Innlesing fra fil og metoder med returverdier

Løse reelle problemer

INF1000, uke4
Geir Kjetil Sandve

En merknad angående kompleksitet

- Jeg vil i dag vise mer komplekse problemstillinger enn dere er forventet å kunne løse selv per nå
- Jeg vil med dette gi et inntrykk av hvordan sentrale konstrukter (bl.a. metoder, løkker og arrayer) kan brukes i reelle problemer
- Målet er at dere fanger opp essensen i dette og kan anvende innsikten til å løse enklere problemer

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 også være krøkkete å lese inn
  - Å lese krøkkete formater er ikke vanskelig, bare langtekkelig
  - I faget 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 en fil den skal lese

Exceptions (forts)

- Vi nøyer oss med en overfladisk tilnærming:
- I alle metoder som åpner filer, legg til "throws Exception" bak parantesene 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

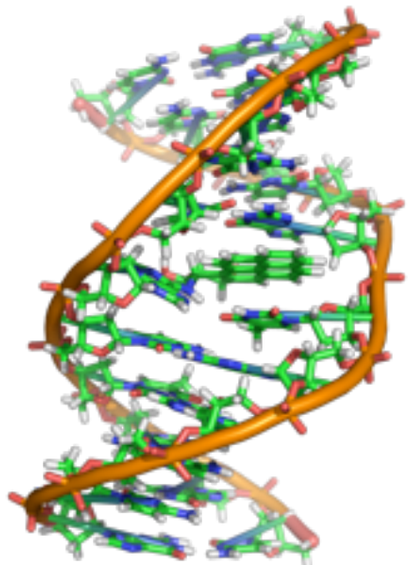
- {U4LesHalloVerden.java}

Modeller og representasjon

- Ting i verden er alltid komplekse
  - Når man programmerer tar man bare med det som trengs for å løse et gitt problem
- Å representere alt ved et menneske - fra kropp til sinn - er umulig
  - Ofte er det nok med navn og fødselsnummer..

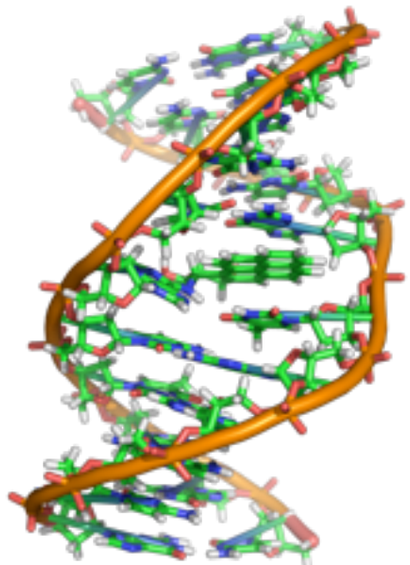
Representere vårt arvemateriale

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



Representere vårt arvemateriale

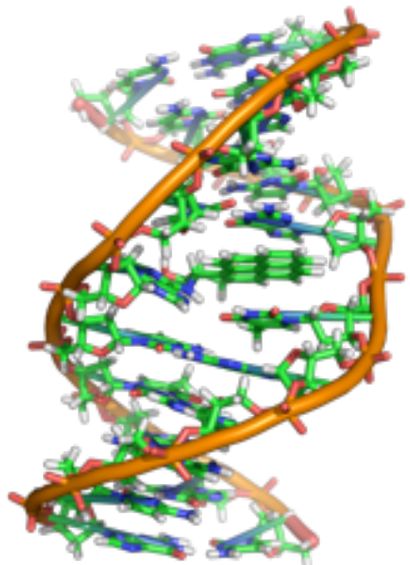
- Oppskriften på et menneske ligger i vårt DNA
  - En lang kjede av millioner av molekyler kalt nukleinsyrer, langs en dobbel heliks



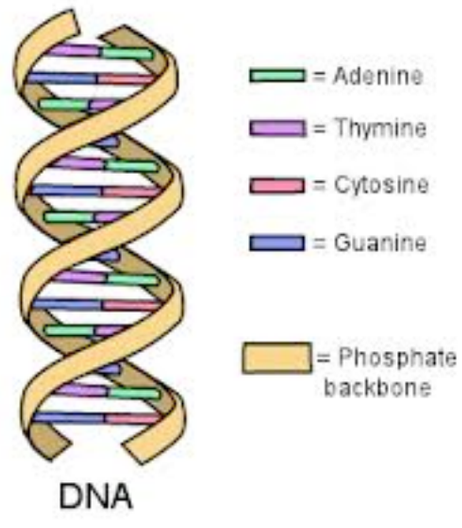
- DNA er imidlertid essensielt bare bygget av fire ulike slike syrer

Representere vårt arvemateriale

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

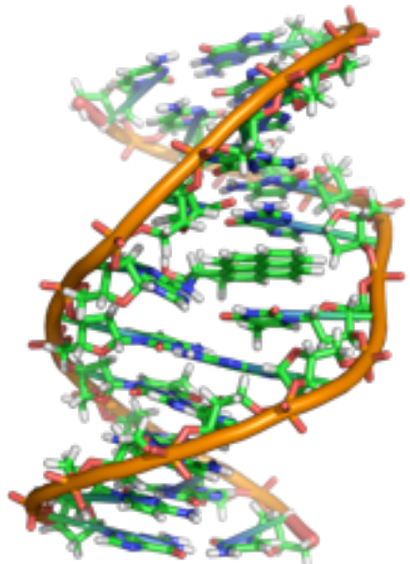


- DNA er imidlertid essensielt bare bygget av fire ulike slike syrer
  - Kan representere hver syre med en bokstav A, C, G eller T

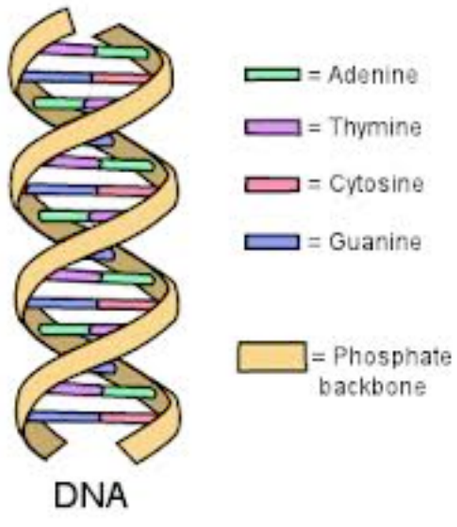


Representere vårt arvemateriale

- 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 syrer
  - Kan representere hver syre 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 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 hver bokstav i vårt menneskelige DNA

Et reelt spørsmål (forts)

- Men start for all del ikke der!
  - Vi får det ikke nødvendigvis til på første forsøk
  - Det er ikke lett å kontrollregne manuelt på milliarder av bokstaver
- Lite eksempel først
  - 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
- {U4DnaBokstaver.java}

Et reelt spørsmål (forts)

- Stor kjøring
  - Slik vi har programmert og representert dataene vil hele menneskets DNA (6GB) ta ca en halvtime
  - Upraktisk her og nå, men absolutt kjørbart
- Vi teller i stedet gjennom DNA i kromosom 21
  - Vårt minste kromosom, men ganske stort likevel (nesten 50 millioner bokstaver)
  - 10 millioner A og T - 7 millioner C og G

Representasjon avhenger av formål!

- Hvor mye av vårt DNA er gener?
  - DNA

a c g g t a t c c a

Representasjon avhenger av formål!

- Hvor mye av vårt DNA er gener?
  - DNA
  - Gener

a c g g t a t c c a

Representasjon avhenger av formål!

- Hvor mye av vårt DNA er gener?
  - DNA
  - Gener
- Kun lokasjoner (utguling) er av betydning - bokstaver uviktig

Representasjon avhenger av formål!

- Hvor mye av vårt DNA er gener?
  - DNA
  - Gener
- Kun lokasjoner (utguling) er av betydning - bokstaver uviktig

0 0 1 1 1 0 0 1 1 0

Representasjon avhenger av formål!

- Hvor mye av vårt DNA er gener?
  - DNA
  - Gener
- Kun lokasjoner (utguling) er av betydning - bokstaver uviktig

0 0 1 1 1 0 0 1 1 0

{U4GenDekning.java}

Alternativ representasjon av samme informasjon

a c g g t a t c c a

0 1 2 3 4 5 6 7 8 9

Alternativ representasjon av samme informasjon

a c g g t a t c c a

0 1 2 3 4 5 6 7 8 9

2 4

7 8

Alternativ representasjon av samme informasjon

a c g g t a t c c a

0 1 2 3 4 5 6 7 8 9

2 5

7 9

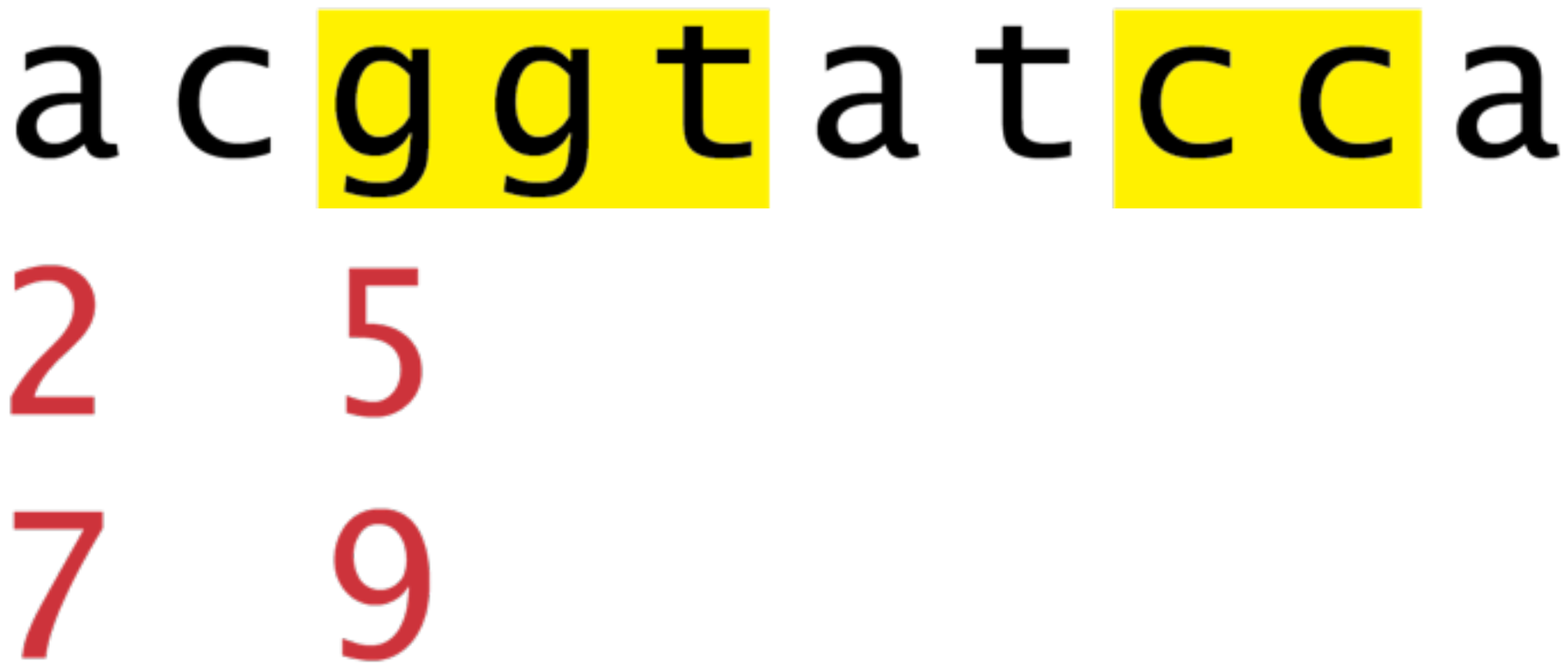
Alternativ representasjon av samme informasjon

2 5

7 9

Alternativ representasjon av samme informasjon

- Mer sparsommelig representasjon av det samme - ca 200 000 linjer, sammenlignet med 3 milliard
- Vi kan da kjøre på hele genomet
- Kun en liten hake: store tall ..



Datatyper og representasjon

- Tall er representert vha brytere på en datamaskin (mer i uke 10)
  - Int benytter 32 slike brytere, og kan representere tall opp til ca 2 milliard
  - Genomet er 3 milliard!
- Vi må sette av noe mer plass (brytere) for å representere store nok tall
  - Long bruker 64 brytere, og takler opp til trillioner. Oppfører seg ellers likt.

Hvor mye av vårt DNA er gener? (forts)

- {U4GenIntervaller.java}

Metoder med returverdi

- Void-metodene fra uke2
  - static void minMetode(...) {...}
- Metode med returverdi
  - static **DATATYPE** minMetode(...) {...}
  - static **int** gangMedTo(int innTall) {return innTall*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);
- static String lagVelkomst
  - String velkomst = lagVelkomst("inf1000", "Geir Kjetil");
  - System.out.println(velkomst);

Eksempel på forenkling av kode vha metode

- Forenkle summering av to tall som gjort i uke 1:
  - {U4InputSum1.java-U4InputSum2.java}

Med/uten parametre/returverdi

- En litt filosofisk distinksjon
  (ikke forventet å ta poenget nå, men nyttig på sikt)
- **Void**-metoder **uten** parametre:
  - Handler stort sett om kontrollflyt
- **Void**-metoder **med** parametre:
  - Handler stort sett om tilpasset gjenbruk av kode
- Metoder **med returverdi** (og typisk med parametre):
  - Sender noe tilbake, basert på hva det fikk inn (transformerer)

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

Arrangere barnebursdag: feil nivå av planlegging

Gå til bilen

Kjør til butikken

Gå til handlevogner

Dersom man mangler tier: gå til kassen, be om å få veksle, gå tilbake

Trill handlevognen bort til hylleseksjon med sukker

Legg sukker oppi handlevognen

Trill handlevognen til hylleseksjon med hvetemel

...

Arrangere barnebursdag: riktig nivå av planlegging

```
Send ut invitasjoner

Kjøp alle ingredienser som trengs

Bak alle kaker

Dekk på bord

Rydd bort alt knuselig i huset

..

Send ut invitasjoner:

    Bestem tidspunkt

    Lag dokument med invitasjon

    Skriv ut invitasjonen i X eksemplar

...
```

Hierarkisk problemløsning

- Løs problemet først på et håndterlig abstraksjonsnivå
  - Ikke mer detaljert enn at man har oversikt over stegene
  - Hvert steg kan ofte være et metodekall (eller kommentar)
- Deretter fyll inn detaljene
  - Skriv selve metodene, en om gangen, og gjør eventuelle justeringer som trengs
- Vi er nå klare for å angripe en litt større problemstilling!

Multippel Sklerose og immun-celler

- Formål med å nå vise stort eksempel:
  - Vise hvordan gå fra et reelt problem til et program
  - Vise mer omfattende innlesing fra fil
  - Vise metoder ved en skala hvor de kommer til sin rett
  - Gi en forsmak på mer avansert løkking (nøstede løkker)
  - Vise hvordan man løser et komplekst problem i flere steg
  - Vise hvordan man (jeg) typisk gjør en haug med feil, og (forhåpentligvis) retter opp disse

Vi har ikke nøyaktig likt DNA..

- Tidligere snakket vi om "menneskets DNA".
  - Samtidig er vi jo alle ulike, ikke minst fordi vårt DNA varierer
- Visse former for variasjon har bestemte konsekvenser
  - Varianter bestemmer øyenfarge, melketoleranse osv
- Noen deler av genomet er veldig viktige for sykdom
  - Ved å samle tusenvis av pasienter, kan man finne varianter som går igjen (kun) hos de som har en bestemt sykdom

Tenke med hjertet

- Rent biologisk gir det ikke så mye mening
  - Likevel: hjerte- og hjerne-celler har samme DNA
- DNAet er likt, men bruken av det ulik
  - For hver celle er det visse deler av DNA som brukes mer aktivt enn annet
- Man kan med ulike teknikker måle hvilke deler av DNA som er aktivt i en gitt type celler
  - Resulterer i en samling regioner av genomet (samme representasjon som vi så for gener tidligere)

I hvilke celler slår en arvelig sykdom ut

- Dersom endringer i DNA fører til sykdom, må dette skje i bestemte celler i kroppen
- For at endring i en gitt lokasjon skal påvirke en celle, må lokasjonen være i et område som er i aktiv bruk i denne typen celle

Problemet vi skal løse

- Medisinsk problemstilling:
  - Virker Multippel Sklerose gjennom å endre immunceller?
- Analytisk problemstilling:
  - Ligger MS-assosierte lokasjoner uforholdsmessig mye i områder av genomet som er aktive i immunceller?
- Programmerings-problemstilling:
  - Ligger koordinatene (tallene) fra en fil ofte inni intervaller definert av to andre filer

Første fase:
Skrive ned overordnet fremgangsmåte

- Les inn MS-lokasjoner til en array
  - Hvor stor array?: Tell antall linjer i MS-fil
  - Les inn MS-lokasjoner fra fil til array
- Les inn immun-intervaller
  - Finn størrelse og les inn startpunkt og sluttpunkt til arrayer
- Tell antall MS-lokasjoner innenfor immun-intervaller
- Simuler tilfeldige MS-lokasjoner og beregn p-verdi
- {U4ImmuneAndMS1.java}

Andre fase:

- Tell antall linjer i hver fil
- (midlertidig skriv ut antallet MS- og immun-lokasjoner)
- {U4ImmuneAndMS2.java}

Tredje fase

- Lag arrayer av riktig størrelse for MS og immun
- Les inn MS-lokasjoner og immun-intervaller
- (midlertidig skriv ut en bestemt MS-lokasjon og immun-intervall)
- {U4ImmuneAndMS3.java}

Fjerde fase

- Tell hvor mange av MS-lokasjonene er inni immun-intervall
- (Midlertidig skriv ut hvor mange MS-lokasjonene som er innenfor immun)
- {U4ImmuneAndMS4.java}

Femte fase:

- Simuler tilfeldige MS-lokasjoner 500 ganger, og sjekk hvor ofte de tilfeldige havner like mye inni (p-verdi)
- {U4ImmuneAndMS5.java}

Konklusjon på eksempel

- MS-lokasjoner ligger uventet mye i aktive immun-regioner
- -> Immunceller virker sentrale for Multippel Sklerose
- -> Kan forsøke å utvikle medisiner mot MS som kun påvirker immun-celler

Oppsummering

- Å hente data fra filer er gøy!
  - Kan jobbe med store og reelle data, uten tasting
- Metoder med returverdier er behagelig
  - Tillater å tenke overordnet først,
    og deretter ordne detaljene
- Man kan besvare store spørsmål med programmering
  - Veldig mye fremgang i samfunnet er avhengig av programmering