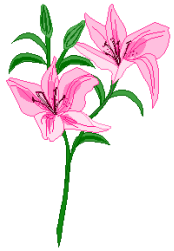
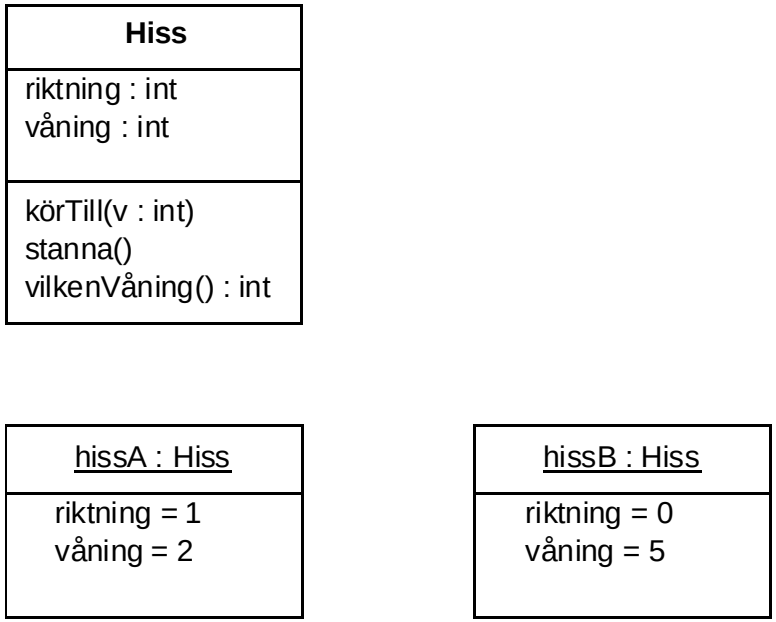


```
class Blomma {
   ...
}
```

```
class Hus {
   ...
}
```

```
class MyFrame
      extends
JFrame {
   ...
}
```


Hiss
riktning : int
våning : int
körTill(v : int)
stanna()
vilkenVåning() : int
hissA : Hiss
riktning = 1
våning = 2
hissB : Hiss
riktning = 0
våning = 5

```
public class Hiss {
  // instansvariabler
  private int riktning;  // -1 neråt, 0 stilla, 1 uppåt
  private int våning;

  // metoder
  public void körTill(int v) {
    ...
  }

  public void stanna() {
    ...
  }

  public int vilkenVåning() {
    ...
  }
}
```

Variabler:

- *lokala variabler inne* i en metod
- *instansvariabler*
- *klassvariabler*
- *parametrar* till metoder

Två kategorier:

- *referensvariabler*
- *enkla variabler*

Deklaration:

```
modifierare typ variabelnamn = initieringsvärde;

typ variabelnamn;     // enklaste form

final typ v = init;  // v får sedan inte ändras
```

Inbyggda numeriska typer			
typ	storlek	minsta värde	största värde
`byte`	8 bitar	-128	127
`short`	16 bitar	-32 768	32 767
`int`	32 bitar	-2 147 483 648	2 147 483 647
`long`	64 bitar	-9 223 372 036 854 775 808	9 223 372 036 854 775 807
`float`	32 bitar	Ungefär $-3{,}4 \times 10^{38}$ 7 siffrors noggrannhet	Ungefär $3{,}4 \times 10^{38}$ 7 siffrors noggrannhet
`double`	64 bitar	Ungefär $-1{,}7 \times 10^{308}$ 15 siffrors noggrannhet	Ungefär $1{,}7 \times 10^{308}$ 15 siffrors noggrannhet

Numeriska operatorer

`+ - *` addition, subtraktion och multiplikation

`/` division (heltalsdivision om båda operanderna är heltal, vanlig division annars)

`%` ger resten vid heltalsdivision

Operatorerna `* / %` har högre prioritet än `+ -`

```
i + j          x + 12.6
1 - i              x - y
x * y          i * maxStorlek
i / 10             x / y
```

```
int i = 14;
int j = 5;

i/j   // heltalsdivision, ger värdet 2

i%j   // rest vid heltalsdivision, ger värdet 4
```

Operatorerna ++ och --

`++`	ökar operandens värde med `1`. Värdet av `k++` är `k`:s gamla värde. Värdet av `++k` är `k`:s nya värde.
`--`	minskar operandens värde med `1`. Värdet av `k--` är `k`:s gamla värde. Värdet av `--k` är `k`:s nya värde.

```
int i, k, m, n;
i = 4; k = 7;
m = i++ * k--;    // postfix.
                  // m får värdet 4*7
i = 4; k = 7;
n = ++i * --k;     // prefix.
                   // n får värdet 5*6
```

Typen **`boolean`**

Möjliga värden: **`true`** och **`false`**

Jämförelseoperatorer					
==	lika med	<	mindre än	>	större än
!=	icke lika med	<=	mindre än el. lika med	>=	större än el. lika med

```
i > j              lägstaPoäng == 10
x <= 5.75          i != 0
```

Logiska operatorer

`A && B`	`A || B`	`! A`
sant om *både* `A` och `B` är sanna	sant om *minst en av* `A` eller `B` är sann	sant om `A` är falsk

`&&` och `||` har *lägre* prioritet än jämförelseoperatorer

`!` har *högre* prioritet än aritmetiska operatorer och jämförelseoperatorer

```
temp>20 && temp<30
i==1 || i==3 || i==5 || i==7 || i==9
!(temp>20 && temp<30)
```

Typen **char**

Möjliga värden: enskilda tecken

```
char c1, c2;
c1 = 'A';
c2 = 'é';
```

Escape-sekvenser:

```
'\''
'\\'
'\t'  (tabulator)
'\n'  (ny rad)
'\0'     '\33'   '\177'   '\266'   '\377'  (oktalt)
'\u2663'   '\u03A8'  (unicode, hexadecimalt)
```

Jämförelser:

```
if (c1 == '%')
 System.out.println("Procent");
if (c1 >= '0' && c1 <= '9')
 System.out.println("Siffra");
```