Løse reelle problemer

Litt mer om løkker,
metoder med returverdier,
innlesing fra fil og strenger

INF1000, uke5
Ragnhild Kobro Runde

MER OM LØKKER

Repetisjon fra forrige uke: while

- Syntaks:

  while (condition) {do1; do2; ...}

- Eksempel:

```
int tall = 1;
while (tall < 100) {
    System.out.println(tall);
    tall = tall + 5;
}
```

Initialiser-sjekk-inkrementer

- Et vanlig mønster knyttet til løkker:

  – Gi startverdi til variabel før selve løkken (initialiser).

  – Gå i løkke inntil variabelen passerer en gitt verdi.

  – Inkrementer variabel i slutten av kodeblokken.

Initialiser-sjekk-inkrementer: for

- U5SumTallrekkeVhaFor.java

Initialiser-sjekk-inkrementer

While

```
int sum = 0;
int tall = 1; intialiser
while (tall<=100){ sjekk
  sum = sum + tall;
  tall = tall + 1; inkrementer
}
System.out.println(sum);
```

For

```
int sum = 0;
for (int tall=1; tall<=100; tall=tall+1){
  sum = sum + tall;
}
System.out.println(sum);
```

Initialiser-sjekk-inkrementer: for

- Litt mer kompleks syntaks:

  for (initialization; condition; update) {do1; do2; ...}

- For-uttrykket inneholder tre helt ulike ting:
  - Initialization: kjøres bare én gang, før selve løkken
  - Condition: sjekkes hver runde før kodeblokken kjøres (tilsvarer uttrykket inni while-testen)
  - Update (inkrementering): kjøres i slutten av løkken (legges til i slutten av kodeblokken)

Initialiser-sjekk-inkrementer: for

- for er egentlig ikke mer enn en forkortet skrivemåte
  - Alt man kan gjøre vha for kan man også gjøre vha en tilsvarende while.
  - Eneste formål er å forkorte og tydeliggjøre en typisk bruksmåte av (while-) løkker.
  - Den typiske bruken er noe som skal repeteres et fastlagt antall ganger.
  - I praksis brukes likevel for oftere enn while, fordi denne typiske bruksmåten er så veldig vanlig.

METODER MED RETURVERDIER

Metoder med returverdi

- Void-metodene fra uke2:

  static void minMetode(...) {...}

- Metode med returverdi:

  static DATATYPE minMetode(...) {...}

- Eksempler:

```
static int gangMedTo(int tall) {
    return tall*2;
}
static String lagVelkomst(String fag, String navn) {
     return "Velkommen til " + fag + " kjaere " + navn;
}
```

Bruke (kalle) metoder med returverdier:

- static int gangMedTo:

```
int dusin=13;
int toDusin = gangMedTo(dusin);
System.out.println(toDusin); //Skriver ut 26
```

- static String lagVelkomst

```
String velkomst = lagVelkomst("inf1000", "Eric");
System.out.println(velkomst);
//Skriver ut "Velkommen til inf1000 kjaere Eric"
```

Eksempel på forenkling av kode vha metode

- Forenkle summering av to tall:

{U5InputSum1.java-U5InputSum2.java}

Utsett til i morgen, det du ikke trenger gjøre i dag

- Metoder med returverdi tillater å utsette problemer!
  - Fokuser først på hva som trengs overordnet.
  - Deretter gå løs på detaljene.
- Eksempel:

```
int kvm=60; int postnr=0316; int lonn=500 000;
pris = regnBoligPris(kvm, postnr);
maksLaan = regnKredittVerdighet(lonn);
if (maksLaan > pris) {
    System.out.println("Yes!");
}
// Deretter skriv selve metodene..
```

Prøv selv (5 min)
(Oppgave hentet fra eksamen 2014)

Skriv en metode

static int pris (boolean gratis, int alder)

Dersom parameteren gratis har verdien true, skal metoden alltid returnere 0. Dersom parameteren gratis har verdien false og verdien av alder er mindre enn 18, skal metoden returnere 100, ellers 200. Altså skal f.eks. kallet pris (true, 10) returnere 0, kallet pris(false,10) returnere 100 og kallet pris(false, 50) returnere 200.

En mulig løsning

```
static int pris(boolean gratis, int alder) {
    int svar;
    if (gratis) {
        svar = 0;
    } else if (alder < 18) {
        svar = 100;
    } else {
        svar = 200;
    }
    return svar;
}
```

En annen mulig løsning (returnere inne i if)

```
static int pris(boolean gratis, int alder) {
    if (gratis) {
        return 0;
    }
    if (alder < 18) {
        return 100;
    }
    return 200;
}
```

En tredje mulig løsning (bare rene if - ingen else)

```
static int pris(boolean gratis, int alder) {
    int svar=0;
    if (gratis) {
        svar = 0;
    }
    if (!gratis && alder < 18) {
        svar = 100;
    }
    if (!gratis && alder >= 18) {
        svar = 200;
    }
    return svar;
}
```

INNLESING FRA FIL

Innlesing fra fil

- Å hente data fra filer er gøy!
  - Man kan jobbe på mye større og mer spennende data enn fra tastatur
  - Man slipper å taste det inn hver gang man kjører
- Noen filer kan imidlertid være krøkkete å lese inn
  - Å lese krøkkete formater er ikke vanskelig, bare langtekkelig.
  - I INF1000 unngår vi krøllet, og jobber med filer som er greie å lese inn.
  - I reelle situasjoner jobber man uansett ofte med store systemer hvor innlesing er godt tilrettelagt.

Lese fra fil er nesten som å lese fra tastatur:

- Helt likt:
  - 1: import java.util.Scanner;
  - 2: Scanner minScanner;
  - 5: minScanner.nextLine();
- Unikt for tastatur:
  - 3: minScanner = new Scanner(System.in);
- Unikt for fil:
  - 3: File minFil = new File("mittFilnavn.txt");
  - 4: minScanner = new Scanner(minFil);

Exceptions

- Når man jobber med filer kreves en liten utvidelse av main:

  public static void main(String[] args) throws Exception{...}

- Exceptions er nyttig og viktig, men ikke pensum i INF1000
  - Er en måte å håndtere at f.eks. programmet ikke finner filen den skal lese.

Exceptions (forts)

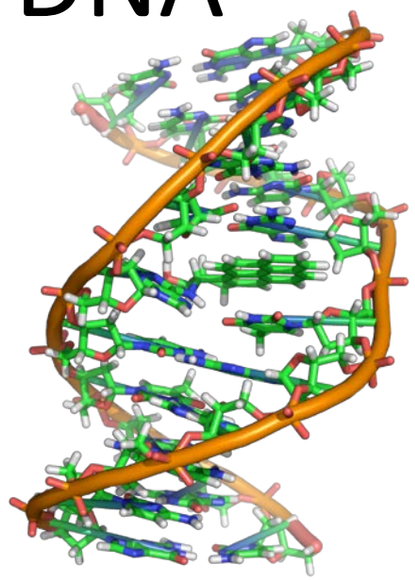
- Vi nøyer oss med en overfladisk tilnærming:
  - I alle metoder som åpner filer, legg til "throws Exception" bak parentesene etter metodenavnet.
  - Dersom en metode bruker (kaller) en annen metode som har "throws Exception" blir den 'smittet' og må selv legge det til.
  - Det er ikke noe problem å legge det til én gang for mye.

Hallo fra fil

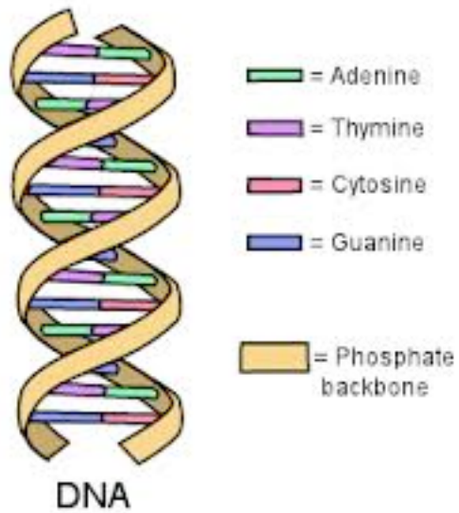
- U5LesHalloVerden.java

Gjøre noe med ekte data!

- Oppskriften på et menneske ligger i vårt DNA
  - En lang kjede av millioner av molekyler kalt nukleinsyrer, langs en dobbel heliks



- DNA er imidlertid essensielt bare bygget av fire ulike slike molekyler
  - Kan representere hvert molekyl med en bokstav A, C, G eller T
  - Hele arvemateriale er dermed bare en lang tekststreng



acggact

Et reelt spørsmål (forts)

- I og med at vårt arvemateriale essensielt sett bare er en lang sekvens av 4 molekyler (bokstaver), er det da også like mange av hvert slikt molekyl (bokstav)?
  - La oss skrive et program som teller antallet av hvert molekyl (bokstav) i vårt menneskelige DNA
- Nå kan vi glemme alt om DNA igjen!
  - Alt vi trenger å tenke på er en tekststreng som består av 4 ulike bokstaver, og vi skal telle antallet av hver
  - Ligger som en vanlig tekstfil med 1 bokstav per linje

Et reelt spørsmål (forts)

- Men start aldri med de ekte dataene!
  - Vi får det ikke nødvendigvis til på første forsøk
  - Det er ikke lett å kontrollregne manuelt på milliarder av bokstaver
- Vi forenkler:
  - Lag en liten tekstfil med ca 10 bokstaver
  - Skriv koden, se at den kompilerer og kjører underveis, og at den gir svaret vi forventer
- Og husk: bruk alltid equals for å sammenligne tekst
- U5DnaBokstaver.java

Et mer alvorlig eksempel

(samt litt mer om filer og tekst, inkludert en forsmak på objekter)

Problemstilling

- Inneholder en spyttprøve DNA-sekvensen TGA?
- Dvs, inneholder en tekstfil ordet "TGA"?

Fremgangsmåte

- Les inn tekst fra filen til en variabel av type String
- Sjekk om "TGA" forekommer inni denne strengen

Litt mer om String

- En String er mer enn bare teksten den representerer
  - En String-verdi er egentlig et objekt (mer i neste uke!)
- String har instans-metoder man kan kalle (mer senere)
  - (dette har vi egentlig allerede sett med Scanner sin nextLine)
- Vi gir her en liten forsmak på hva String tilbyr
  - "hallo verden".length() gir lengden (12)
  - "hallo verden".substring(1,3) gir tekst fra pos 1-2 ("al")
  - "hallo verden".indexOf("er") gir posisjonen til "er" (7)

Et program som finner TGA

- U5FinneTGA1.java

Tilbake til det alvorlige

- Med fila "HappyEnding.txt" fikk vi -1
  - "TGA" finnes ikke i teksten, og kroppen lager et livsviktig protein LGR4
- Med fila "BadEnding.txt" fikk vi 500
  - "TGA" finnes (fra posisjon 500), proteinet LGR4 ødelegges, og man får stor risiko for benskjørhet, kreft og mere
- Er eksempelet realistisk?
  - Faktisk! Firmaer har lov å lese teksten til dine gen, men har ikke lov å sjekke om f.eks. "TGA" finnes der
  - Men ingen bekymring: "BadEnding" er heldigvis svært sjelden..

Oppsummering

- For-løkke er egentlig bare en slags forenklet, typisk versjon av while-løkke
- Metoder med returverdier er behagelig
  - Tillater å tenke overordnet først, og deretter ordne detaljene
- Å hente data fra filer er gøy!
  - Kan jobbe med store og reelle data, uten tasting