# **Deklarasjon av variabler og Klasser**

* Klasse

En klasse skal alltid ha stor bokstav i begynnelsen av navnet.

* + - class Reise { … }
* Variabler

Kan her legges til private, eller public foran for å bestemme hvem som kan se den.

Merk her at variabelnavn alltid skal begynne på liten bokstav.

* + - int reise; int reise = 4;
    - String reise; String reise = “Reisen”;
    - int[] reise; int[] reise = {0, 2, 53, 8};
    - boolean reise; boolean reise = true;

# **Konstruktører**

For å lage en konstruktør, må en bare skrive navnet på klassen, fulgt av parenteser.

Eksempel under for klassen "Reise", hvor det er en variabel kalt "reisen" av typen int.

Reise(int antallStoppesteder) {

reisen = antallstoppesteder;

}

for dyp kopiering av en array, brukes en løkke som vist under.

Reise(int[] stoppestedene) {

int antStoppested = stoppestedene.length;

int[] kopi = new int[antStoppested];

for (int i = 0; i < antStoppested; i ++) {

kopi[i] = stoppestedene[i];

}

reisen = kopi;

}

**Eksempel på bruk av konstruktør**

Reise reisen = new Reise(4);

# **Funksjoner**

Når funksjoner deklareres må en oppgi hva som skal returneres. I en funksjon kan en hvilken som helst datatype returneres (boolean, String osv. se liste for flere), det kan også sies at ingenting skal returneres (void)

En funksjon skal alltid starte med liten bokstav.

Når er funksjon deklareres, må det oppgis om denne er public eller private.

public void skrivTilBruker(String tekst) { … }

public boolean erLike(int Tall1, int Tall2) { … return svar}

private String settSammen(String tekst1, String tekst2) { … return sammenslått}

Når et program skal kjøres vil en trenge en funksjon som kan kalles automatisk, og sa brukes den følgende:

public static void main(String[] args) { … }

# **Datatyper**

* Numeriske
  + byte (heltall, 8 bit eller -128 til 127)
  + short (heltall, 16 bit eller - 32 768 til 32 767)
  + int (heltall, 32 bit eller +- ca. 2 mrd.)
  + long (heltall, 64 bit eller veldig store og små tall)
  + float (desimaltall, 32 bit)
  + double (desimaltall, 64 bit)
* Tekst
  + String (tekst)
  + char (ett unicode tegn)
* Logisk
  + boolean (false/true)

# 

# 

# **Konvertering eksempler**

* String til tall, samme metode skal funke for alle talltyper

String tallet = "15"

int tall1 = Integer.parseInt(tallet);

float tall2 = Float.parseFloat(tallet);

* tall til String

Gjøres automatisk om du har en string å slå den sammen med

String tekst = "tallet er " + tall1;

String tekst = "" + tall2;

# **Import av utvidelser**

* JOptionPane

Brukes til å vise popup bokser til bruker

imports static javax.swing.JOptionPane.\*

Denne vil gi tilgang til flere nyttige ting som nevnt under med eksempel.

* Message box

String beskjed = "Sikkert noe viktig";

showMessageDialog(null, beskjed);

* Input box

String navn = showInputDialog("Skriv inn navnet ditt");

* Option box

final String[] muligheter = {valg 1, valg 2, valg3};

int valg = showOptionDialog(null, "velg en", "velg", DEFAULT\_OPTION, PLAIN\_MESSAGE, null, muligheter, muligheter[0]);

# 

# 

# 

# **Sortering av egenproduserte klasser**

I java er det en ferdiglaget funksjon for å sortere klasser (slik som string ol.) når de er i en array eller liste. Denne funksjonen for arrays er: java.util.Arrays.sort()

**Numerisk sortering**

For å implementere denne i egne klasser, må det gjøres visse steg.

1. Det må i klasse hodet settes inn "implements Comparable<klassenavn>"

f.eks.

"class Kort implements Comparable<Kort> { … }

1. Det må legges inn en compareTo-funksjon i klassen som sammenligner seg selv og en gitt variabel av samme type, og som deretter returnerer en integer.

f.eks.

public int compareTo(Kort annetKort) {

return (verdi - annetKort.getVerdi());

}

Denne metoden vil bare kunne sortere om vi har numeriske verdier for de forskjellige gjenstandene skal sorteres.

**Alfabetisk sortering**

For å implementere denne i egne klasser, må det gjøres visse steg.

1. Det må i klasse hodet settes inn "implements Comparable<klassenavn>"

f.eks.

"class Matrett implements Comparable<Matrett> { … }

1. Deretter må vi lage et "norsk" collator-objekt i klassen som kan sammenligne to tekststrenger:

"private final static java.text.Collator kollator = java.text.Collator.getInstance(new java.util.Locale("no", "NO"));

1. Det må legges inn en compareTo-funksjon i klassen som sammenligner seg selv og en gitt variabel av samme type, og som deretter returnerer en integer.

f.eks.

public int compareTo(Matrett annenMatrett) {

return kollator.compare(navn, annenMatrett.getNavn);

}

Når disse stegene er gjennomført kan du bruke java.util.Arrays.sort() på klassene dine ved å sette det innenfor parentesene

# **Løkker**

* For-løkke

for (int i = 0; i < antallRunder; i ++) {

teller ++;

}

for (int i = 0; i < liste.length; i ++) {

linje = linje + liste[i];

teller ++;

}

* For each-løkke

for (String ord : listeOrd) {

linje = linje + ord;

}

* While-løkke

while (teller < 10) {

teller ++;

}

* Do while-løkke

Fordelen med denne over while-løkken er at denne vil gå gjennom løkka minst en gang uansett hva.

do {

teller ++;

} while (teller < 10);

# **Try/Catch**

Illustrerer dette med å prøve å konvertere string til tall, som er mest sannsynlige bruken.

Den første av catch-blokkene fanger feil i forbindelse med tallkonvertering, den andre vil fange alle andre eventuelle feil.

try {

int tall = Integer.parseInt(tallet);

}

catch(NumberFormatException nfe) {

int tall = 0;

}

catch(Exception e) {

System.Out.println("Noe annet gikk galt");

}

# **If og select case**

* If setninger

if (i < 10) {

...

} else if (i == 10) {

...

} else {

...

}

* select case

switch (i) {

case 1: tall = 1;

break;

case 2: tall = 2;

break;

case 3: tall = 3;

break;

default: tall = 0;

break;

}

# **Logiske operatører**

* Aritmetik
  + + (pluss)
  + - (minus)
  + \* (gange)
  + / (dele)
  + % (rest , 15 % 4 = 3 og 10 % 3 = 1 , teknisk sett modulo)
* likhet og relasjon
  + == (lik)
  + != (ulik)
  + > (større enn)
  + < (mindre enn)
  + >= (større eller lik)
  + <= (mindre eller lik)
* annet
  + && (og , tilsvarende "And" fra vb.net , True && False = False)
  + || (eller , tilsvarende "Or" fra vb.net , True && False = True)
  + ! (ikke , tilsvarende "Not" fra vb.net , ! True = False)
  + ++/ -- (øker/reduserer med en)

# **Test av tegn**

for å sjekke om et tegn er en bokstav (fra A til Z), kan en bruke følgende hvor vi går ut fra at tegn er en variabel av typen char.

boolean erBokstav = tegn >= 'A' && tegn <= 'Z';

Denne vil bare gi deg svaret true om du har en stor bokstav mellom A og Z, dermed ikke Æ, Ø eller Å. For små bokstaver kan bare den store A-en og Z-en byttes med små. For å sjekke Æ, Ø og Å med samme metode må disse skrives individuelt ettersom de ligger i feil rekkefølge.

Den følgende vil sjekke om noe er et siffer av typen char for deg.

boolean erSiffer = tegn >= '0' && tegn <= '9';

For å få ett tegn ut av en String, kan den følgende String-metoden brukes:

public String tekst = "teksten som testes";

char tegn = tekst.charAt(index);

# **Tips**

Variabler av typen char kan brukes i løkker i stedet for tall.

for (char i = 'A'; i <= 'Z'; i ++) {

System.Out.println(i)

}

eksempel equals funksjon:

public boolean equals(Object objekt) {

if (!(objekt instanceof Kort)) {

return false;

}

if (this == objekt) {

return true;

}

Kort annetKort = (Kort) objekt;

return verdi == annetKort.getVerdi();

}