

Klasser

Grundlæggende Programmering med Projekt

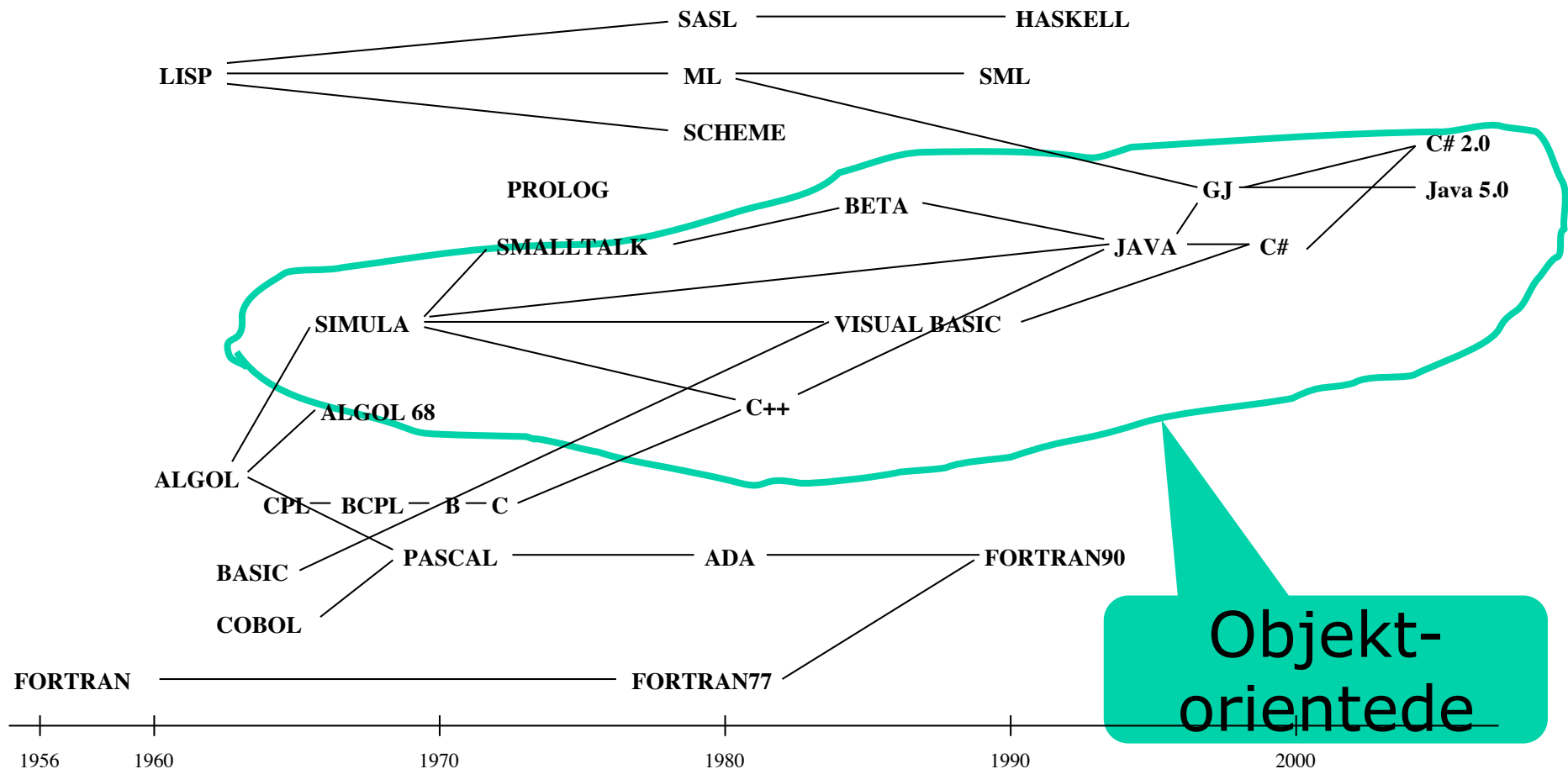
Dan Witzner Hansen

Sidste gang?

Dagens begreber

- Felt (field)
- Metode (method)
- Parameter (parameter)
- Sætning, ordre (statement)
- Tildeling (assignment statement)
- Betinget sætning (conditional statement)

Programmeringssprogs historie



Objekt-orienteret programmering

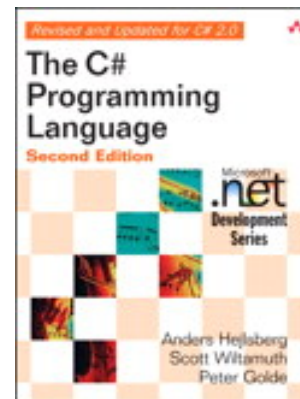
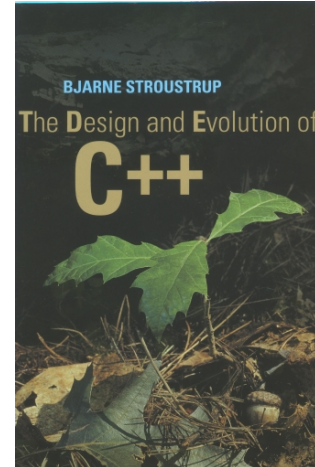
Historisk set

- Simula 67, ca 1966
 - Ole Johan Dahl og Kristen Nygaard
 - Norsk Regnesentral
 - IEEE John von Neumann Medal
 - ACM Turing Award
- Smalltalk, ca 1970
 - Alan Kay m.fl.
 - Xerox PARC
- Beta, ca 1980
 - Ole L. Madsen m.fl.
 - Aarhus Universitet



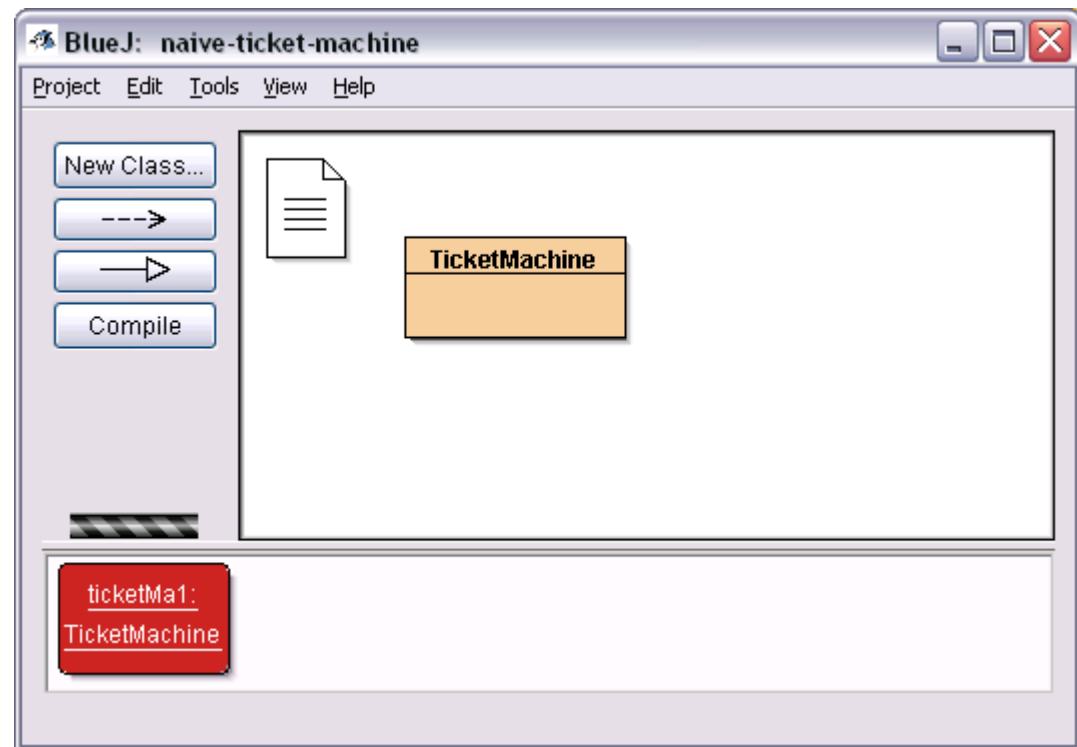
Nyere OOP sprog

- C++, ca 1980
 - Bjarne Stroustrup
 - Aarhus, Cambridge, Bell labs
- Java, ca 1995
 - James Gosling
 - SUN Microsystems
- C#, ca 2000
 - Anders Hejlsberg
 - København, Californien, Microsoft



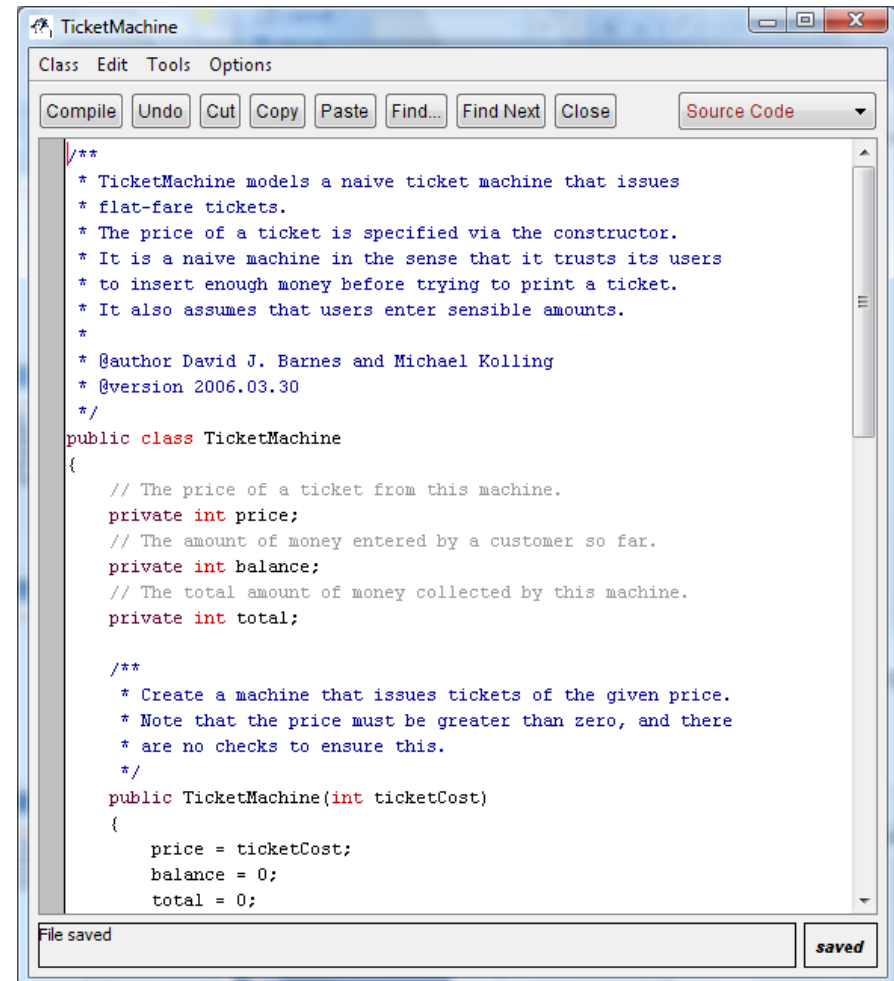
Billeteautomat – set udefra

- En billeteautomat laver billetter til en fast pris
 - Hvordan er prisen fastlagt?
 - Hvordan får man penge ind i maskinen?
 - Hvordan holder maskinen styr på pengene?
 - Hvordan får man en billet?



Billetautomat – set indefra

- Ved at læse kildeteksten kan vi se hvad programmet gør
- Vi kan se hvordan det er *implementeret*
- Alle Java klasser har samme struktur



```
/**
 * TicketMachine models a naive ticket machine that issues
 * flat-fare tickets.
 * The price of a ticket is specified via the constructor.
 * It is a naive machine in the sense that it trusts its users
 * to insert enough money before trying to print a ticket.
 * It also assumes that users enter sensible amounts.
 *
 * @author David J. Barnes and Michael Kolling
 * @version 2006.03.30
 */
public class TicketMachine
{
    // The price of a ticket from this machine.
    private int price;
    // The amount of money entered by a customer so far.
    private int balance;
    // The total amount of money collected by this machine.
    private int total;

    /**
     * Create a machine that issues tickets of the given price.
     * Note that the price must be greater than zero, and there
     * are no checks to ensure this.
     */
    public TicketMachine(int ticketCost)
    {
        price = ticketCost;
        balance = 0;
        total = 0;
    }
}
```


Anatomien for en klasse

```
public class TicketMachine {  
    ... erklæringer ...  
}
```

Erklæring af
klasse
TicketMachine

```
public class ClassName {  
    ... felter ...  
    ... konstruktorer ...  
    ... metoder ...  
}
```

Generel form
af klasse-
erklæring

Felter

- Et felt indeholder en *værdi* for et objekt
- Kaldes også *instansvariabel*
- Højreklik og “Inspect” i BlueJ til at se objekts felter

```
public class TicketMachine
{
    private int price;
    private int balance;
    private int total;

    ...
}
```

synlighed

type

feltnavn

```
private int price;
```

Konstruktører

- En konstruktor initialiserer et objekt
- Kaldes når man laver et objekt med “**new**”
- En konstruktor er en speciel metode som
 - har samme navn som klassen
 - ikke har nogen returværdi
- Konstruktoren bruges typisk til
 - at initialisere objektets felter
 - ved brug af værdier givet som parametre

```
public TicketMachine(int ticketCost)
{
    price = ticketCost;
    balance = 0;
    total = 0;
}
```

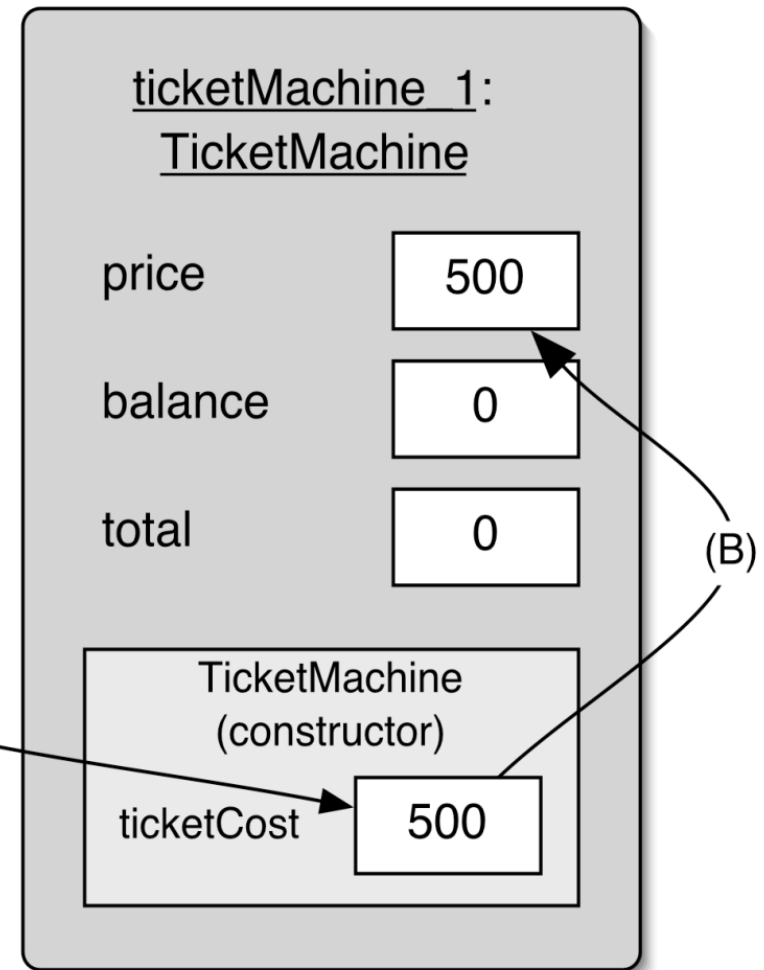
... og hvordan gør man så det?

BlueJ: Create Object

// Create a machine that issues tickets of the given price.
// Note that the price must be greater than zero, and there
// are no checks to ensure this.
TicketMachine(int ticketCost)

Name of Instance:

new TicketMachine(**)**



Tildeling (Assignment)

- En værdi kan gemmes i et felt ved hjælp af en tildelingssætning, fx

```
price = ticketCost;
```

```
felt = udtryk ;
```

Udregn udtryk og gem resultatet i **felt**

- Et felt gemmer kun en enkelt værdi
– dvs. den gamle værdi *overskrives*

```
price = 42;  
price = ticketCost;
```

42 gemmes i price

men bliver straks
overskrevet

Metoder

- En metode har
 - en *signatur*: navn, returtype og parametre
 - en *implementation* (metodekrop) med sætninger

```
public int getPrice()  
{  
    return price;  
}
```

Signatur

Krop

- Signaturen afgør hvordan metoden kaldes
- Kroppen afgør hvilken effekt metoden har

Udskrift i billetautomaten

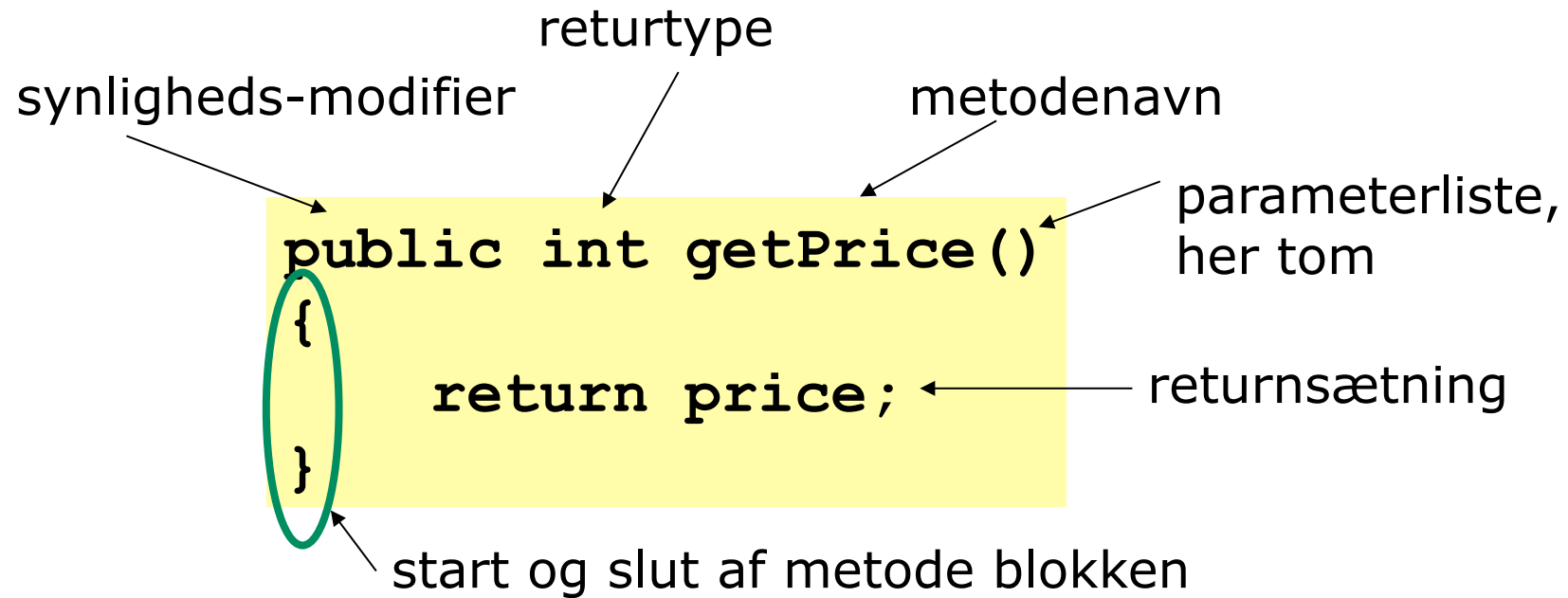
```
public void printTicket()
{
    // Simulate the printing of a ticket.
    System.out.println("#####");
    System.out.println("# The BlueJ Line");
    System.out.println("# Ticket");
    System.out.println("# " + price + " cents.");
    System.out.println("#####");
    System.out.println();

    // Update the total collected with the balance.
    total = total + balance;
    // Clear the balance.
    balance = 0;
}
```

Accessor- og mutator-metoder

- Accessor metoder
 - giver informationer om et objekts tilstand
 - typisk returnerer værdien af et felt eller en simpel beregning
 - tager typisk ingen parametre
 - typisk navn: `getEtEllerAndet()`
- Mutator metoder
 - bruges til at ændre et objekts tilstand
 - ændre typisk et eller flere felter
 - tager typisk parametre
 - har typisk ingen return sætning
- Nogle metoder er både accessor og mutator

Accessor metode



Mutator metode

visibility modifier return type metode navn parameter

```
public void insertMoney(int amount)
{
    balance = balance + amount;
}
```

felt som ændres tildeling

Efter Forlæsning: Gruppe arbejde

Miniøvelser

- 5. Ed

B&K 2.23, 2.25, 2.26, 2.27, 2.28, 2.29

B&K 2.37, 2.38, 2.39, 2.42

... men er der ikke noget galt med billetautomaten?

- Kan man indsætte negative beløb?
- Kan man få en billet uden at betale fuld pris?
- Kan man få penge tilbage?
- Er det rimeligt at man kan sætte billetprisen til at være negativ fra start?
- Hvordan kan vi gøre det bedre?

Betingede sætninger

- Hvis en betingelse er sand, så gør en ting, ellers gør noget andet:

```
public void insertMoney(int amount)
{
    if (amount > 0) {
        balance = balance + amount;
    } else {
        System.out.println("Use a positive amount: " + amount);
    }
}
```

Betingede sætninger, if-else

if-
sætning

logisk udtryk
(boolean)

sætninger der udføres
hvis betingelsen er sand

```
if (betingelse) {  
    Gør dette hvis betingelsen er sand  
} else {  
    Gør dette hvis betingelsen er falsk  
}
```

sætninger der udføres
hvis betingelsen er falsk

Lokale variable

- Et felt `private int balance;`
 - gemmer en værdi i hele objektets levetid
 - kan tilgås fra alle objektets metoder
 - eksisterer sammen med objektet
- En *lokal variabel* `int amountToRefund;`
 - erklæres i en metode eller konstruktor
 - eksisterer kun indenfor denne metode
 - dvs. kan ikke tilgås fra andre metoder

Lokale variable versus felter

```
public class TicketMachine
{
    private int balance;
    ...
```

felt

Ingen
modifier!

```
    public int refundBalance()
    {
        int amountToRefund;
        amountToRefund = balance;
        balance = 0;
        return amountToRefund;
    }
```

lokal variabel

```
    public int otherMethod() {
        ... kan ikke se amountToRefund ...
    }
    ...
}
```


Navnekonventioner i Java

- Klasser har stort begyndelsesbogstav:
`Circle, Triangle, ...`
- Felter, metoder og lokale variable har lille:
`price, balance, total, name, ...`
- Sammensatte navne har "camel case"
`getPrice, ticketCost, TicketMachine...`
- Konstruktorer har samme navn som klassen:
`Circle, Triangle, TicketMachine, ...`

Opsummering

- Klasser rummer
 - felter
 - konstruktorer
 - metoder
- Et felt gemmer en værdi
- Konstruktorer initialiserer objektet når det skabes
- Metoder implementerer objektets opførsel
- Felter, parametre og lokale variable er alle variable
 - Felter lever i hele objektets levetid
 - Parametre bruges til at modtage værdier i en konstruktor eller metode
 - Lokale variable bruges til mellemregninger
- ‘If’ sætninger bruges til at lave betinget opførsel

Udtryk eller sætning?

- Et ***udtryk*** har en værdi, fx

`x+2*y`

- En ***sætning*** har ingen værdi, kun effekt, fx

```
if (...) { x=2; } else { x=3; }
```

```
System.out.println("Godnat");
```

Udtryk opbygges af ...

- Konstanter, fx
`117 false "abc" 3.14159`
- Variable og felter, fx
`balance`
- Metodekald, fx
`machine1.getBalance()`
- Aritmetiske operatorer
`+ - * / %`
- Sammenligningsoperatorer
`== != < > <= >=`
- Logiske operatorer
`! && ||`

Nogle matematiske funktioner (metoder, faktisk)

Java	Matematik
<code>Math.pow(x, y)</code>	x^y
<code>Math.exp(x)</code>	e^x
<code>Math.log(x)</code>	$\ln(x)$
<code>Math.log10(x)</code>	$\log(x)$
<code>Math.sqrt(x)</code>	\sqrt{x}
<code>Math.sin(x)</code>	$\sin(x)$
<code>Math.asin(x)</code>	$\sin^{-1}(x)$
<code>Math.abs(x)</code>	$ x $

Microsoft software fra MSDNAA

- Microsoft Developer Network Academic Alliance giver adgang til:
 - Windows XP, Windows Vista, ...
 - Visual Studio 2008, .NET, ...
 - MS Project, MS Visio, ...
- Men **ikke MS Office** (Word, Excel, osv)
 - bortset fra MS Access som kan bruges i projektet
- I får en mail med login fra sysadm
- I stedet for MS Office kan man fx bruge OpenOffice, fra <http://www.openoffice.org/>
- Det er **ikke nødvendigt at piratkopiere**

Næste uge

- Til tirsdag
 - Læs B&K kapitel 2
 - Læs i B&K kapitel 3 men spring det over der virker uforståeligt
- Til Torsdag
 - Start på B&K kapitel 4