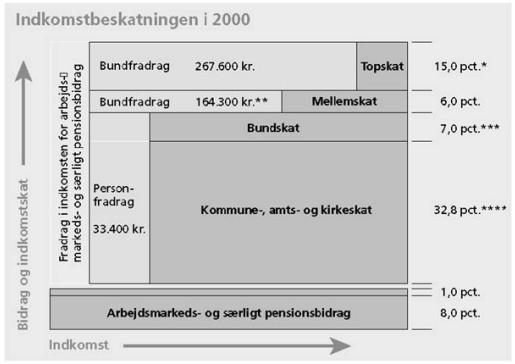
Kursus 02199: Programmering
afsnit 2.1 - 2.7

Anne Haxthausen

IMM, DTU

1. Værdier og typer (bl.a. **char, boolean, int, double**) (afsnit 2.4)
2. Variable og konstanter (afsnit 2.3)
3. Sætninger (bl.a. assignments) (afsnit 2.3)
4. Udtryk (bl.a. aritmetiske) (afsnit 2.4)
5. Data konvertering (afsnit 2.4)
6. Objekter og klasser (bl.a. `String`) (afsnit 2.1, 2.2, 2.5, 2.6)
7. Udskrift til skærmen (afsnit 2.1)
8. Indlæsning fra skærmen (afsnit 2.7)

Eksempel: Skat i år 2000



Kilde: *FORSTÅ Skatten...*, Skatteministeriet

Eksempel: Beregning af AMBI og særligt pensionsbidrag

```
public class Skat1 {
  public static void main(String[] args) {
    int indkomst = 120000;
    double ambi, pension;

    ambi = indkomst * 8.0 / 100.0;
    pension = indkomst * 1.0 / 100.0;

    System.out.print("Arbejdsmarkedsbidrag: ");
    System.out.println(ambi);
    System.out.print("Særlig pensionsopsparing: ");
    System.out.println(pension);
  }
}
```


Uddata fra Skat1 programmet

```
Arbejdsmarkedsbidrag: 9600.0
Særlig pensionsopsparing: 1200.0
```

Grundbegreber: værdi og type

En *værdi* (eller data) kan f.eks. være

- et heltal: `120000`
- et kommatal: `8.0`
- et tegn: `'!'`
- en sandhedsværdi: `false` eller `true`
- en tegnstreng: `"Bundskat: "`

En *(data) type* er en familie af værdier og operationer på disse.

Typer i Java

`byte, short, int, long` er typer af heltal: ..., -2, -1, 0, 1, 2, **`...`**

`float, double` er typer af kommatal: f.eks. -32.3, 1.0, 42.456, , 4.5E6, **`...`**

`char` er typen af tegn: `'a', ..., '1', ..., '!', ..., '\n', ...`

`boolean` er typen af sandhedsværdier: `true`, `false`

`String` er typen af tegnstrenge: `"Bundskat: "`, `"Ole-Johan"`, , **`...`**

`String` er ikke en primitiv type, men en såkaldt klasse. Vi vender tilbage til dette.

Tal

I datamaten repræsenteres hvert tal ved et binært tal, dvs. kombinationer af bits (0'er og 1'er).

F.eks. repræsenteres tallet 4 ved det binære tal 100.

Forskellen på de numeriske typer er, hvor megen plads, der afsættes til at gemme værdier af deres type i lageret. Typen **int** bruger f.eks. 32 bit.

Tegn

I datamaten repræsenteres hvert tegn ved et heltal, f.eks. 65 for 'A'.

En *tegntabel* er en 'oversættelse' fra kode (heltal i maskinen) til tegn (grafik på skærmen eller papiret).

Standard tegntabeller: ASCII (i USA), ISO Latin1 (i Vesteuropa), Unicode (hele verden).
Java benytter Unicode.

Opgave: hvilken type har følgende værdier?

Værdi	Type
`58`	
`true`	
`-23`	
`"afd "`	
`42.0`	
`'$'`	
`"42.0"`	
`"true"`	
`'7'`	

Grundbegreber: variable, erklæringer og tildeling

En *variabel* kan indeholde en værdi af en given type.

En variabel svarer til et sted i datamatens lager.

Du navngiver selv dine variable.

En *erklæring* indfører type og navn(e) for en eller flere variable:

```
double ambi, pension;
```

Erklæringen afsætter også plads af til variablene i datamatens lager.

En variabelerklæring kan indeholde en *initialisering*:

```
int indkomst = 120000;
```

En *tildelings-ordre* ændrer en variabels værdi ved at skrive i datamatens lager:

```
ambi = indkomst * 8.0 / 100.0;
```


Grundbegreb: konstanter

En *konstant* minder om en variabel, men den har samme værdi hele tiden.

Eksempel på erklæring:

```
final double AMBI_PROCENT = 8.0;
```

Eksempel på brug:

```
ambi = indkomst * AMBI_PROCENT / 100.0;
```


Grundbegreb: sætninger

Tilstanden består af datamatens lagerindhold, uddata (på skærmen), osv.

En *sætning* ændrer tilstanden.

Eksempel 1: en tildelings-sætning kan ændre en variabels værdi:

```
pension = indkomst * 1.0 / 100.0;
```

Eksempel 2: en print-sætning kan skrive uddata på skærmen:

```
System.out.print("Særlig pensionsopsparing: ");
```

Grundbegreb: udtryk

Et *udtryk* beregner/angiver en værdi.

Eksempel:

```
indkomst * 8.0 / 100
```

Der findes udtryk af forskellig typer, f.eks.:

- aritmetiske udtryk (beregner numeriske værdier)
- logiske udtryk (beregner logiske værdier)
- streng udtryk (beregner strenge)

Aritmetiske udtryk

Er kombinationer af aritmetiske *operatorer* og *operander*.

Operator	Betydning	Eksempler	
*	Multiplikation	`1.5 * 60.0`	`24 * 60`
/	Division	`13.0 / 2.0`	`13 / 2`
%	Rest	`13.0 % 2.0`	`13 % 2`
+	Addition	`1.1 + 60.0`	`14 + 60`
-	Subtraktion	`1.1 - 60.0`	`134 - 60`

Resultattypen er `int` hvis begge argumenter er `int`, ellers `double`.

Læs selv om operator præcedence.

Logiske udtryk

Forklares i kapitel 3 (næste gang).

Streng udtryk

Ligesom man kan lave aritmetiske udtryk, der angiver tal værdier,
kan man lave udtryk, der angiver tegnstrenge.

En vigtig streng operator:

- streng konkatenering ($+$)

Eksempel:

- `"Dette er "+"kursus 02199"`
  (angiver tegnstrengen `"Dette er kursus 02199"`)

Typer af udtryk

Udtryk har en type.

Eksempler:

- $1 + 2$ har typen **int**
- `"Dette er " + "kursus 02199"` har typen `String`

Grundbegreb: data konvertering

Værdier af en type kan (i visse tilfælde) konverteres til en anden type:

1. Konverteringer mellem numeriske typer:
   (a) "widening": fra en mindre type til en større type (f.eks. fra **int** til **double**)
   (b) "narrowing": fra en større type til en mindre type (f.eks. fra **double** til **int**)
2. Konvertering fra enhver type til `String` typen

I Java kan 1a og 2 ske *automatisk*, men 1b kun *explicit* ved *casting*.

Eksempler på implicit widening konvertering

```
double money; int dollars;

// tildelingskonvertering:
dollars = 25;
money = dollars; //nu er money == 25.0

// ulovlig tildeling:
dollars = money;

//2. argument (2) af / konverteres til en double (2.0):
money  = money / 2; //nu er money == 12.5

//Her konverteres 2 ikke
money = 25.0;
money  = (int) money / 2; //nu er money == 12.0
```


Eksempler på narrowing konvertering ved casting

```
money = 84.69;
// casting, som smider decimaler væk:
dollars = (int) money; //nu er dollars == 84
```

Eksempel på konvertering til tegnstreng

```
int x = 7;
System.out.println("indholdet␣af␣x␣er:␣" + x);
```

2. argument af + konverteres automatisk til tegnstrengen `"7"`, før den konkateneres med 1. argument.

Udskrift på skærmen bliver: `"indholdet af x er: 7"`

Generelt gælder for `S + V` og `V + S`,
hvor `S` er en tegnstreng og `V` et udtryk af en anden type,
at `V` automatisk konverteres til sin tekstrepræsentation.

Opgave: hvilken værdi og type har følgende udtryk

Udtryk	Udtrykkets værdi	Udtrykkets type
`1.5 * 60.0`		
`1.5 * 60`		
`24 * 60`		
`1.1 + 60 - 1`		
`150.0 / 60`		
`150 / 60`		
`134.0 % 60`		
`134 % 60`		
`"02199"`		
`"x er lig med "+ "0"`		
`"x er lig med "+ 0`		

Kort introduktion til klasser og objekter

Uformelt

- En *klasse* repræsenterer et begreb: tidspunkt, aftale, bil, ko, person, . . .
- Et *objekt* repræsenterer en ting, en instans af et begreb: et bestemt tidspunkt, en bestemt bil, en bestemt ko, en bestemt person, . . .
- En klasse har tilhørende *metoder*, der er de operationer (funktioner), der kan udføres på dens objekter.

I programmeringssproget Java

- En *klasse* er en type, svarende til `int`, `double`, `boolean`, . . .
- Et *objekt* er en værdi, ligesom `17`, `18.01`, `false`, . . .
- En *metode* er en operation, ligesom $+$, $-$, . . .

Tre trin i brugen af klasser, objekter og metoder

1. Definer en *klasse* (inkl. dens metoder).
2. Skab objekter af klassen.
3. Brug objekternes metoder.

Nogle klasser (`String`) er allerede defineret i et klassebibliotek. Så kan trin 1 springes over.

Hvis en klasse har *statiske* metoder, kan disse bruges uden trin 2-3.

Definition af klasser

Hvordan man *definerer* en klasse, lærer vi senere.
(Hver metode gives et navn og en række sætninger, der udføres, når den *kaldes*.)

Nu vil vi kun *bruge* klasser fra biblioteker.
For hver klasse er for hver metode beskrevet:
navn, argument typer og en resultat type, og hvad der sker, nå den *kaldes*.

Eksempel: Uddrag af interface for `String`

int `length()` angiver længden af strengen

char `charAt(int index)` angiver tegnet på plads `index`

Skabelse af objekter

Et objekt er en stump data (f.eks. en tegnstreng) i maskinens lager.

Et objekt kan have tilknyttede metoder.

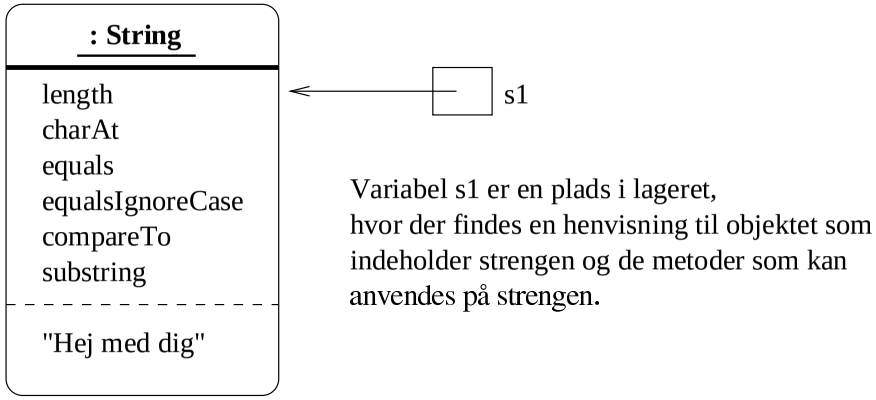
En variabel indeholder enten

- en primitiv værdi, eller
- en *henvisning* til et objekt.

Eksempel: et objekt af klassen String

```
String s1 = "Hej med dig";
```


Eksempel: Et objekt af klassen String



Variabel s1 er en plads i lageret,
hvor der findes en henvisning til objektet som indeholder strengen og de metoder som kan anvendes på strengen.

Brug af metoder

Kald af et objekts metoder gøres med *dot operatoren*.

Eksempel: kald af en `String` metode

```
s1.length();
```

giver værdien `11` (når objektet, `s1` henviser til, har værdien `"Hej med dig"`).

Klassebiblioteker og pakker

Et *klassebibliotek* er en mængde klasser som understøtter programudvikling.

Klasserne placeres i forskellige *pakker* (eng. packages).

De mest benyttede pakker er:

Package	Understøtter	Klasser
`java.lang`	Generel programmering; er *automatisk* importeret	`Math`, `String`, ...
`java.io`	Input og output fra/til omgivelserne (f.eks. filer)	...
`java.util`	Generelle hjælpeklasser	`Random`, ...

Ønsker man at bruge de klasser, der ligger i en pakke uden at bruge pakkenavnet, skal pakken *importeres*, f.eks.:

```
import java.util.*; // alle klasser i pakken
import java.util.Random; // kun Random klassen
```


Udskrift til skærmen

Metoderne

- `System.out.print`
- `System.out.println`

kan bruges til udskrive tekst på skærmen.

Indlæsning af data fra tastaturet

Bogens `Keyboard` klasse har metoder til indlæsning:

```
static boolean readBoolean()
static byte    readByte()
static char    readChar()
static double  readDouble()
static float   readFloat()
static int     readInt()
static long    readLong()
static short   readShort()
static String  readString()
```

static betyder, at metoderne kan kaldes via deres klassenavn, f.eks. `Keyboard.readBoolean()`

Klassen kan hentes fra kursets hjemmeside og cd, samt fra bogens cd.

Eksempel på indlæsning (og udskrift)

```
import cs1.Keyboard;

public class Question
{
  public static void main (String[] args)
    {
      String name; int cars;

      System.out.print("Hvad er dit navn? ");
      name = Keyboard.readString();

      System.out.print("Hvor mange biler ejer du, " + name + "? ");
      cars = Keyboard.readInt();

      if (cars>1)
         System.out.println(name + " ejer mange biler!");
}   }
```

Kørsel af Question

```
Hvad er dit navn? Anne
Hvor mange biler ejer du, Anne? 1

Hvad er dit navn? Henrik
Hvor mange biler ejer du, Henrik? 2
Henrik ejer mange biler!
```