

# IDAT1003 Programmering 1

Kontrollstrukturen valg

Mål for dagen:

Bli kjent med kontrollstrukturen valg

Kunne illustrere valg ved hjelp av aktivitetsdiagram

Kunne programmere valg ved hjelp av if-setning og flervalg ved hjelp av nøstet if-setning og ved bruk av switch

Kunne bruke logiske variabler og enkle sammensatte logiske uttrykk

Kunne lage strukturert og oversiktlig programkode

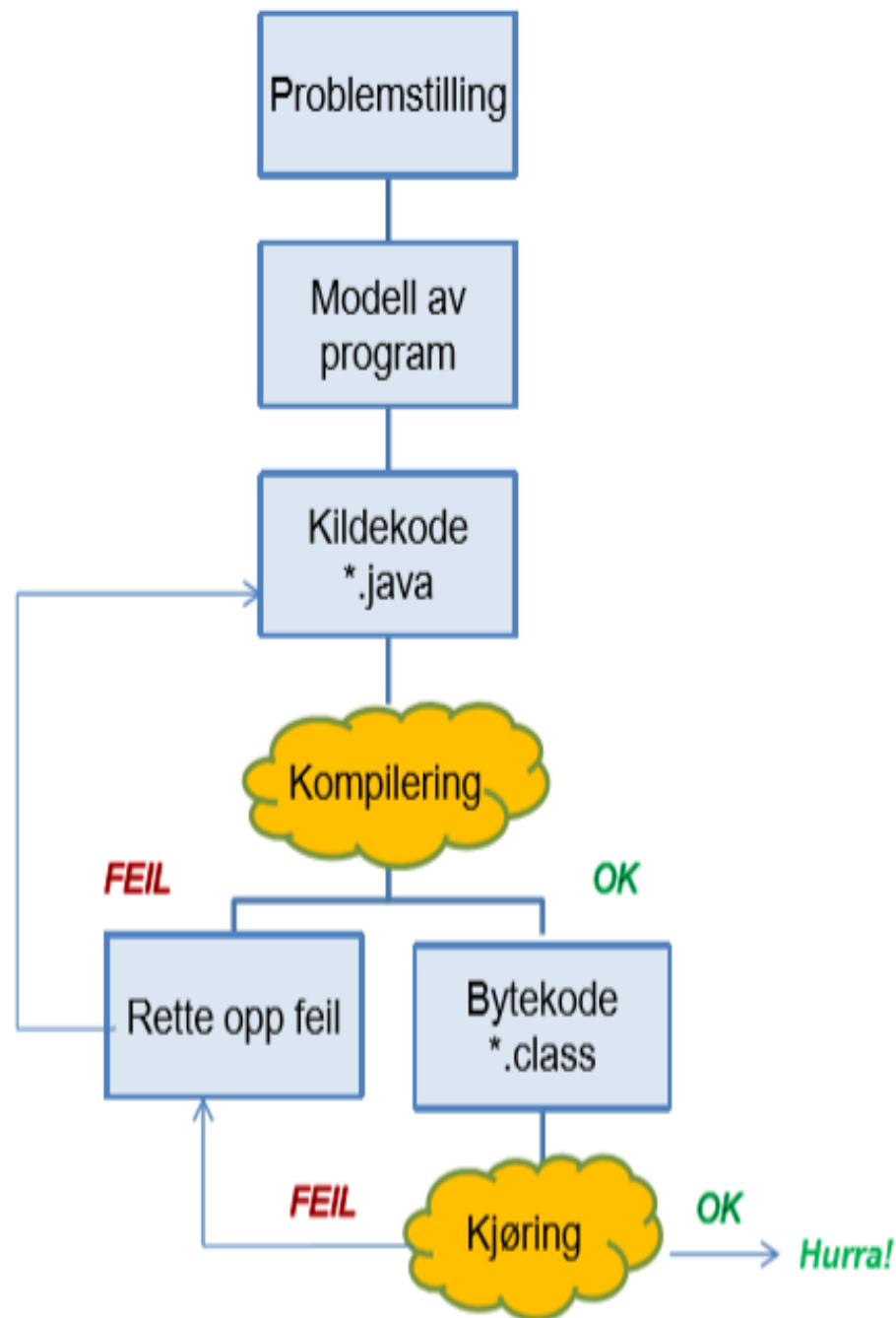
Muhammad Ali Norozi

# Agenda

- Repetisjon fra forrige uke
- Læringsutbytter kapittel 3.5 – 3.8
- Valg, aktivitetsdiagram og kalkulator eksempel
- Gruppeoppgave som løses på labben
- Flervalg, if og if-else
- Nøstet if-elseif-else, switch – flervalg
- Lysark: Beslutningoperator ?: og Logiske uttrykk
- Å sammenligne beregnede desimaltall
- Å sammenligning tekststrenger
- **Gjennomfør flervalgstest**

# Repetisjon

- Sette opp programmiljø
- Utskrift
  - `System.out.println();`
  - `JOptionPane`
- Variabler og konstanter
- Syntaksfeil
- Semantiske feil
- Kommentarer



# **MENTIMETER.COM** **(TILBAKEMELDING)**

En liten undersøkelse

# Læringsutbytter, forelesning 2

- Kunne identifisere **kontrollstrukturen valg** og illustrere den ved hjelp av **aktivitetsdiagram**
- Programmere valg ved hjelp av **if-setning** og flervalg ved hjelp av **nøstet if-setning** og ved bruk av **switch**
- Bruke betingelsesoperatoren **? :**
- Bruke **logiske variabler** og enkle **sammensatte logiske uttrykk**
- Lage **strukturert og oversiktlig programkode** ved bruk av innrykk (indentation) og navnekonvensjon
- Kunne gjøre **nøyaktige beregninger ved regning med desimaltall** slik at avrundingsfeil unngås.

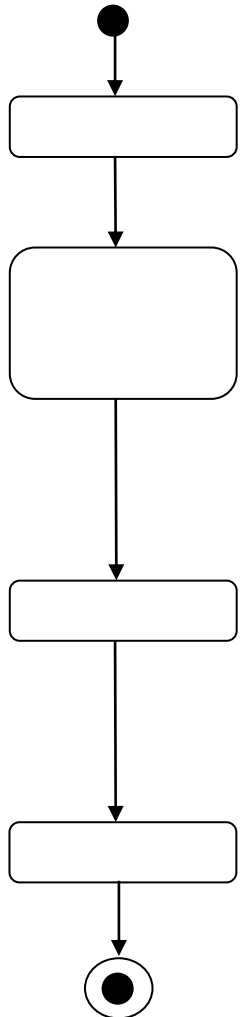
# Kontrollstrukturer

bestemt rekkefølge

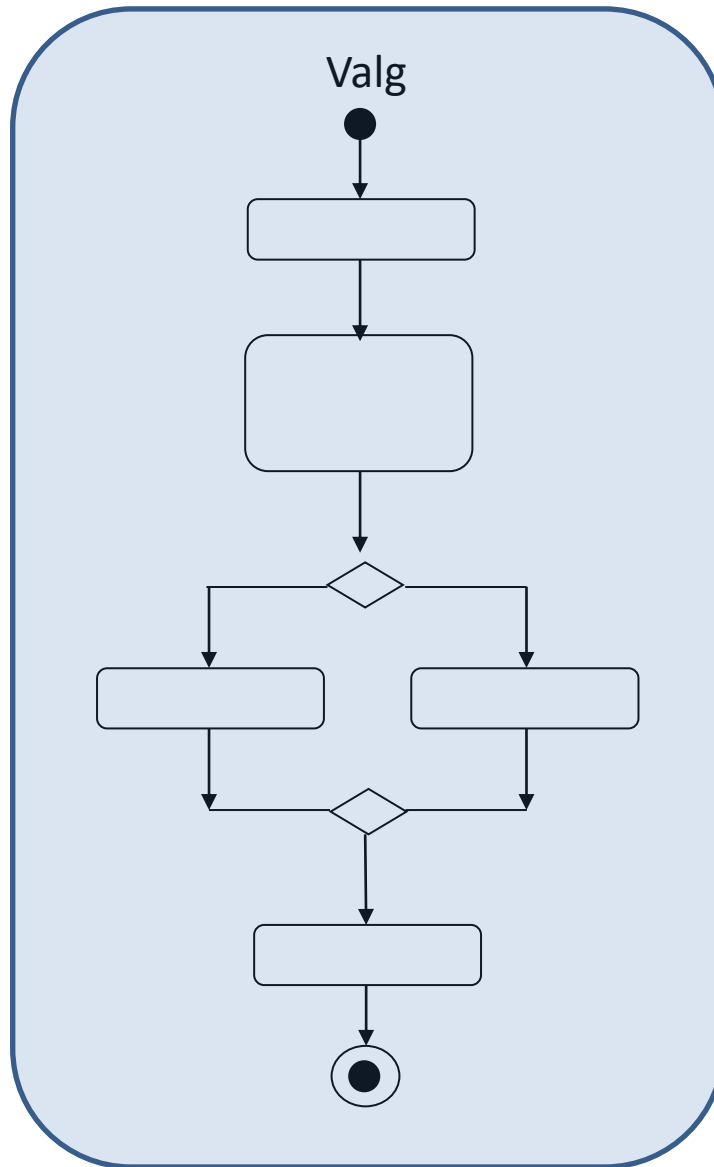
- En algoritme er et begrenset og ordnet sett med veldefinerte regler for løsning av et problem.
- Tre kategorier rekkefølge eller kontrollstrukturer:
  - sekvens (alle klientprogram hittil)
  - valg (if / switch)
  - løkke (while / do while / for se kapittel 4)
- Aktivitetsdiagram illustrerer kontrollstrukturer.

# UML – AKTIVITETSDIAGRAM (flowchart)

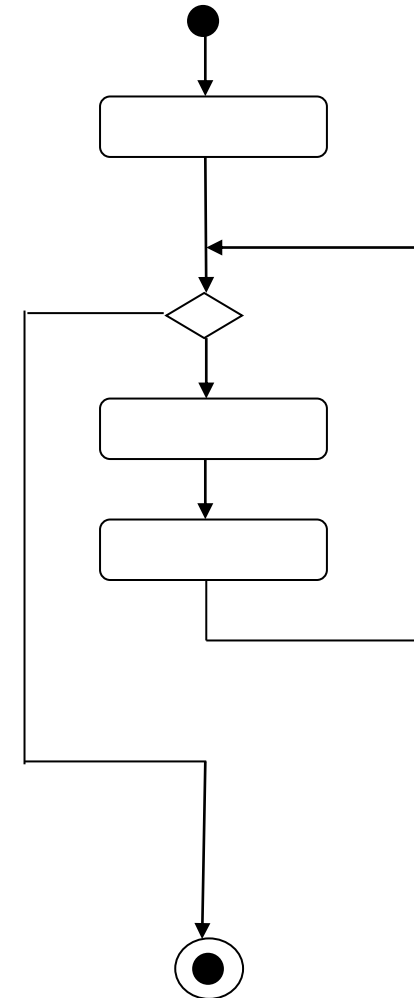
Sekvens



Valg

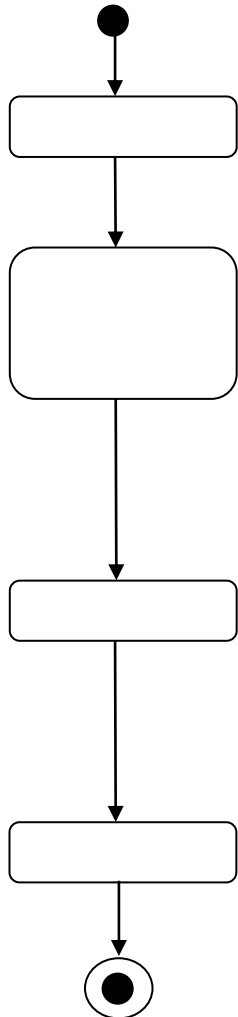


Løkke

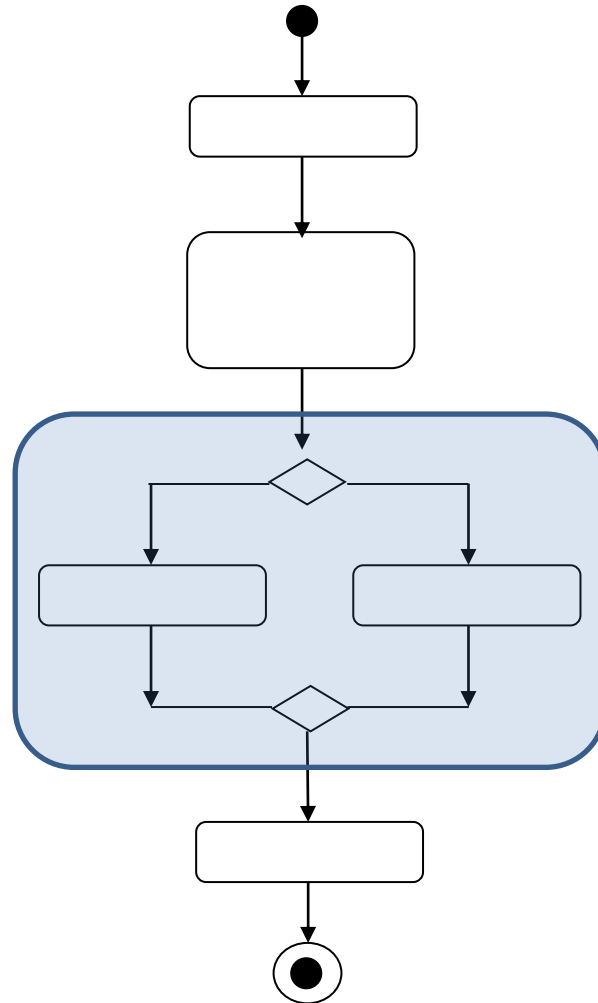


# UML – AKTIVITETSDIAGRAM (flowchart)

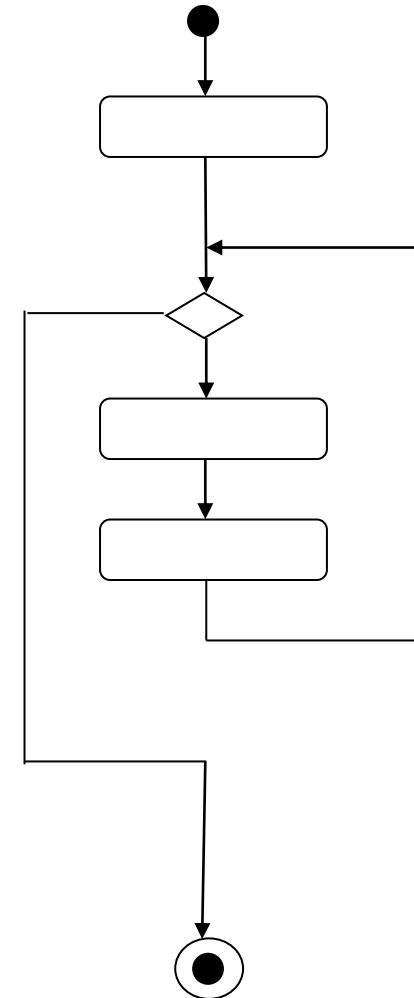
Sekvens



Valg

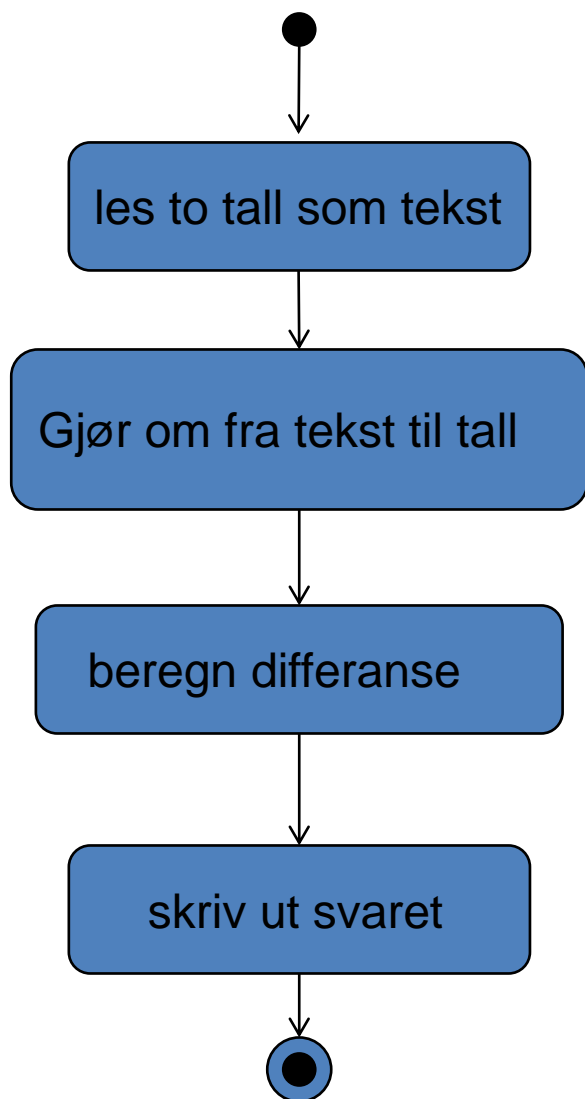


Løkke



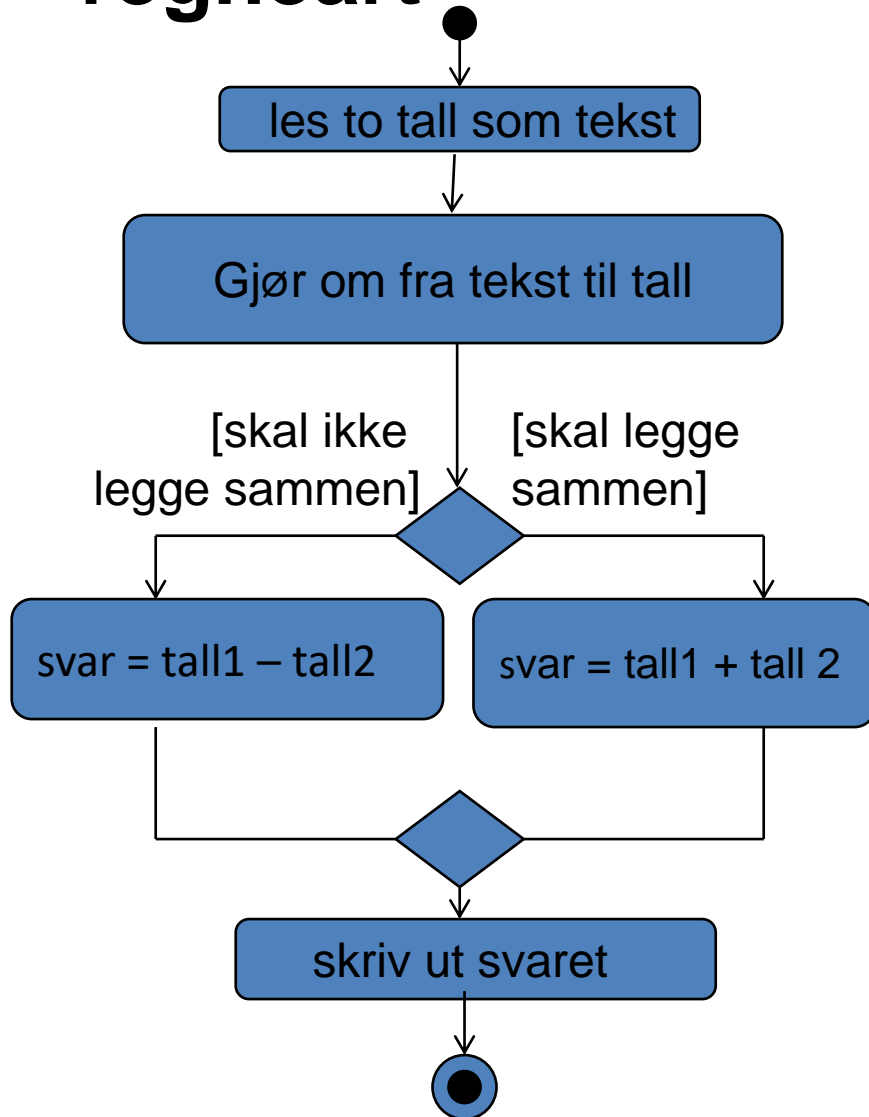


# Enkelt klientprogram som beregner differanse



```
5  import static javax.swing.JOptionPane.*;
6  class Calculator1{
    Run | Debug
7      public static void main (String[] args){
8
9          String readNumber1 = showInputDialog("Number1: ");
10         String readNumber2 = showInputDialog("Number2: ");
11
12         int number1 = Integer.parseInt(readNumber1);
13         int number2 = Integer.parseInt(readNumber2);
14
15         int answer = number1 - number2;
16
17         showMessageDialog(null, "Answer: " + answer);
18     }
19 }
```

# Klientprogram som lar bruker velge regneart

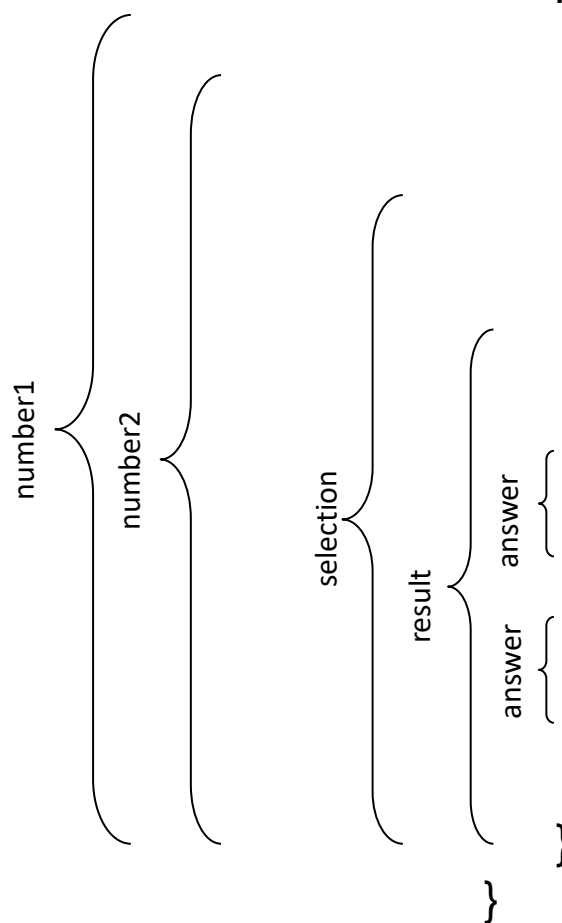


```
5 import static javax.swing.JOptionPane.*;
6 class Calculator2{
    Run | Debug
7     public static void main (String[] args){
8
9         String readNumber1 = showInputDialog("Number1: ");
10        String readNumber2 = showInputDialog("Number2: ");
11
12        int number1 = Integer.parseInt(readNumber1);
13        int number2 = Integer.parseInt(readNumber2);
14
15        int choice = showConfirmDialog(null, "Plus?", "Calculator", YES_OPTION);
16
17        int answer = 0;
18
19        if (choice == YES_OPTION) {
20            answer = number1 + number2;
21        } else{
22            answer = number1 - number2;
23        }
24
25        showMessageDialog(null, "Answer: " + answer);
26    }
27 }
```

# Blokker inne i metoder

- *Lokale variabler* er variabler deklartert inne i metoder.
- Definisjonsområdet (Skopet) til en lokal variabel er resten av den blokken der variabelen er deklartert.
  - { = start blokk
  - } = slutt blokk
- En metode kan inneholde mange blokker inne i hverandre.
- Vi kan deklarere variabler inne i indre blokker.
- Generelt bør vi deklarere lokale variabler i nærheten av der vi bruker dem.
- Neste lysark: alternativt kalkulator-program.  
Definisjonsområdet til variablene er vist.

# Blokker inne i metoder, def.omr.



```
class TestCalculatorBlocks {  
    public static void main(String[] args) {  
        double number1 = 12;  
        double number2 = 5;  
  
        int selection = showConfirmDialog(null, "Legge sammen? ",  
            «Calculator", YES_NO_OPTION);  
        String result = "\nThe answer is ";  
        if (selection == YES_OPTION) {  
            double answer = number1 + number2;  
            result += answer;  
        } else {  
            double answer = number1 - number2;  
            result += answer;  
        }  
        showMessageDialog(null, result);  
    }  
}
```

The diagram illustrates the following code blocks:

- number1**: A block containing the line `double number1 = 12;`
- number2**: A block containing the line `double number2 = 5;`
- selection**: A block containing the line `int selection = showConfirmDialog(null, "Legge sammen? ", «Calculator", YES_NO_OPTION);`
- result**: A block containing the line `String result = "\nThe answer is ";`
- answer**: Two blocks, one for the `if` branch and one for the `else` branch, each containing a calculation of `answer` and its addition to `result`.

# Gruppeaktivitet

- *Gjør oppgave 3. Gjennomgang i plenum*

*– Kode på neste foil*

- a) Hvor mange variabler deklarereres?
- b) Nummerer linjene. Hva er skopet til de ulike variablene
- c) Hva skrives ut hvis både  $a > 10$  og  $b < 20$  er sann.
- d) Hva skrives ut hvis  $a > 10$  er sann og  $b < 20$  er usann

```

class Oppgave3{
    public static void main(String[] args){

        int a = 11; // tatt med for testformål - SKAL IKKE TELLES MED I ANT VAR
        int b = 19; // tatt med for testformål- SKAL IKKE TELLES MED I ANT VAR
        if(a>10){ // start blokk 1
            int tall1 = 60;
            int tall2 = 50;
            System.out.println("1. tall1= " + tall1 + ", tall2= " + tall2);

            if(b<20){ // start blokk 2
                int tall3 = 20;
                tall1 = 30;
                tall2 = 100;
                int tall4 = tall1 + tall2 + tall3;
                System.out.println("2. tall1= " + tall1 + ", tall2= " + tall2);
                System.out.println("3. tall3= " + tall3 + ", tall4= " + tall4);
            }else { // slutt blokk 2, start blokk 3
                int tall3 = 65;
                System.out.println("4. tall3= " + tall3);
            } // slutt blokk 3
            System.out.println("5. tall1= " + tall1 + ", tall2= " + tall2);
        } // slutt blokk 1
    } // main
} // class

```

# Løsning oppgave 3

a) Hvor mange variabler deklarerer 5:

- I blokk 1: tall1 og tall2
- I blokk 2: tall3 og tall4
- I blokk 3: tall3

b) Hva er skopet til de ulike variablene?

- tall1 er linje 2-17. (blokk 1)
- tall2 er linje 3-17. (blokk 1)
- tall3 er linje 6-12. (blokk 2)
- tall4 er linje 9-12. (blokk 2)
- tall3 er linje 13-15. (blokk 3)

# Løsning oppgave 3

c) Hva skrives ut hvis  $a > 10$  og  $b < 20$  begge er sann?

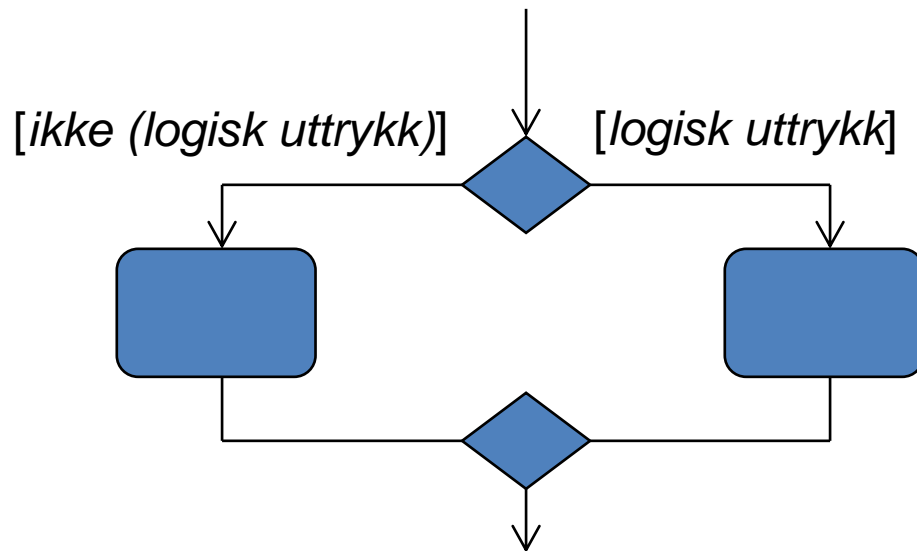
- $\text{tall1} = 60, \text{tall2} = 50$
- $\text{tall1} = 30, \text{tall2} = 100$
- $\text{tall3} = 20, \text{tall4} = 150$
- $\text{tall1} = 30, \text{tall2} = 100$

d) Hva skrives ut hvis  $a > 10$  er sann og  $b < 20$  er usann?

- $\text{tall1} = 60, \text{tall2} = 50$
- $\text{tall3} = 65$
- $\text{tall1} = 60, \text{tall2} = 50$



# if-setningen har to former



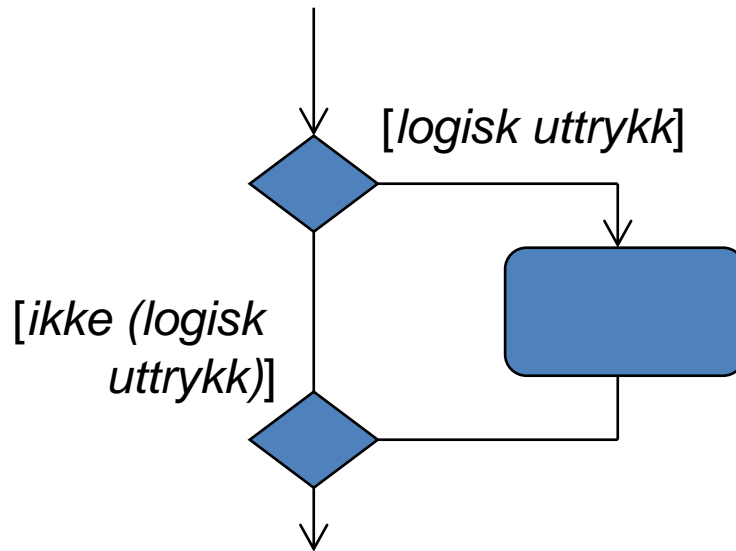
```
if (logisk uttrykk) {  
    setning1  
} else {  
    setning2  
}
```

OBS! Likhet skrives med to likhetstegn.

Et logisk uttrykk kan være en logisk variabel eller en sammenligning, eks:

(a > 10)  
(antJenter == antGutter)  
(ok)

# if-setningen har to former



```
if (logisk uttrykk){  
    setning1  
}
```

# Klammeparenteser og innrykk–VIKTIG!

```
if (logisk uttrykk)
    setning1
else
    setning2
```

- Hvis if/else består av mer enn én enkelt setning, **må** setningene omslutes av klammeparenteser: {--}.
- Dersom if/else kun består av en enkelt setning, er det syntaksmessig lov å droppe klammeparentesene. ***Av sikkerhetsgrunner bør en bruke klammeparenteser uansett.***

# Klammeparenteser og innrykk–VIKTIG!

- Eksempel, vi har oppdaget at vi også skal øke b med 1 dersom  $a > b$ :

Utgangspunkt:

```
if (a > b)
    sum += a;
```

Feilretting ??:

```
if (a > b)
    sum += a;
    b++;
```

- Tror du sjansen for å skrive riktig er større dersom utgangspunktet er:

- `if (a > b) sum += a;` eller

```
if (a > b) {
    sum += a;
}
```

```
if (a > b) {
    sum += a;
    b++;
}
```

# Oppgave 4 i plenum

*Skriv kode i henhold til anbefalinger for innrykk og {}.*

*Finn verdiene til alle variablene etter at kodebiten er utført*

```
int a = 20;  
int b = 30;  
int p = 20;  
int q = 40;  
int r = 30;  
int s = 15;  
if(a<b) a = b; b = 10;  
if(p==20) q = 13; else q = 17;  
if(r>s){ q = 100;  
} s = 200;
```

# Løsning oppgave 4

```
int a = 20; int b = 30; int p = 20; int q = 40;  
int r = 30; int s = 15;
```

```
if (a < b) {  
    a = b;  
}  
b = 10;  
if (p == 20) {  
    q = 13;  
} else {  
    q = 17;  
}  
if (r > s) {  
    q = 100;  
}  
s = 200;
```

Variablene har følgende verdier etter at kodebiten er kjørt:

a = 30

b = 10

p = 20

q = 100

r = 30

s = 200

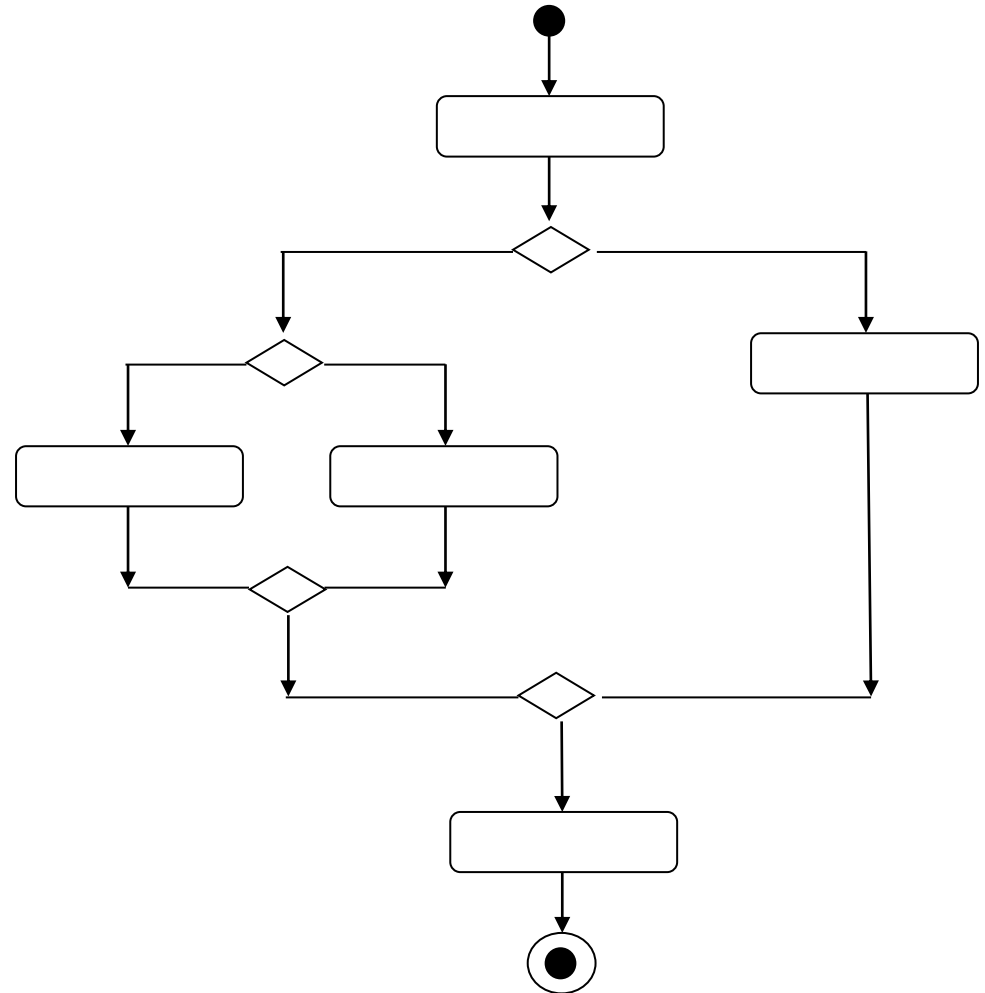
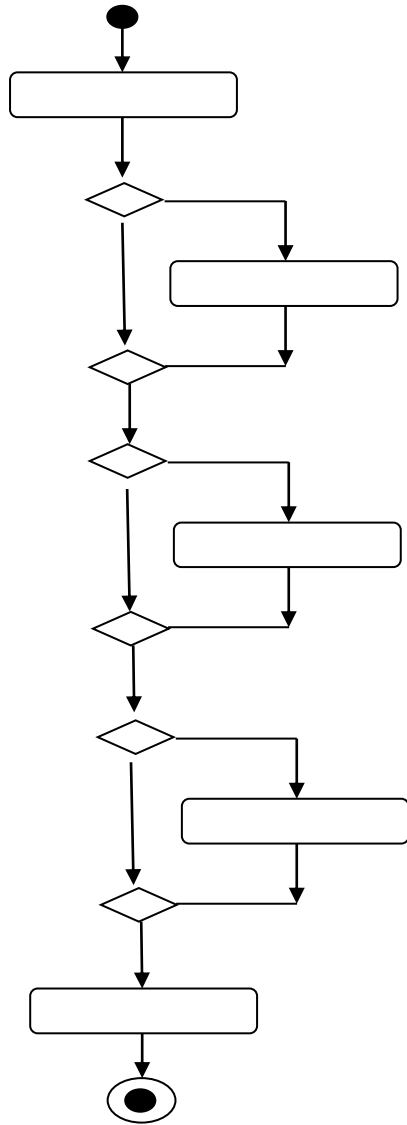
# Nøstet if og flervalgsetninger

```
if (temperature > 0) {  
    System.out.println("Hot");  
}  
if (temperature == 0) {  
    System.out.println("Zero degrees");  
}  
if (temperature < 0) {  
    System.out.println("Cold");  
}
```

```
if (temperature > 0) {  
    System.out.println("Hot");  
}  
else {  
    if (temperature == 0) {  
        System.out.println("Zero degrees");  
    }  
    else {  
        System.out.println("Cold");  
    } // slutt på innerste if-else-setning  
} // slutt på ytterste if-else-setning
```

## Tegn aktivtetsdiagram for alle tre

```
if (temperature > 0) { System.out.println("Hot"); }  
else if (temperature == 0) { System.out.println("Zero degrees"); }  
else { System.out.println("Cold"); }
```





# Hva er forskjellen mellom følgende to kodebiter?

```
int a = -10;  
int b = 20;  
if (a > 0) {  
    if (b > 10) b = 10;  
}  
else a = 0;  
System.out.println(a + " " + b);
```

Gir utskrift: 0 20

```
int a = -10;  
int b = 20;  
if (a > 0)  
    if (b > 10) b = 10;  
else a = 0;  
System.out.println(a + " " + b);
```

Gir utskrift: -10 20

Dersom du bruker flere if-setninger inne i hverandre (nøstet if), og en else-blokk ikke tilhører den nærmeste if'en foran, må dette markeres med {}.

# Beslutningstabeller

- En karakter på en prøve beregnes etter følgende *beslutningstabell*:

poeng	karakter
96–100	A
86–95	B
71–85	C
55–70	D
36–54	E
0–35	F

- Oppgave: *Lag et program som lar brukeren skrive inn poengsum og får ut karakteren. Algoritmen ser slik ut:*
  - Les inn poeng
  - Bestem karakter ut fra beslutningstabell
  - Skriv ut karakteren
- *Utvikle Karakter.java*

## switch-setningen

- switch-setningen kan brukes dersom hvert tilfelle i en flervalg-setning svarer til en bestemt *heltallsverdi* eller et *tegn*.
- Eksempel til høyre.
- Hver enkelt case-etikett må representere forskjellige konstantuttrykk, kun ett uttrykk for hver etikett. Kan ikke oppgi intervaller eller flere enkeltverdier.
- Bare uthopp dersom break påtreffes.

... les inn plassering fra brukeren ...

```
switch (plassering) {  
    case 1:  
        System.out.println(«Gold!»);  
        break;  
    case 2:  
        System.out.println(«Silver!»);  
        break;  
    case 3:  
        System.out.println(«Bronz!»);  
        break;  
    case 4:  
        /* ikke break */  
    case 5:  
        /* ikke break */  
    case 6:  
        System.out.println("Points!");  
        break;  
    default:  
        int happynumber = (int) (100 * Math.random() + 1);  
        System.out.println("Thank you for the effort!");  
        System.out.println("You get a happynumber: " +  
            lykketall);  
        break;  
}
```

# Gruppeoppgave

Påpek feil og svakheter ved følgende switch-setning:

```
switch (weekday) {  
case 1,2:  
    System.out.println(«The beginning of the week»);  
case 3,4:  
    System.out.println("Middle of the week");  
case 5:  
    System.out.println("The end of the week");  
case 6,7:  
    System.out.println("Weekend");  
};
```

# Betingelsesoperatoren ?:

- Den eneste operatoren som tar tre operander
- Eksempel:

```
max = (number2 > number1) ? number2 : number1;
```

- Er logisk det samme som:

```
if (number2 > number1) {  
    max = number2;  
} else {  
    max = number1;  
}
```

# Logiske uttrykk

- Et logisk uttrykk har verdien sann (true) eller usann (false).
- IKKE-operatoren ! snur verdien til uttrykket.
  - (!true) er lik false
  - (!false) er lik true
- Merk skrivemåten:
  - if (found)        // slik, men ikke slik: if (found == true)....
  - if (!found)       // slik, men ikke slik: if (found == false)...

# Logiske uttrykk

- Enkle logiske uttrykk lages ved å bruke *sammenligningsoperatorene*:

< mindre enn	(numberOfKilo < 10)
<= mindre enn eller lik	(numberOfKilo <= 10)
> større enn	..
>= større enn eller lik	..
!= ikke lik	(price * number) != 100
== lik	(number1+number2==number3+number4)
instanceof	(calculus instanceof Calculator)

# Logiske uttrykk

- Sammensatte logiske uttrykk lages ved å sette sammen enkle logiske uttrykk ved hjelp av *operatorene*:

&&      og      (number > 30 && number < 50)

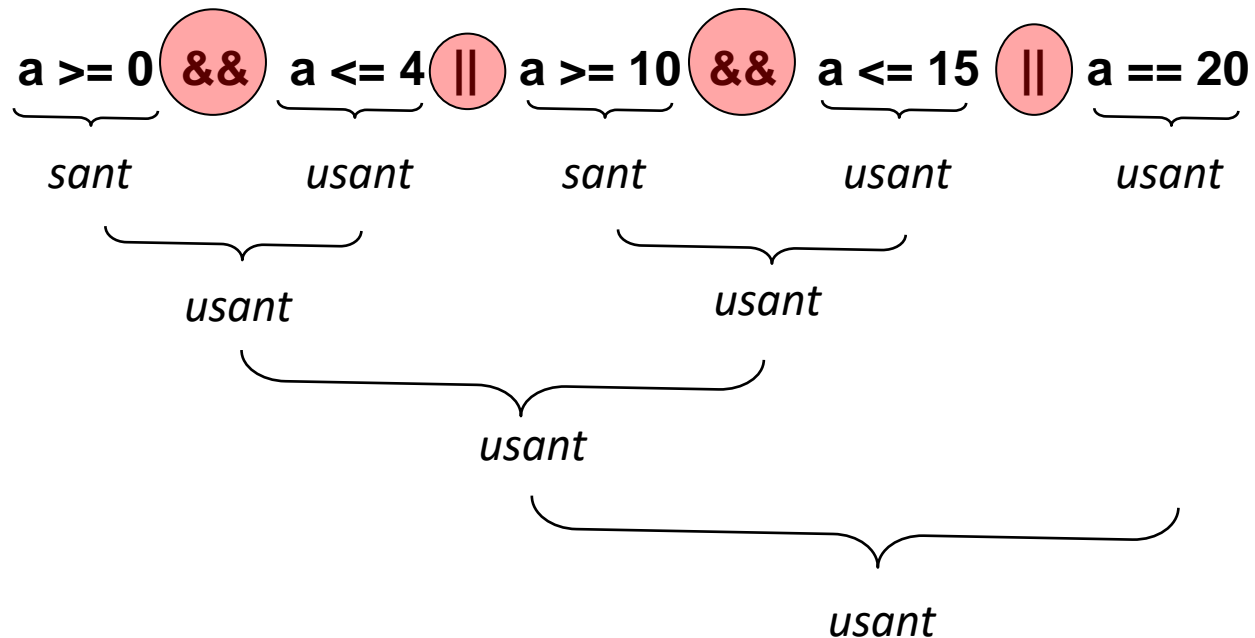
||      eller      (number <= 30 || number >= 50)



# Operatorer - prioritet

Operator	Navn/beskrivelse	Assosiativitet
* / %	Multiplikasjon Divisjon Modulus	Venstre
+ -	Addisjon Subtraksjon	Venstre
< <= > >=	Mindre enn Mindre enn lik Større enn Større enn lik	Venstre
== !=	Lik Ikke lik	Venstre
&&	Og	Venstre
	Eller	Venstre
=	Tilordning	Høyre

# Beregning av et komplisert logisk uttrykk



a er lik 16

# Avbrutt beregning

- Logiske uttrykk som inneholder && eller || beregnes ved å bruke teknikken "avbrutt beregning".
- Beregningen avsluttes så fort resultatet er bestemt.
- Operatoren && ("og") mellom uttrykkene:
  - Fortsett beregningen så lenge uttrykkene er sanne.
  - Beregningen avbrytes dersom et ikke-sant uttrykk påtreffes.

# Avbrutt beregning

- Operatoren `||` ("eller") mellom uttrykkene:
  - Fortsett beregningen så lenge uttrykkene er usanne.
  - Beregningen avbrytes dersom et sant uttrykk påtreffes.
- Eksempel:
  - `if (number >= 0 && Math.sqrt(number) < limit) ....`
- Rekkefølgen på sammenligningene kan ikke byttes om. Da risikerer vi å forsøke å beregne kvadratroten til et negativt tall

# Gruppeoppgave

## Oppgave 1

- a)  $5 * 3 / 4 + 3 < 10$
- b)  $!(4 > 3)$
- c)  $2 < 20 \ \&\& \ 4,5 > 20$
- d)  $2 < 20 \ \parallel \ 4,5 > 20$

## Oppgave 3

Hva blir den logiske verdien til uttrykket

```
a == - 20 || a >= 0 && a <= 10 || a >= 15 && a <= 20 || a > 100
```

Når a har verdiene 17, 120, -30

## Oppgave 2 Sett opp logiske uttrykk

- a) Antall elever skal være mellom 20 og 30
- b) Gevinster tilfaller lodd nr 3, 18 og 25
- c) Svaret på spørsmålet skal være stor eller liten 'J'
- d) Temperaturen skal være utenfor det lukkede intervallet [15, 25]
- e) Summen skal være positiv og utenfor det åpne intervallet  $<10, 100>$
- f) Tegnet må være en bokstav (A-Å) stor eller liten Tegnet skal være et siffer (0-9)

# Løsning oppgave 1

a)  $5 * 3 / 4 + 3 < 10$

$$15/4 + 3 < 10$$

$$3 + 3 < 10 \Rightarrow \text{TRUE}$$

b)  $!(4 > 3) \Rightarrow !(\text{TRUE}) \Rightarrow \text{FALSE}$

c)  $2 < 20 \ \&\& \ 4,5 > 20 \Rightarrow \text{FALSE}$

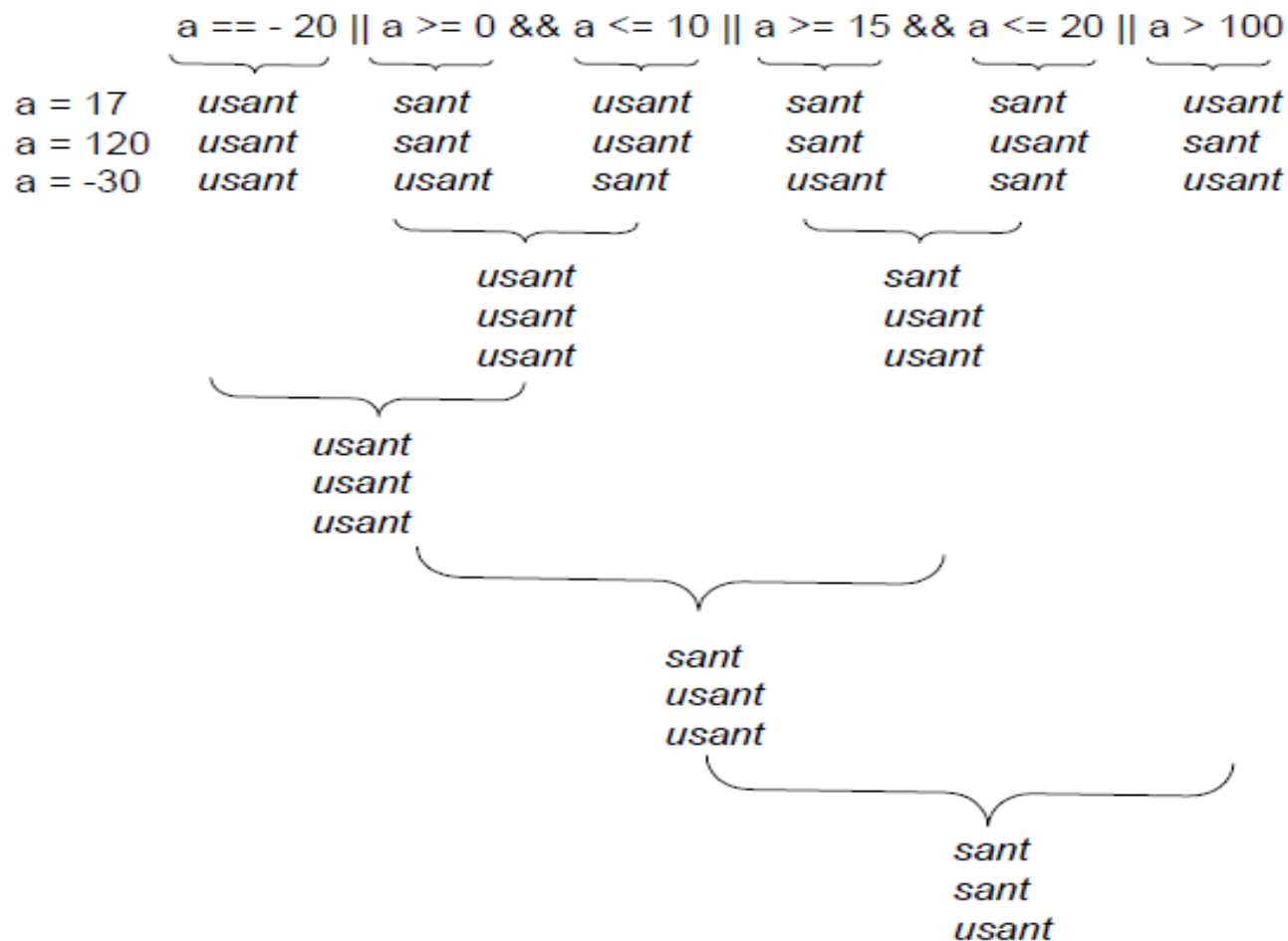
d)  $2 < 20 \ \parallel \ 4,5 > 20 \Rightarrow \text{TRUE}$

# Løsning oppgave 2

- a) `numberOfStudents > 20 && numberOfStudents < 30;`
- b) `ticketNr == 3 || ticketNr == 18 || ticketNr == 25;`
- c) `answer == 'y' || answer == 'Y';`
- d) `temp < 15 || temp > 25;`
- e) `sum > 0 && sum <= 10 || sum >= 100;`
- f) `letter >= 'A' && letter <= 'Z' || letter >= 'a' && letter <= 'z'`  
`|| letter == 'æ' || letter == 'ø' || letter == 'å' || letter == 'Æ'`  
`|| letter == 'Ø' || letter == 'Å';`
- g) `letter >= '0' && letter <= '9';`

# Oppgave 3

$a = 17$  og  $a = 120$  gir at verdien til uttrykket er sant, mens  $a = -30$  gir verdien usann.





# Å sammenligne beregnede desimaltall

- Hva blir verdien av tall3 etter at følgende kodebit er utført:
  - `double number1 = 1.0e20;`
  - `double number2 = number1 + 1.0;`
  - `double number3 = number2 - number1;`      Svar: number3 blir 0.
- Hvorfor?
  - $1 + 1,0 \cdot 10^{20} = 1,0000000000000000000001 \cdot 10^{20}$ .
  - Datatypen double klarer bare ca. 15 signifikante sifre.
  - Mister derfor verdien 1 som vi prøver å addere til det kjempestore tallet.

# Å sammenligne beregnede desimaltall

- Bruk aldri operatorene == og != for å sammenligne resultater fra beregninger med desimaltall.
- Kontroller heller at forskjellen mellom tallene er mindre enn en gitt toleranse:

```
final double toleranse = 0.00001; // størrelsen i forhold til tallene som  
sammenlignes  
if (Math.abs(tall1 - tall2) < toleranse) System.out.println("The number  
are equal enough");  
else System.out.println("The numbers are different.");
```

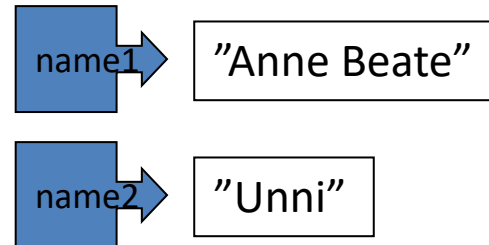
**husk å kontrollere mot  
absoluttverdien!**

# Sammenligning av strenger

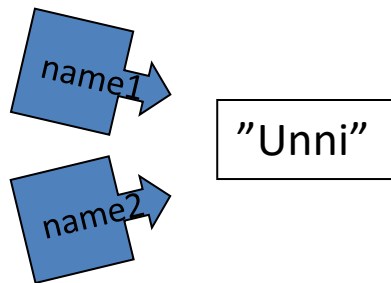
- To strenger er gitt:

```
String name1 = "Anne Beate";
```

```
String name2 = "Unni";
```



- Uttrykket `(name1 == name2)` ... sammenligner referansene.
- Likhet dersom referansene er like, det vil si dersom de peker til det samme objektet.



- Må bruke metoder fra klassen `String` for å sammenligne innholdet i objektene.

# Sammenligningsmetoder i klassen String

- `public boolean equals(Object theOtherObject)`  
`public boolean equalsIgnoreCase(String theOtherObject)`
  - Returner true eller false
- `public int compareTo(String theOtherObject)`  
`public int compareToIgnoreCase(String theOtherObject)`
  - Sammenligning skjer i henhold til Unicode-rekkefølgen.
  - `compareTo()`-metodene returnerer negativ verdi dersom strengen meldingen sendes til, ligger *foran* strengen som sendes inn som argument til metoden, positiv dersom den ligger *etter*, og 0 dersom de er like.

# Sammenligning av strenger, Eks.

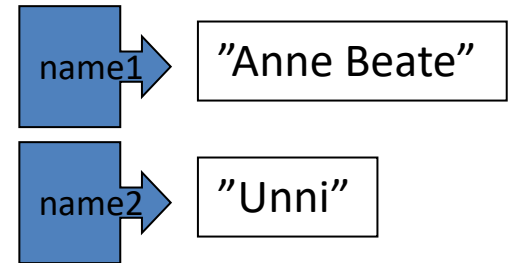
- Vi bruker metoden equals():

```
if (name1.equals(name2)){  
    System.out.println(«The names are the same.»);  
}else{  
    System.out.println(«The names are different.»);  
}
```

  - Utskrift: *The names are different.*
- Vi bruker metoden compareTo():

```
int result = name1.compareTo(name2);  
if (result < 0){  
    System.out.println(name1 + " comes first.");  
} else if (result > 0){  
    System.out.println(name2 + " comes first.");  
} else{  
    System.out.println(«The names are the same.»);  
}
```

  - Utskrift: *Anne Beate comes first.*



# Råd og tips om strenger

- Dersom tekstene som skal sammenlignes leses inn fra brukeren, bør de “trimmes” før sammenligning, det vil si at blanke i begynnelsen og slutten av teksten fjernes:  

```
String name = JOptionPane.showInputDialog("Name: ");  
name = name.trim();
```
- Vær obs på kompilatorens “intelligens”. Studér følgende kodebit:  

```
String name1 = "Ole Petter";  
String name2 = "Ole Petter"; // vi lager et objekt til, eller?  
if (name1 == name2) System.out.println("Similar"); // prøver sammenligningsoperatoren  
else System.out.println(">Not similar");
```
- Utskriften blir “Similar”. Hvorfor?
- Årsaken er at Java-kompilatoren observerer at de to tekstene er like og derfor kun lager ett objekt i første omgang.