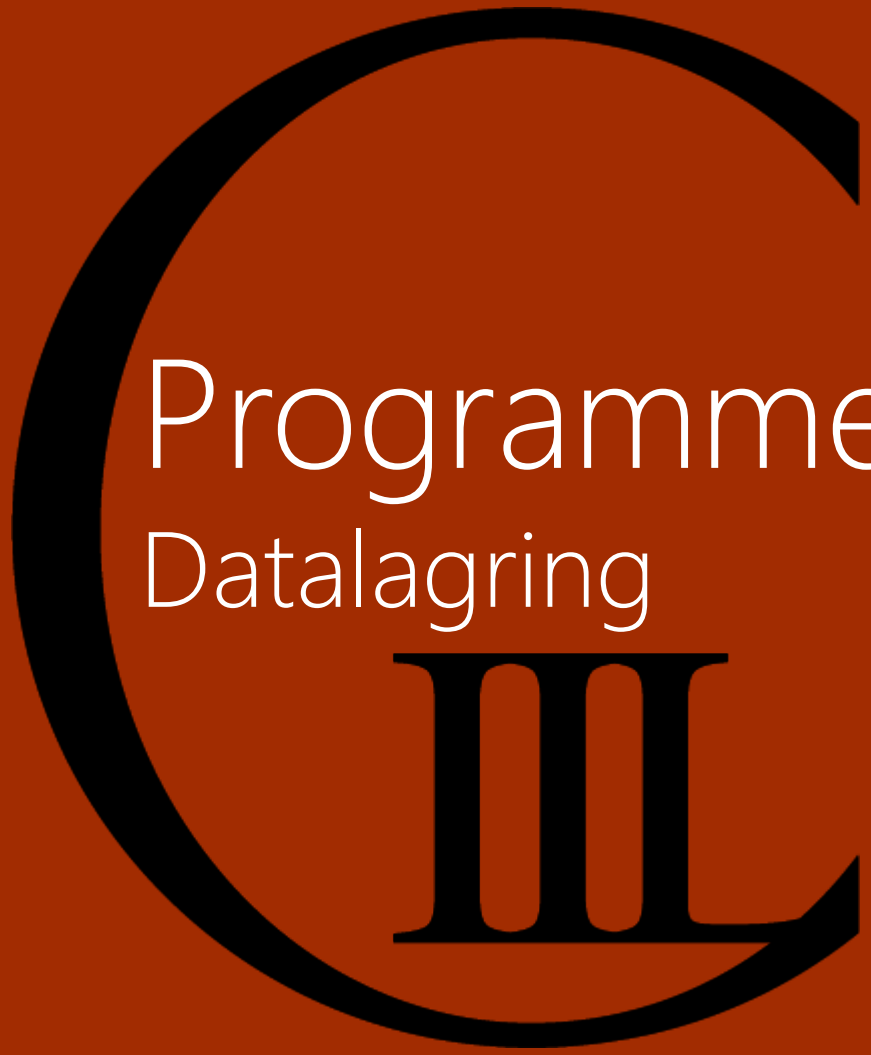


Programmering 1

Datalagring

Datalagring

Variabler

- I ett program skapar man en mängd av variabler av olika datatyper och lagrar olika uppgifter i dessa variabler, kallas variabels värde.
- Så här skapa man variabeln x, och tilldela den till 20:

```
int x = 20;     // skapa variabeln x, och initiera den till 20
```

- Det går också att först skapa en variabel och senare lagrar ett värde till den. Man kan göra så här:

```
int x;    // skapa variabeln x (deklarera eller införa)
x = 20;   // tilldela värdet 20 till variabeln
```

- Det är bäst att skapa varje variabel i sin egen sats, eftersom det är lätt att läsa och förstå.

```
int x = 15;
int y = 25;
```

- Man kan i vissa fall skapa flera variabler i en och samma sats.

```
int x, y;
x = 15;
y = 25;
```

- Det går också att skapa flera variabler och tilldela värde i en och samma sats.

```
int x = 15, y = 25;
```

Datalagring

Namn på variabler

- Ett program kan göras mer lättbegripligt genom att man anger vad en viss variabel representerar när man skapar den.
- Ett bättre sätt att göra ett program mer förståeligt är att variablerna använder lämpliga namn.
- Här är några exempel:

```
int x = 15; // antalet lärare

// antalet elever
int y = 150;

int antalet_elever = 150;
int antaletElever = 130;
```

Man kan tilldela ett specifikt värde till en variabel, eller värdet av en annan variabel.

```
int x = 15;      // initiera variabeln x med 15
int y = x;       // initiera y med x:s värde

// eller så här
int m = 15;      // initiera variabeln m med 15
int n;           // skapa variabel n
n = m;           // kopiera m:s värde till n
```

Konstanter

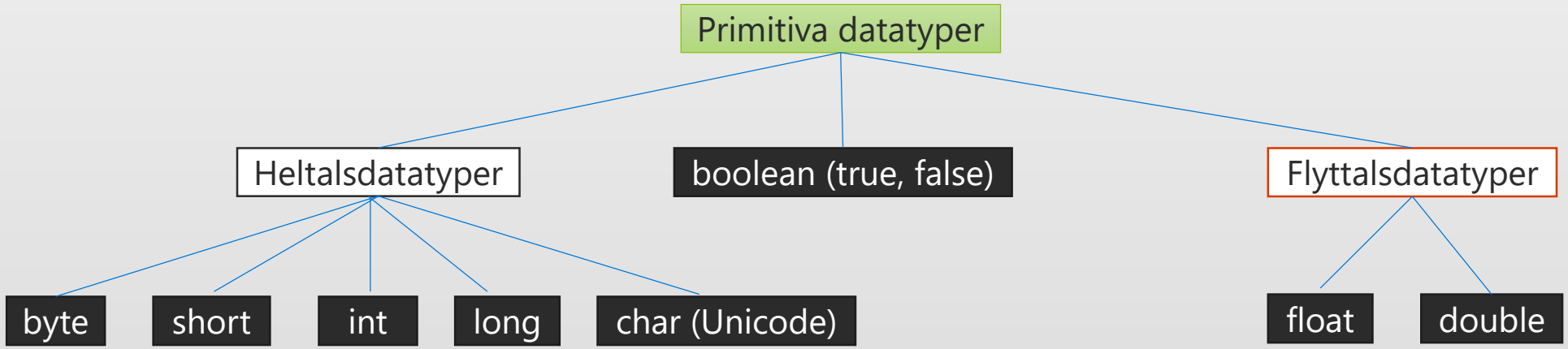
I vissa fall representerar en variabel ett värde som inte är avsett att ändras. För att undvika misstag kan man göra så här:

```
final int antalet_elever = 30;
```

Datalagring

Datatyper i Java

Det finns exakt åtta grundläggande primitiva datatyper i Java:



Datalagring

byte

- En variabel med datatypen `byte` tar upp 1 byte minne. Med denna datatyp kan man representera heltal i intervallet `[-128, 127]`.
- Så här skapar och initierar man en variabel av datatypen `byte`:

```
byte betyg = 5;
```

- Det går inte att lagra ett större tal än vad datatypen tillåter, ty detta ger ett kompileringsfel.

```
byte avstand = 546;
```

Fel, för stort tal för datatypen `byte`

short

- Varje variabel av typen `short` tar upp 2 byte minne. Med denna `short` kan man lagra heltal i intervallet `[-32768, 32767]`.
- Så här skapar och initierar man en variabel av datatypen `short`:

```
short avstand = 546;
```

Datalagring

`int`

- När man lagrar heltal använder man vanligen datatypen `int`. Varje variabel av typen `int` tar upp 4 byte minne.
- Med denna `int` kan man representera alla heltal i intervallet

  `[-2147483684,2147483647].`

- Så här skapar och initierar man en variabel av datatypen `int`:

```
int tal = 123456;
```

`long`

Datatypen `long` tar upp 8 byte minne och kan lagra heltal från intervallet

`[-9223372036854775808, 9223372036854775807].`

När man lagrar ett heltal som `long` måste man ange antingen `L` eller `l` i slutet av heltalet.

- Det är bättre att använda suffixet L, eftersom `l` kan växlas med `1`. Så här gör man:

```
long tal = 9876543210L;
```

Datalagring

`double`

- Datatypen `double` har större noggranhet och kan lagra tal från ett större talområde. Datatypen `double` tar upp 8 byte minne.
- En variabel med datatypen `double` skapas och initieras så här:

```
double d = 12345.01234567;
```

`float`

- Datatypen `float` tar upp 4 byte minne.
- När man lagrar ett tal som float , måste man lägga till `F` eller `f` på slutet av talet.

```
float f = 2.45f;
```

- Om man inte anger `F` (eller `f`) då lagras `2.45` automatiskt som `double`.

Datalagring

E-notation

- Flyttal kan skrivas på två olika sätt. Talet `12500000.0`, till exempel, kan skrivas som `12.5E6` (eller `12.5E+6` eller `12.5e6` eller `12.5e+6`). Det betyder 12.5 gånger 10 upphöjt till 6.
- 0.0000000001 kan skrivas som `1E-9`

```
double d = 1E-9;
```

- samma som:

```
double d = 0.000000001
```

- När ett flyttal av typen `float` anges med E-notationen, läggs till `F` eller `f` på slutet:

```
float f = 1.2E-5F;
```

- Man kan välja att använda E-notation, eller att inte använda den.
- Mycket stora och mycket små tal skrivs ut med E-notation.

Datalagring

char

- Datatypen för tecken. Datatypen `char` är en heltalsdatatyp, och tar upp 2 byte minne.
- I Java lagrar man ett tecken genom att ange tecknets ordningsnummer i Unicode-listan
- Tecken `A` har, till exempel, ordningsnumret `65`, `B 66`, `C 67` och så vidare.

```
char c = 'A';
```

- I stället för att ange teckenet mellan apostrofer, kan man ange tecknets ordningsnummer (kod).

```
char c = 65;  // tecknet A lagras i variabeln c
```

- I vissa sammanhang använder man hexadecimala koder för att presentera olika tecken.
- Tecken A, till exempel, kan representeras på detta sätt:

```
char c = '\u0041'  // A
```

❑ Strängar `Buss` kan skrivas så här:

```
System.out.println ("\u0042\u0055\u0053\u0053");
```

❑ Vad är skillnaden mellan dessa variablerna?

```
char a = '0';
int b = 0;
char c = 0;
```

Datalagring

boolean

- Det finns två booleska värden `true` (sant) och `false` (falskt).
- Man kan skapa och initiera en boolesk variabel så här:

```
boolean b1 = 3 > 5;
System.out.println("b1 = " + b1);

boolean b2 = b1;
System.out.println("b2 = " + b2);
```

Omvandlingar mellan heltalsdatatyper.

- Man kan initiera en större datatyp med en mindre datatyp, eller tilldela en mindre datatyp till en större datatyp.

```
byte a = 5;
short b = a; // initiera en större datatyp
// med en mindre datatyp

short c;
c = a;       // tilldela en mindre datatyp
// till en större datatyp
```

- Men det går inte att tilldela en variabel av en datatyp med ett värde av en större av datatyp.

```
short d = 5;
byte e = d; // kompileringsfel, short är en större datatyp
```

Datalagring

Omvandlingar mellan flyttalsdatatyper.

- Datatypen `float` är mindre än datatypen.
- Därför kan man tilldela ett värde av typen `float` till en variabel av typen `double`. Eller initiera en variabel av typen `double` med ett värde av typen `float`. Så här:

```
float f = 1.25F;
double d = f;
```

- Det går inte att lagra ett värde av typ `double` i en variabel av typen `float`.

```
double u = 1.2;
float v = u;    // kompileringsfel
float w = 2.0;  // kompileringsfel, 2.0 är ett flyttal, varför?
```

Omvandlingar mellan heltalsdatatyper och flyttalsdatatyper.

- Man kan göra så här:

```
long a = 123456L;
float f = a;        // OK!, float är större än long
```

- Flyttal representerar ett större talområde än heltaltyper, men de har en begränsad noggrannhet.

```
long a = 1234567890L;
float x = a;  // lagras som 1.23456794E9
```

- Det krävs speciell syntax om man vill omvandla en större datatyp till en mindre datatyp. Till exempel:

```
short a = 5;
byte b = (byte)a;
```

Datalagring

- Precision kan gå förlorad när man typomvandla en datatyp till en annan datatyp.

```
float f = 25.02F;
int i = (int)f;      // i blir 25

double x = 1.23456789;
float y = (float)x; // y blir 1.2345679
```

- En annan typisk situation när man vill framtvinga typomvandlingen uppstår när man vill få heltalsdelen av ett decimaltal

```
double d = 125.12;
int p = (int)d; // p blir 125
```

- Ett värde kan helt går förlorat vid en typomvandling.

```
short m = 1225;
byte n = (byte)m; // byte kan inte lagra 1225, n blir -55
```

Omvandling mellan tecken och heltal

- Man kan utföra följande typomvandlingar:

```
char c1 = 'A';
int p = c1; // char kan lagras som int
int q = 97;
char c2 = (char) q;  // c2 blir tecknet a
// char c2 = q; // kompileringsfel: int kan inte lagras som char
```

- Det går inte att utföra automatisk typomvandlingar mellan dessa datatyper. Vill man utföra sådana omvandlingar, måste man framtvinga dem. Man kan göra så här:

```
char c1 = 'A';
// byte p = c1; // kompileringsfel
byte p = (byte) c1; // p blir heltalet 65

byte q = 97;
// char c2 = q;  // c2 blir tecknet a
char c2 = (char) q; // c2 blir tecknet a
```

Datalagring

Omvandling mellan tecken och heltal

- Man kan representera negativa heltal med datatypen `byte` och `short`., men inte datatypen `char`.
- Heltalsområdet som kan representeras med datatypen täcks inte av heltalsområdena som kan representeras med datatyperna `short` eller `byte`, och vice versa.
- Därför kam man inte säga att datatypen char är mindre än eller större än datatypen byte eller datatypen short.
- Detta innebär att man inte kan lagra ett värde av typen `char` i en variabel av typen `short` eller `byte`

- Det går inte att utföra automatisk typomvandlingar mellan dessa datatyper.
- Vill man utföra sådana omvandlingar, måste man framtvinga dem. Man kan göra så här:

```
char c1 = 'A';
// byte p = c1; // kompileringsfel
byte p = (byte) c1; // p blir heltalet 65

byte q = 97;
// char c2 = q;  // c2 blir tecknet a
char c2 = (char) q; // c2 blir tecknet a
```

Slut