Objekt-Orienteret Programmering

Primitive Typer

Aslak Johansen asjo@mmmi.sdu.dk
Peter Nellemann pmn@mmmi.sdu.dk

September 6, 2023

SDU

Part 0:
Hello, World

Klasser

Java programmer består af én eller flere "klasser". Hver klasse repræsenterer noget funktionalitet.

Hver klasse er i udgangspunktet gemt i en fil (der er navngivet efter klassen):

Klassen `Person` er gemt i filen `Person.java`

Funktionalitet i en klasse gøres tilgængeligt igennem *metoder*.

Hvis en klasse har en `main` metode kan den afvikles.

Lad os tage et eksempel ...

Hello, World

> *"When the first caveman programmer chiseled the first program on the walls of the first cave computer, it was a program to paint the string 'Hello, world' in Antelope pictures. Roman programming textbooks began with the 'Salut, Mundi' program. I don't know what happens to people who break with this tradition, but I think it's safer not to find out."*
> – The Linux Kernel Module Programming Guide (2007-05-18 ver 2.6.4)

```
public class HelloWorld
{
    public static void main (String[] args) {
        System.out.print("Hello, World");
    }
}
```

Hello, World

> *"When the first caveman programmer chiseled the first program on the walls of the first cave computer, it was a program to paint the string 'Hello, world' in Antelope pictures. Roman programming textbooks began with the 'Salut, Mundi' program. I don't know what happens to people who break with this tradition, but I think it's safer not to find out."*
> – The Linux Kernel Module Programming Guide (2007-05-18 ver 2.6.4)

```
public class HelloWorld
{
    public static void main (String[] args) {
        System.out.print("Hello, World");
    }
}
```

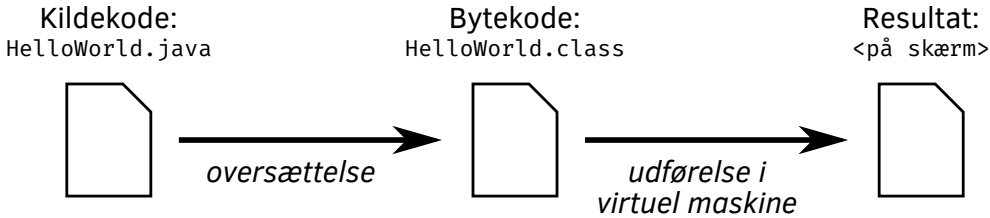
Hello, World

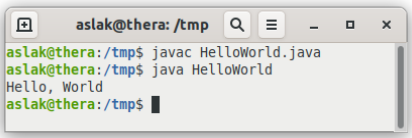
> *"When the first caveman programmer chiseled the first program on the walls of the first cave computer, it was a program to paint the string 'Hello, world' in Antelope pictures. Roman programming textbooks began with the 'Salut, Mundi' program. I don't know what happens to people who break with this tradition, but I think it's safer not to find out."*
> – The Linux Kernel Module Programming Guide (2007-05-18 ver 2.6.4)

```
public class HelloWorld
{
    public static void main (String[] args) {
        System.out.print("Hello, World");
    }
}
```

Fra Kildekode til Afvikling

Hvad sker der egentligt, når I afvikler jeres javaprogram?





Part 1:
Datatyper

Primitive Datatyper

Datatyper bruges til at beskrive "typer af data".

Data er ofte noget der bliver produceret (eller ændret) når vi kører et program. Vi bruger typen til at beskrive rummet af gyldige værdier for data.

Java har 8 primitive datatyper. Det er de første typer vi skal beskæftige os med.

Datatype	Mulige værdier:	Plads
`byte`	Heltal: -128 til 127	8 bit
`short`	Heltal: -32768 til 32767	16 bit
`int`	Heltal: -2147483648 til 2147483647	32 bit
`long`	Heltal: -9223372036854775808 til 9223372036854775807	64 bit
`float`	Decimal tal: 7 betydende cifre†	32 bit
`double`	Decimal tal: 15 betydende cifre†	64 bit
`char`	Tegn: A, i, ?, 2, 0, k, _, . osv. *Et unicode tegn**	16 bit
`boolean`	true / false	8 bit

Datatype Øvelser

Opsummering:

- Heltal (4 stk)
- Kommatal (2 stk)
- Tegn (1 stk)
- Boolean (1 stk)

Hvad passer til at beskrive:

- Højden af jeres forelæser?
- Antallet af studerende i denne klasse?
- Om man er kommet til tiden?
- Ens navn?
- Hvor mange jordbær der er per deltager til en børnefødselsdag?

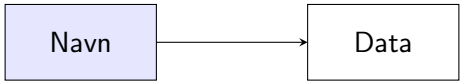
Part 2:
Variable

Variable ▷ Definition

En variabel er den konstruktion man bruger til at *navngive* et stykke data.

Værdien af de data som variabel peger på kan ændres* ... via navnet.

Variable ▷ Regler



Syntaksregler for navnet:

- Begynder med et bogstav eller en underscore*.
- Kan bestå af tal, bogstaver og/eller underscores.

Dataene:

- Har en datatype.

I Java knyttes navnet til en bestemt datatype.

Variable ▷ Erklæring

Hver gang vi skriver og afslutter en "linje" kode i Java har vi lavet et *statement**.

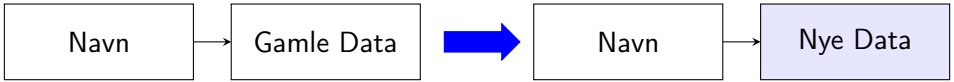
Syntaks:
⟨*type*⟩⟨*variable*⟩;

Eksempel:

```
double value;
```

Variable ▷ Tildeling

Ved tildeling overskrives det datarum som variablen peger på med en (ny) værdi:



Syntaks:
⟨*variable*⟩ = ⟨*expression*⟩;

Eksempel:

```
value = 42;
```

Bemærk: Dette kræver at variablen allerede er erklæret.

Variable ▷ Erklæring og Tildeling

Erklæring og tildeling kan udtrykkes igennem et enkelt *statement*.

Syntaks:
⟨*type*⟩⟨*variable*⟩ = ⟨*expression*⟩;

Eksempel:

```
int value = 42;
```

Variable ▷ Eksempler

```
double price       = 0.0;
int    pageCount   = 773;
int    year        = 2022;
int    authorCount = 1;
```

Part 3:
Statements og Expressions

Statements og Expressions ▷ Statements

Et *statement* er enheden af et trin der skal udføres.

Statements skrives typisk på en linje for sig selv afsluttet af et semikolon.

Statements udføres én af gangen i rækkefølge.

Eksempel på en sådan sekvens:

```
double height = 1.84;
int
count;
count = 1;
```

Statements og Expressions ▷ Expressions

Et *expression* er en stump af kode der evaluerer til en værdi.

Vi kan bruge matematiske operatorer til at opbygge expressions:

Navn	**Betydning**	**Eksempel**	**Resultat**
+	Addition	34+1	35
-	Subtraktion	34.0-0.1	33.9
*	Multiplikation	300*30	9000
/	Division	1.0/2.0	0.5
%	Rest	20%3	2

Statements og Expressions ▷ Tildelinger

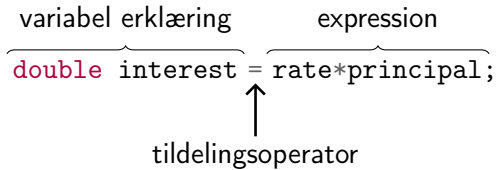
Vi kalder "=" for en type af tildelingsoperator (eng: assignment operator): Det er en operator, der kan give en variabel en ny værdi.

Der findes andre typer af tildelingsoperatorer (mathematically augmented assignments) der fungerer som *shorthands* for hyppige anvendelser:

Operator	**Navn**	**Eksempel**	**Ækvivalent**
`+=`	Addition tildeling	`i += 8`	`i = i + 8`
`-=`	Subtraktion tildeling	`i -= 8`	`i = i - 8`
`*=`	Multiplikation tildeling	`i *= 8`	`i = i * 8`
`/=`	Division tildeling	`i /= 8`	`i = i / 8`
`%=`	Rest tildeling	`i %= 8`	`i = i % 8`

Statements og Expressions ▷ Kombinationer

Vi kan kombinere en variabel erklæring, en tildelingsoperator og et expression til et statement:



Hermed opnår vi:

- Erklæring af en variabel.
- Tildeling af den værdi som er resultatet af evalueringen af et expression.
  - Værdien af dette expression afhænger af værdierne af to andre variable.

Part 4:
Eksempler

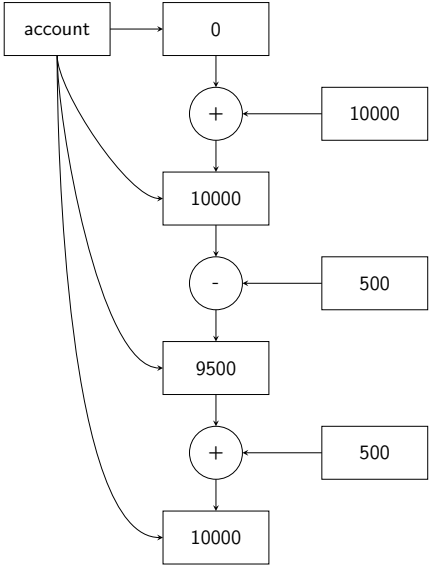
Eksempel ▷ Konto

Kontekst: Variablen `account` bruges til at representere en bankkonto.

Variablens livscyklus:

1. **Oprettelse af konto**.
2. **Indsættelse af løn: +10.000,00 kroner**.
3. **Køb af kursusbog: -500,00 kroner**.
4. **MobilePay overførsel for vens kursusbog: +500,00 kroner**.

```
double account = 0; ←
account += 10000; ←
account -= 500; ←
account += 500; ←
```



Eksempel ▷ Arealberegning

```
public class ComputeArea {
  public static void main (String[] args) {
    double radius; // declare radius
    double area;   // declare area

    // assign a radius
    radius = 20; // radius is now 20

    // compute area
    area = 3.14159 * radius * radius;

    // display results
    System.out.println("The area for the circle of radius "
                       + radius + " is " + area);
  }
}
```

Eksempel ▷ Kommentarer

```
/*
 * this is called a block comment
 */

/* comments do not affect the execution of code */

// this comment ends with the line
```

(Skriv på engelsk i koden – man skriver kun kode på engelsk)

Part 5:
Type Casting

Automatisk Typecasting

Gyldigt cast:

```
int    i = 2;
double d = i;
```

Ugyldigt cast:

```
double d = 2.0;
int    i = d;
```

Hvorfor?

Oversætteren checker om den kan garantere at operationen kan udføres uden tab af information, og dén arbejder udelukkende på typer.

Eksplicit Typecasting

Løsning:

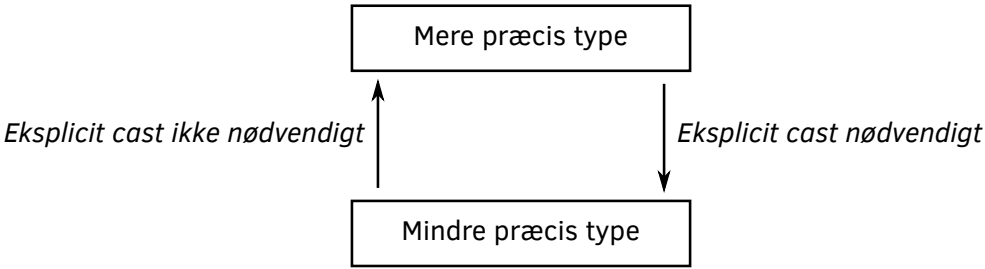
```
double d = 2.0;
int    i = (int)d;
```

Bemærk: Ekplicit typecasting kan koste præcision.

- Ikke nødvendigvis som man intuitivt tænker: $7.96 \rightarrow 7$

Ved at foretage et eksplicit cast fortæller man oversætteren at man påtager sig ansvaret for sådanne tab af præcision.

Typecasting Reglen



Part 6:
Identifiers

Identifiers

Navne på "ting" i programmer kaldes identifiers.

Ting kan fx være klasser og variable.

De må ikke starte med cifre.

Identifiers

Visse navne er reserveret i programmeringssprog.

- Eksempler fra Java: `if`, `while`, `for`, `public`, `static`, `void`, `else` ...

Data lagres i maskinens hukommelse på lokationer med numeriske adresser hvis der ikke er plads til dem i processoren.

- Identifiers bruges til at tilgå disse data symbolsk i stedet for manuelt at skulle holde styr på de specifikke lokationer.

Eksempler på identifiers: `$2`, `ComputeArea`, `area`, `radius`, `print`, `_test`

CaSe SeNsItIvItY

CaSe SeNsItIvItY

CaSe SeNsItIvItY

Brug IKKE retard case!

Case Sensitivity

Der er forskel på STORE og små bogstaver i visse sprog ... som fx Java. Vi siger at disse sprog er *case sensitive*

Der er "god skik" for hvordan ting navngives:

- Klasser starter med stort.
- Variable starter med småt.

Tilsvarende skelner mange filsystemer også case sensitive. Hvis du sidder på et filsystem der ikke er case sensitive, så betyder dét at noget virker på din maskine ikke nødvendigvis at det virker på andres. Sørg derfor for at copy-paste filnavne og kommandoer.

Part 7:
De Sidste Detaljer

(Post|Pre) (Increment|Decrement) Operators

Increment – noget forøges
Decrement – noget formindskes
Pre – Ændringen foretages **før** værdien udtrækkes
Post – Ændringen foretages **efter** værdien udtrækkes

Eksempel:

```
int counter = 5;
counter = counter + 1;
counter++;
counter--;
```

Hvad printer følgende kode (hvis vi tilføjede den enkelte linje til ovenstående)?

1. `System.out.println(counter);`
2. `System.out.println(++counter);`
3. `System.out.println(counter++);`

Opsummering

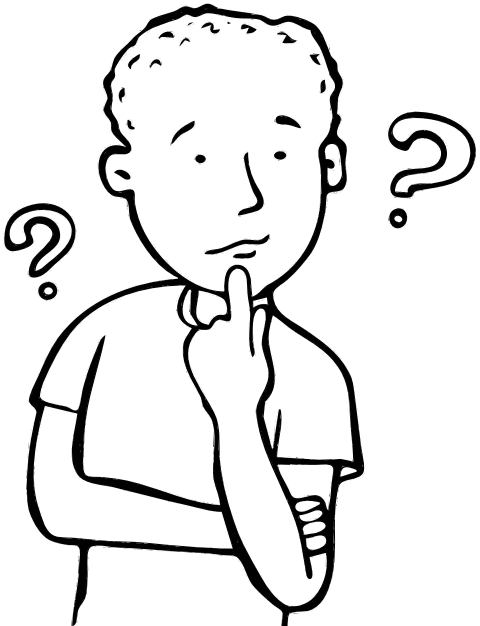
```java
/**
 * This class implements a simple program that will compute the amount of interest that is earned on $17,000 invested at
 * an interest rate of 0.027 for one year. The interest and the value of the investment after one year are printed to
 * standard output.
 */
public class Interest {
  public static void main(String[] args) {
    /* Declare the variables. */
    double principal; // The value of the investment.
    double rate; // The annual interest rate.
    double interest; // Interest earned in one year.

    /* Do the computations. */
    principal = 17000;
    rate = 0.027;
    interest = principal * rate; // Compute the interest.
    principal += interest;

    // Compute value of investment after one year, with interest.
    // (Note: The new value replaces the old value of principal.)
    /* Output the results. */
    System.out.print("The interest earned is $");
    System.out.println(interest);
    System.out.print("The value of the investment after one year is $");
    System.out.println(principal);
  } // end of main()
} // end of class Interest
```

Fra bogen "Introduction to Programming Using Java" af David J. Eck (lettere modificeret).

Spørgsmål?



https://openclipart.org/detail/236687/boy-thinking-of-question