...
  - Men inte bara - glöm inte bort teorin.
- Starten är viktig. Häng med från början! Det man lär sig kommer att byggas på och användas genom hela kursen.
- De olika undervisningsmomenten (labbar, övningsuppgifter, föreläsningar ...) finns av en anledning och kompletterar varandra.

Program

Ett datorprogram är stegvisa instruktioner till datorn som beskriver hur ett problem ska lösas.

- I instruktionerna manipuleras information (data)
- Instruktionerna skrivs i ett programspråk, t ex Java.

Exempel på ett Javaprogram

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

Duy behöver inte alls förstå alla detaljer i detta program. Men kanske kan du ana att texten ”Hello world!” skrivs ut på datorns skärm.

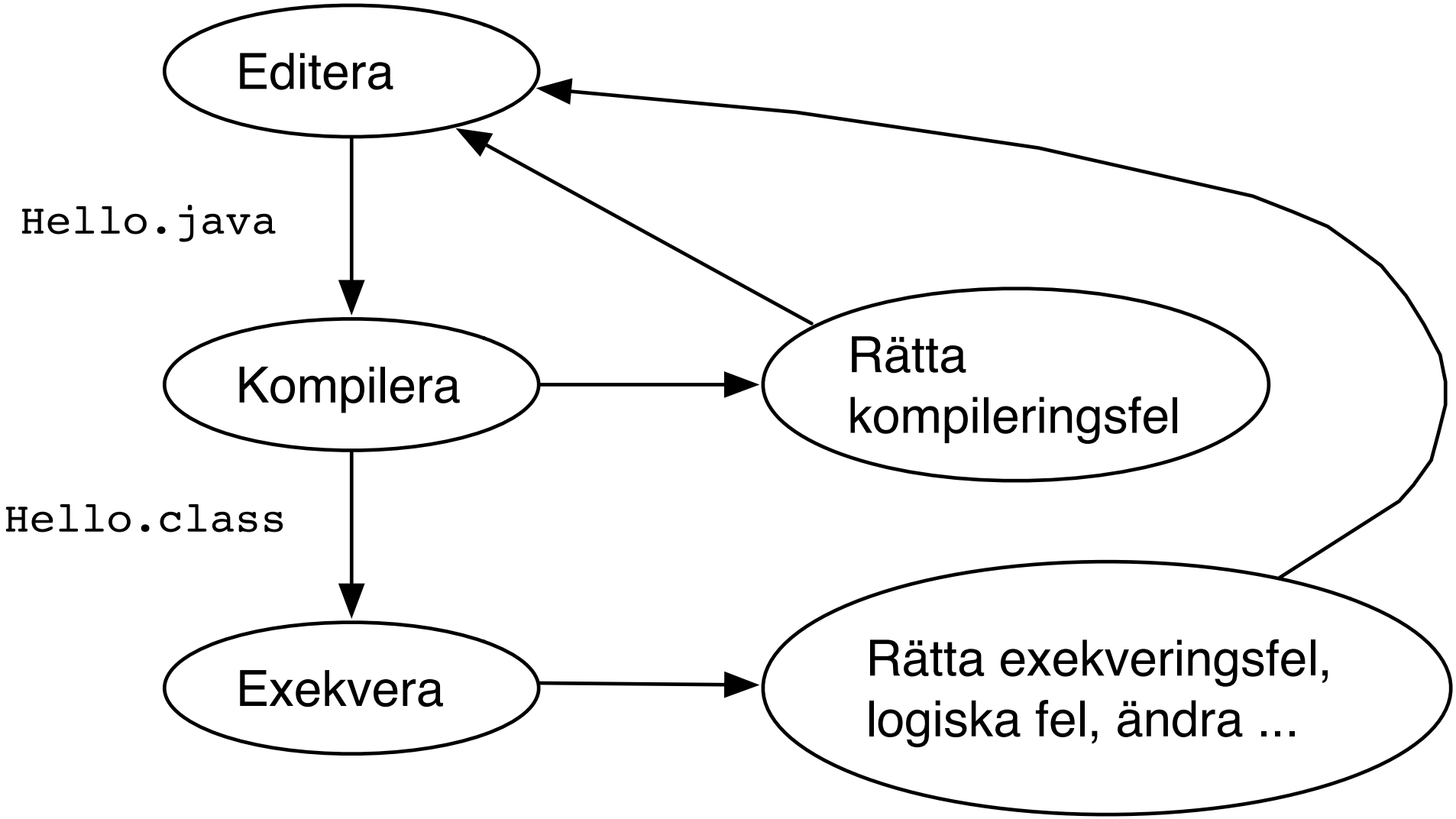
Programkod

- Programmet på förra bilden är skrivet i Java.
  - Java är ett högnivåspråk (kan läsas och förstås av oss människor).
  - Denna kod kallas källkod (eng. source code) och lagras i en fil med tillägget `.java`.
- Innan programmet kan köras (exekveras) måste det översättas till enklare instruktioner som går att köra i datorns processor (CPU).
  - Java-programmet översätts (kompileras) till bytekod som sedan interpreteras i processorn.
  - Denna kod lagras i en fil med tillägget `.class`.

Editera - kompilera - exekvera

- **Editera**
  - Skriv programmet (källkoden) i en fil vars namn har tillägget `.java`.
- **Kompilera**
  - Programkoden kontrolleras. Om koden är korrekt (dvs. inte bryter mot språkets grammatik) översätts den till bytekod. Bytekoden lagras i en fil vars namn har tillägget `.class`.
- **Exekvera**
  - Programmet körs, dvs. bytekodsinstruktionerna i programmet utförs.

Editera - kompilera - exekvera

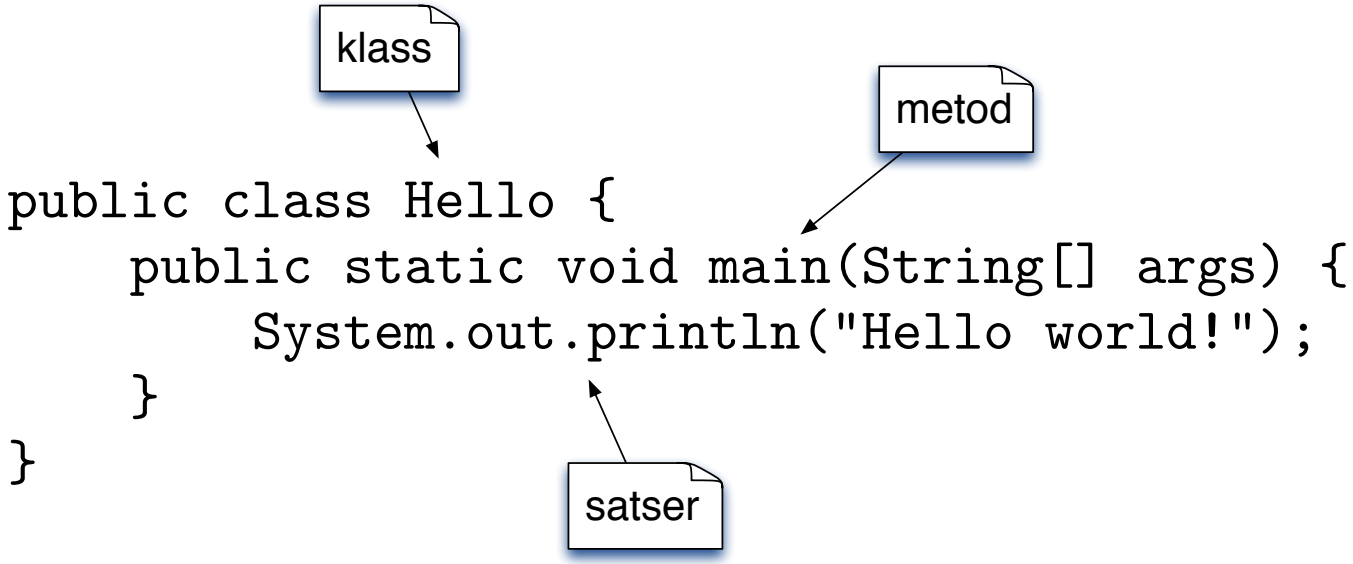


Programmet delas upp i klasser och metoder

- Större datorprogram måste delas upp i hanterliga delar. Man kan inte hålla tusentals programrader i huvudet samtidigt!
- I Java delar man upp programmet i **klasser**.
- En klass kan innehålla flera **metoder**.
- En metod är en sekvens av **satser**.

Exempel på ett Javaprogram

Metoden `main`



- Det måste finnas en klass med en `main`-metod. Satserna i `main`-metoden utförs i tur och ordning när programmet exekveras. (I exemplet finns bara en enda sats.)

Att skriva program

- Att skriva program innebär bl.a. att
  - sätta sig in i det problem man ska lösa.
  - dela upp problemet i mindre delar.
  - sätta sig in i färdigskriven kod som ska felsökas eller ändras.
  - skissa på en lösning (på papper, på dator, svenska, pseudokod).
  - välja bland färdiga klasser att använda i programmet.
  - välja bland färdiga algoritmer för att lösa ett delproblem (t.ex. söka bland data) eller formulera egna algoritmer.
  - skriva programkod.
- Man skriver *inte* programmet rad för rad uppifrån och ned.
- Tips! Utgå från ett litet program (som programmet Hello) och ändra och utöka efter hand. Då har du alltid ett program som fungerar. Testa ofta. Använd debugger för att felsöka *och* för att förstå vad som händer i programmet.

Algoritm

En algoritm är en följd av instruktioner som beskriver hur man ska göra för att stegvis lösa ett problem.

Exempel:

- matrecept
- beskrivning för att montera en möbel
- beskrivning (på svenska eller pseudokod) hur man löser ett delprolem, t.ex. hitta minsta, beräkna summa, söka ...
- programkod
- ...

Exempel på en algoritm

- Problem:

  Beräkna summan av talen 12.5 och 0.7 och skriv ut summan.

- Algoritm:

```
Tal nr 1 är 12.5
Tal nr 2 är 0.7
Summan = tal nr 1 + tal nr 2
Skriv ut summan
```

Exempel på programkod

- Problem:

  Beräkna summan av talen 12.5 och 0.7 och skriv ut summan.

- Programkod:

```
double nbr1 = 12.5;
double nbr2 = 0.7;
double sum = nbr1 + nbr2;
System.out.println("Summan av talen är " + sum);
```

Exempel på ett Javaprogram

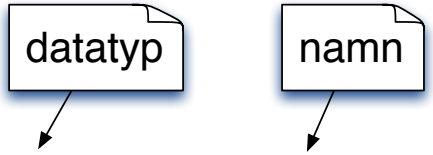
```
public class Calculator {
    public static void main(String[] args) {
        double nbr1 = 12.5;
        double nbr2 = 0.7;
        double sum = nbr1 + nbr2;
        System.out.println("Summan av talen är " + sum);
    }
}
```

Programmet beräknar beräknar summan av två tal och skriver ut summan på datorns skärm.

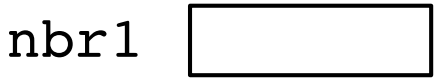
Variabel

Variabler används för att lagra värden som programmet måste komma ihåg.

- Variabler är ett namngivet utrymme i datorns minne.
- Variablerna måste deklareras.
  - Variabler har ett **namn**.
  - Variabler har en **datatyp** som talar om vilka slags värden som kan lagras i variabeln.



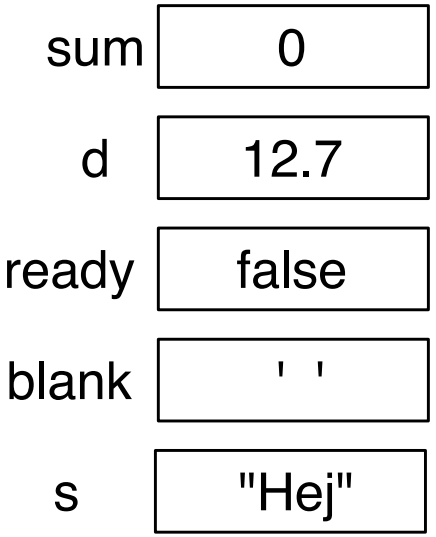
Exempel: `double nbr1;`



Datatyper

- De data som behandlas i ett program är av olika typer, t.ex. heltal, flyttal, teckensträngar ...
- Exempel:

```
int sum = 0;
double d = 12.7;
boolean ready = false;
char blank = ' ';
String s = "Hej";
```



Primitiva datatyper

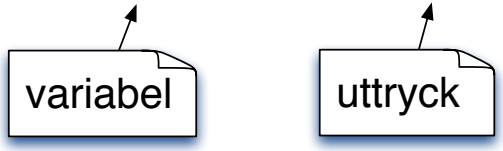
Primitiva datatyper

- börjar med gemen (liten bokstav).
- innehåller ett enda värde, ett heltal, ett flytttal, ett tecken eller ett logiskt värde.

byte	heltal (-128...+127)
short	heltal (-32768...+32767)
int	heltal (-2 147 483 648...+2 147 483 647)
long	heltal (ca -9*10^{18}... +9*10^{18})
float	flyttal tal (tal med decimaldel)
double	flyttal tal (tal med decimaldel)
boolean	logiska värden (`true` eller `false`)
char	tecken

Tilldelningssats

- I en **tilldelningssats** ger man en variabel ett nytt värde.
  Exempel: `nbr1 = 12.5;`



- Deklaration och tilldelning kan göras i samma sats.
  Exempel:

```
double nbr1 = 12.5;
double nbr2 = 0.7;
double sum = nbr1 + nbr2;
```

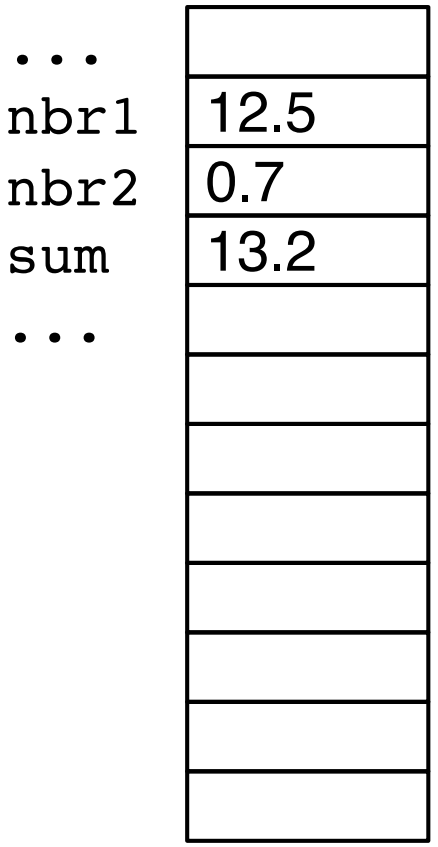
nbr1	12.5
nbr2	0.7
sum	13.2

- Variabeln och uttrycket måste ha samma typ (några undantag finns).
- I Java betyder = tilldelas (inte lika med).

Variabler i datorns minne

Variablernas värden lagras i datorns primärminne. Minnet är uppdelat i olika minnesceller.

- Vid deklarationen av en variabel reserveras det plats för variabeln i minnet.
- Vid tilldelning läggs variabelns värde i motsvarande minnescell/er.
- När en variabel används hämtas dess värde från minnet.

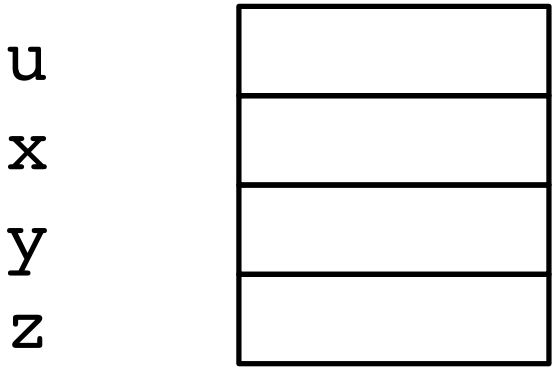


Övning

Tilldelningssatser

Vilka värden har variablerna när följande satser exekverats:

```
int u, x, y, z;
x = 10;
y = 2 * x + 1;
z = (y + x) + (y - x);
x = x + 1;
```



Aritmetiska uttryck

- Uttryck talar om hur värden ska beräknas.

Aritmetiska operatorer:

+	addition
−	subtraktion
*	multiplikation
/	division
%	rest vid heltalsdivision

- Exempel:

```
2 * (x + y) - 10
```

Heltalsdivision och rest

- `a / b` ger ett heltal som resultat om både `a` och `b` är heltal.
- Exempel:
  `17 / 3` är lika med 5 (**heltalsdivision**)
  `17 % 3` är lika med 2 (**rest vid heltalsdivision**)
- 3 personer ska rättvist dela på 17 kakor. Alla får 5 kakor var och 2 kakor blir över.

Övning

Heltalsdivision och rest

Komplettera följande programkod så att variabeln `lastDigit` tilldelas sista siffran i `nbr`.

```
int nbr = 2547;
int lastDigit =
```

Konvertering mellan `int` och `double`

- Ett `int`-värde ”får plats” i en variabel av typen `double`:

```
double d = 42;
```

- Tvärtom går inte utan konvertering eller avrundning:
  - Konvertering till `int`:

```
double d = 42.87;
int a = (int) d;   // talet avkortas till 42
```

  - Avrundning till närmaste heltal:

```
double d = 42.87;
int  b = (int) Math.round(d);   // talet avrundas till 43
```

    Metoden `round` ger ett resultat av typen `long` som måste konverteras till typen `int`.

Objektorientering

- Större program innehåller flera klasser och flera metoder.
  - Det finns många färdiga klasser och metoder i Java.
  - Andra klasser skriver man själv.
- Java är ett objektorienterat språk.
  - Ett program kan ses som en modell av verkligheten.
  - När man delar upp programmet i klasser utgår man från de verkliga saker (**objekt**) som finns i problemet.
  - I programmet skriver man **klasser** som beskriver dessa objekt. Klasserna innehåller variabler som beskriver objekten samt metoder för att arbeta med dem.
- Exempel: I ett program som hanterar kvadrater kan det finnas en klass `Square` med en metod `getArea` innehållande satser för att beräkna kvadratens area.

Exempel på program med objekt

```
public class DrawSquare {
    public static void main(String[] args) {
        SimpleWindow w = new SimpleWindow(600, 600, "Square");
        Square sq = new Square(20, 10, 40);
        sq.draw(w);
        System.out.println(sq.getArea());
    }
}
```

Försök lista ut vad som händer när man kör programmet? Du behöver inte alls förstå alla detaljer ännu. Vi kommer till detta senare i kursen.

Programmet Calculator

Förbättrad version

```
import java.util.Scanner;

public class Calculator {
    public static void main(String[] args) {
        System.out.println("Skriv två tal");
        Scanner scan = new Scanner(System.in);
        double nbr1 = scan.nextDouble();
        double nbr2 = scan.nextDouble();
        double sum = nbr1 + nbr2;
        System.out.println("Summan av talen är " + sum);
    }
}
```

Ny version av programmet `Calculator` där vi kan summera två godtyckliga tal som användaren skriver på tangentbordet.

Inläsning från tangentbordet

- Skapa först en `Scanner`-objekt som ska användas för att läsa från tangentbordet:

```
Scanner scan = new Scanner(System.in);
```

- Med `scan.nextDouble()` läser man nästa tal (av typen `double`) som skrivs på tangentordet:

```
nbr1 = scan.nextDouble();
nbr2 = scan.nextDouble();
```

Import-sats

- Genom att skriva en import-sats först i filen talar vi om att vi kommer att använda en klass från ett klasspaket (en samling redan färdiga klasser).

```
import java.util.Scanner;
```

- I det här exemplet importeras klassen `Scanner` som används för inläsning från tangenbordet.

Utskrift i konsolfönstret

- Utskrift av en teckensträng (omges av citattecken):

```
System.out.println("Skriv två tal");
```

- Utskrift av en variabels värde (OBS! inga citattecken):

```
System.out.println(sum);
```

- Flera saker kan skrivas ut i samma println-sats (med + mellan).

```
System.out.println("Summan av talen är " + sum);
```

Läsliga program

- Detta är ett korrekt Java-program, men ...

```
import java.util.Scanner;public class p {public static void
main(String[] args) {System.out.println("Skriv två tal");
Scanner a=new Scanner(System.in);double b=a.nextDouble();
double c=a.nextDouble();double d=b+c;System.out.println(
"Summan av talen är "+d);}}
```

- Tänk på att skriva läsliga program:
  - Välj vettiga namn på klasser och variabler.
  - Klassnamn ska starta med versal (stor bokstav) och namn på variabler med gemen (liten bokstav).
  - Gör ordentliga indragningar så att man ser vilka satser som hör till vad.

Alternativ – if

Exempel

Vad händer om man kör följande satser:

```
Scanner scan = new Scanner(System.in);
int nbr1 = scan.nextInt();
int nbr2 = scan.nextInt();
if (nbr2 != 0) {
    System.out.println("Kvoten mellan talen är " + (nbr1 / nbr2));
} else {
    System.out.println("Du försökte dividera med 0.");
}
```

Alternativ – if

Mönster

```
if (villkor) {
    satser;     // utförs om villkoret är uppfyllt
}
```

```
if (villkor) {
    satser;     // utförs om villkoret är uppfyllt
} else {
    satser;     // utförs annars
}
```

```
if (villkor 1) {
    satser;     // utförs om villkor 1 är uppfyllt
} else if (villkor 2) {
    satser;     // utförs annars om villkor 2 är uppfyllt
}  else {
    satser;     // utförs annars
}
```

Övning

if-sats

- Komplettera `if`-satsen så att texten ”negativt”, ”0”, ”positivt” skrivs ut beroende på det inlästa talets värde.

```
Scanner scan = new Scanner(System.in);
int n = scan.nextInt();
if (n < 0) {
    System.out.println("negativt");
```

Repetition ett bestämt antal gånger – for

Exempel

Vad händer om man kör följande satser:

```
Scanner scan = new Scanner(System.in);
int sum = 0;
for (int  i = 1; i <= 5; i++) {
    sum = sum + scan.nextInt();
}
System.out.println(sum);
```

Repetition ett bestämt antal gånger – for

Mönster

- Utförs för i-värdena start, start+1, start+2, ..., slut.

```
for (int i = start; i <= slut; i++) {
   satser;
}
```

Övning

for-sats

- Skriv satser för att skriva ut 50 asterisker (*):

- Skriv satser för att beräkna och skriva ut summan
  $1 + 2 + 3 + 4 + ... + 10$

Summering

Algoritm

- Uppgift: Beräkna summan av ett antal termer.
- Lösning: Deklarera och nollställ en variabel som håller reda på det aktuella värdet av summan. Gå sedan igenom termerna en efter en och addera varje term till summan.
- Algoritmen i pseudokod:

```
sum = 0;
för alla termer {
    sum = sum + term;
}
```

Repetition ett okänt antal gånger – while

Exempel

```
Scanner scan = new Scanner(System.in);
int sum = 0;
while (scan.hasNextInt()) {
    sum = sum + scan.nextInt();
}
System.out.println(sum);
```

Repetitionen avslutas när något som inte kan tolkas som ett heltal läses in.

Repetition ett okänt antal gånger – while

Mönster

- Utförs så länge ett villkor är uppfyllt:

```
while (villkor) {
    satser;
}
```

Logiska uttryck

- Villkoret i `if`-satser och `while`-satser är ett **logiskt uttryck**.
  - Logiska uttryck har värdet `true` eller `false`.
  - Villkoret kan bestå av flera delvillkor. Exempel:
    `n >= 1 && n <= 10`

Logiska operatorer:

&&	och
||	eller
!	icke

- I jämförelser mellan t.ex. tal används `<` `<=` `==` `>=` `>` `!=`
  - == betyder ”lika med”.
  - `!=` betyder ”skilt från”.

Diskutera

- Programmet nedan läser ett heltal och skriver ut dess kvadratrot. Vi har skrivit en `main`-metod med satser för att lösa problemet. I dessa satser anropar vi andra metoder som som löser delproblem åt oss.
- Vad heter metoderna som anropas?

```
public class ComputeSquareroot {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.println("Skriv ett tal");
        double nbr = scan.nextDouble();
        double squareroot = Math.sqrt(nbr);
        System.out.println(squareroot);
    }
}
```

Anropa metoder

Argument

- När man anropar en metod ska noll till flera argument skickas med.
- Argumenten är värden som metoden behöver för att fullfölja sin uppgift.
- Antal argument och argumentens typer framgår av metodens dokumentation
- Exempel:

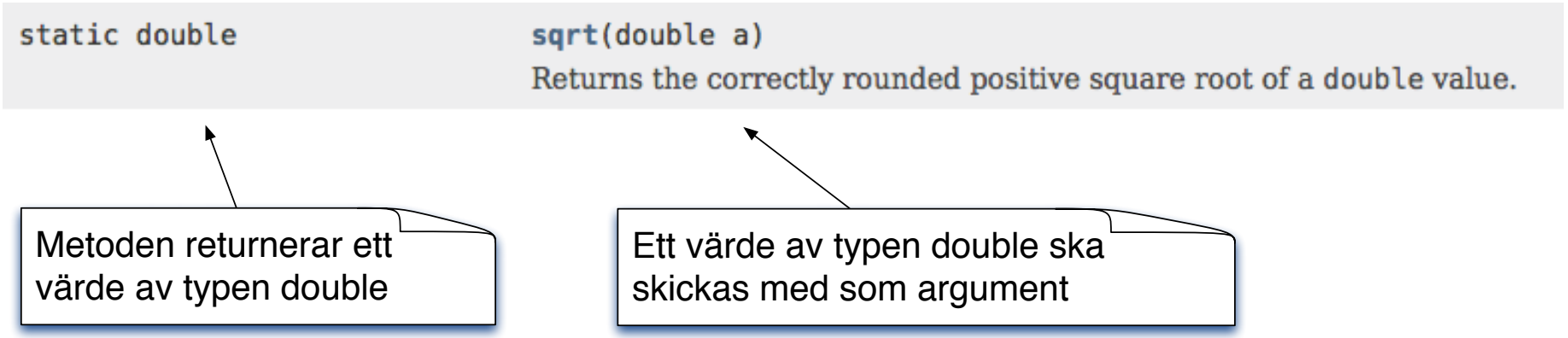
```
double sqrt1 = Math.sqrt(20.5);
double sqrt3 = Math.sqrt(5 + 14.7 * 3);
double nbr = 42.0;
double sqrt2 = Math.sqrt(nbr);
```

Anropa metoder

Dokumentation

- Hur kan man veta att man ska skriva `Math.sqrt(20.5)` för att beräkna $\sqrt{20.5}$?
- Vi måste få reda på
  - att det finns en klass `Math` med en metod `sqrt`.
  - hur många värden, argument, som ska skickas med till metoden.
  - ifall metoden returnerar ett värde (och vilken typ i så fall) eller ej.

- `Math` är en Javas standardklasser och vi kan hitta informationen i Javas dokumentation på nätet:



Diskutera

- Studera anropen av metoderna `nextDouble` och `sqrt`. Likheter/skillnader?

```
public class ComputeSquareroot {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.println("Skriv ett tal");
        double nbr = scan.nextDouble();
        double squareroot = Math.sqrt(nbr);
        System.out.println(squareroot);
    }
}
```

Anropa metoder

- Vissa metoder kan anropas utan att man har skapat något objekt:

```
double sqrt1 = Math.sqrt(20.5);
```

  - Dessa metoder kallas statiska metoder.
  - Före punkten står klassnamnet.

- Andra metoder anropas på objekt:

```
Scanner scan = new Scanner(System.in);
double nbr = scan.nextDouble();
```

`double`	**`nextDouble()`** Scans the next token of the input as a `double`.

Övning

- I klassen Math finns följande metod:

```
static double      pow(double a, double b)
                   Returns the value of the first argument raised to the power
                   of the second argument.
```

Skriv sats/er för att beräkna och skriva ut 4.2 upphöjt till 5.

Klassen Random

- En annan användbar klass är `Random` som har metoder för att generera slumptal:

`int`	**`nextInt(int bound)`** Returns a pseudorandom, uniformly distributed `int` value between 0 (inclusive) and the specified value (exclusive), drawn from this random number generator's sequence.
`double`	**`nextDouble()`** Returns the next pseudorandom, uniformly distributed `double` value between `0.0` and `1.0` from this random number generator's sequence.

- Exempel:

```
Random rand = new Random();
if (rand.nextDouble() < 0.5) {
    System.out.println("krona");
} else {
    System.out.println("klave");
}
```

Övning

Kasta tärning

Komplettera koden med sats/er för att dra och skriva ut ett slumptal i intervallet [1,6].

```
Random rand = new Random();
```

Övning

Kasta tärning - repetition

Komplettera koden med satser för att kasta tärning tills det blir en 6:a (dvs. dra ett slumptal i intervallet [1,6] till resultatet blir 6). Antal kast som krävdes ska räknas och skrivas ut.

```
Random rand = new Random();
```

Checklista

Exempel på vad du ska kunna

- Förklara begreppen:
  - algoritm, program, källkod, kompilering, exekvering
  - variabel, datatyp, tilldelningssats
  - metod, argument
- Förstå att programkod delas upp i klasser, metoder, satser
- Skriva enkla program:
  - deklarera variabler och tilldela dem värden
  - läsa in värden från tangentbordet, skriva ut på skärmen
  - använda if-, for- och while-satser
  - läsa dokumentation och anropa metoder
  - använda metoder i `Random` för att dra slumptal
- Editera, kompilera och exekvera program