

INF1000 – Forelesning 7

Litt om objekter, filer med easyIO, tekst

1

Spørsmål fra forrige forelesning

- Operator-presedens i Java?
- Hvordan virker metodene `inWord()/inInt()/etc.` i `In`-klassen i `easyIO` når vi skriver inn flere verdier på tastaturet?

2

Operator-presedens i Java

- Uttrykk evalueres fra venstre mot høyre
- Enkelte operatører har høyere presedens enn andre

↑ høyere presedens

- ++ -- ! new
- * / %
- + -
- < <= > >=
- == !=
- &&
- ||
- =

NB! Sett parenteser!

- Høyere presedens = høyere prioritet
- Eksempel: $7-4+2 = (7-4)+2 = 5$
- Eksempel: $2*3-4/2 = (2*3)-(4/2) = 4$

3

Oppførselen til `inWord()/inInt()/etc.`

- Input som ikke "brukes" ved ett kall på `inWord()` spares i et buffer til senere

```
In tast = new In();
System.out.print("Skriv to ord: ");
String a = tast.inWord();
String b = tast.inWord();
System.out.println(a);
System.out.println(b);
```

Her stopper programmet og venter på input. Anta at brukeren taster: **ball hus**
Første kall på `inWord()` returnerer **ball**

- Programkjøring:

```
Skriv to ord: ball hus
ball
hus
```

Andre kall på `inWord()` vil ikke vente på input fordi det ligger ubrukte tegn i bufferen: **hus**
Andre kall returnerer derfor **hus** direkte.

4

Dagens forelesning

- Litt om objekter, pekere og **null**
- Filer og easyIO.
- Litt mer om tekster

5

Objekter, pekere og null

Vi lager *pekere* og *objekter* når vi bruker arrayer og klasser

- `int [] a = new int [1024];`
- `Student s = new Student();`
- `String navn = "Jens";`
- Selve variabelen (`a`, `s`, `navn`) er en peker som:
 - Inneholder adressen til hvor objektet (array-objektet eller objektet fra klassen) er i hukommelsen .
 - Tegnes som en pil
- Hva om vi *ikke* har laget et objekt ?
 - `int [] b ;`
 - `Student t;`
 - `String etternavn;`
- Hva peker de på? Svar: *ingenting!*
- Denne 'ingenting' verdien heter **null** og kan testes på og brukes i tilordninger:


```
if (navn == null) print("navn mangler");
if (b != null) print("arrayen b finnes");
if (a != null && a[0] > 10) print("a[0] er stor nok");
else print ("enten finnes ikke arrayen a eller a[0] er for liten");
s = null;
navn = null;
```

6

Lese og skrive fra/til fil

- Klassene In og Out i easyIO
- Les dokumentasjonen
 - In og Out + Format
 - brukes i INF1000
 - Format brukes til mer 'finjustert' formattering
 - InExp og OutExp
 - gir feilmeldinger hvis du gjør noen feil
 - vanskeligere å bruke enn In og Out
 - blir vanskeligere kode, brukes noe i INF1010
 - Det er langt flere metoder enn de som gjennomgås her
- easyIO ble laget fordi javas innebygde IO-metoder var for kompliserte
 - bedre nå, men fortsatt noe vanskeligere enn easyIO

7

Eksempel

```
import easyIO.*;

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

        In fil = new In("filnavn");
        String s = fil.inLine();

        System.out.println("Første linje var: " +
            s);
    }
}
```

Vi importerer pakken easyIO.

Vi åpner filen for lesing

Her leses hele første linje av filen

8

Tre måter å lese på

- **Ord for ord** (tall for tall, ...)
 - Leser med `inDouble()`, `inInt()`, `inWord()`, osv
 - Ser etter slutten med `lastItem()`
 - Hopper over skilletegn (= hoppeover-tegn mellom ordene man leser)
 - linjeskift-tegnene (+ noen sære tegn) er alltid skilletegn
 - Hvis man ikke gjør noe er også blanke, tab,... skilletegn
 - Brukeren kan også spesifisere skilletegn: `i = inInt("F(,");`
(da er bare linjeskift + de som spesifiseres F(,) slike tegn det hoppes over *for* det leses neste 'ord')
- **Tegn for tegn** (Ikke så mye brukt)
 - Leser med `inChar()` [`inChar(false)`], men
 - ikke det samme som `inChar(true)`
 - Ser etter slutten med `endOfFile()`
 - Kan kopler en fil tegn for tegn uansett hva den inneholder
- **Linje for linje**
 - Leser med `readLine()`
 - Ser etter slutten med `endOfFile()`
- **Lesemåtene kan blandes** (hvis man vet hva man gjør)

9

Lese ord for ord (item)

- Metoder:
 - `inInt()` for å lese et heltall
 - `inDouble()` for å lese et flyttall
 - `inWord()` for å lese et ord
 - `lastItem()` for å sjekke om slutten av filen er nådd
- Eksempel: lese en fil tall for tall

```
In fil = new In("item.txt");
while (!fil.lastItem()) {
    int k = fil.inInt();
    System.out.println("Tallet var " + k);
}
```

10

Lese linje for linje

- Metoder:
 - `readLine()` for å lese en linje
 - `inLine()` for å lese resten av en linje (leser neste linje hvis det ikke er mer igjen enn linjeskift på nåværende linje)
 - `endOfFile()` for å sjekke om slutten av filen er nådd
- Eksempel: lese en fil linjevis

```
In fil = new In("fil.txt");
while (!fil.endOfFile()) {
    String s = fil.readLine();
    System.out.println("Linjen var " + s);
}
```

11

Program som leser en tekstfil linje for linje:

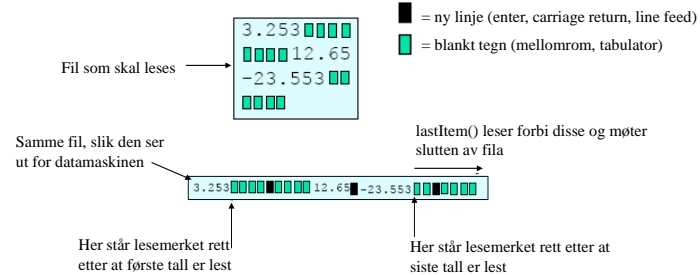
```
import easyIO.*;
class LinjeForLinje {
    public static void main (String[] args) {
        In innfil = new In("filnavn");
        String[] s = new String[100];
        int ant = 0;
        while (!innfil.endOfFile()) {
            s[ant] = innfil.readLine();
            ant = ant + 1;
        }
        for (int i=0; i<ant; i++) {
            System.out.println(s[i]);
        }
    }
}
```

12

lastItem og endOfFile, lesemerket

- `endOfFile()` sjekker 'bare' om siste tegn på fila er lest
- `lastItem()` søker seg fram til første ikke-blanke tegn og returnerer **true** hvis slutten av fila nås og **false** ellers.

- Eksempel:



13

Når filens lengde er kjent

- Når et program skal lese en fil, må det ha en mulighet til å avgjøre når slutten av filen nådd – ellers kan det oppstå en feilsituasjon.
- Metodene `lastItem()` og `endOfFile()` kan benyttes til dette.
- Noen ganger er filens lengde kjent på forhånd:
 - lengden er kjent før programmet kjøres
 - lengden ligger lagret i begynnelsen av filen
- Da kan vi i stedet benytte en for-løkke.

14

Eksempel: fil med kjent lengde

Program som leser en fil med 10 desimaltall, hvor tallene er atskilt med blanke tegn og/eller linjeskift:

```

import easyIO.*;
class Les10Tall {
    public static void main (String[] args) {
        double[] x = new double[10];
        In innfil = new In("tall.txt");
        for (int i=0; i<10; i++) {
            x[i] = innfil.inDouble();
        }
        // Nå kan vi evt. gjøre noe med verdiene
        // i arrayen x
    }
}

```

15

Nok at tallene er atskilt

Programmet på forrige foil ville gitt akkurat samme resultat for alle disse filene:

```

15.2
6.23
3.522
3.6
8.893
-3.533
65.23
22.01
45.02
7.2

```

```

15.2 6.23
3.522 3.6
8.893 -3.533
65.23 22.01
45.02 7.2

```

```

15.2 6.23 3.522 3.6
8.893 -3.533
65.23 22.01 45.02
7.2

```

16

Eksempel: fil med lengde-info

Program som leser en fil med desimaltall, hvor tallene er atskilt med blanke tegn og/eller linjeskift. Antall tall som skal leses ligger øverst i filen.

```
import easyIO.*;
class LesTallMedLengde {
    public static void main (String[] args) {
        double[] x; // bestemmer ikke lengden ennå
        In innfil = new In("tall-med-lengde.txt");
        int lengde = innfil.inInt();// nå vet vi lengden
        x = new double[lengde];
        for (int i=0; i<lengde; i++) {
            x[i] = innfil.inDouble();
        }
        for (int i=0; i<lengde; i++) {
            System.out.println( i + " = " + x[i]);
        }
    }
}
```

17

Eksempel: fil med sluttmerke

Program som leser en fil med desimaltall, hvor tallene er atskilt med blanke tegn og/eller linjeskift. Sluttet av filen er markert med tallet -999.

```
import easyIO.*;
class LesTallMedMerke {
    public static void main (String [] args) {
        double [] x = new double[100]; // antar max 100 tall
        In innfil = new In("tall-med-merke.txt"); // på fil
        double siste = 0;
        int ant = 0;
        while (siste != -999) {
            siste = innfil.inDouble();
            if (siste != -999) {
                x[ant] = siste;
                ant = ant + 1;
            }
        } // Nå ligger det verdier i
        // x[0], x[1], ..., x[ant-1]
    }
}
```

18

Lese en fil med mer komplisert format

- Anta at vi skal lese en fil med følgende format:
 - Først er det en linje med 3 overskrifter (separert av blanke tegn)
 - Deretter kommer det en eller flere linjer, som hver består av et heltall, et desimaltall og en tekststreng (separert av blanke tegn)
- Eksempel:

Antall	Pris	Varenavn
35	23.50	Oppvaskkost
53	33.00	Kaffe
97	27.50	Pizza
...
...

19

Fremgangsmåte

- Den første linja er spesiell, og vi tenker oss her at den ikke er så interessant - vi ønsker bare å få lest forbi den. Det kan vi gjøre med `inLine()`.
- De andre linjene har samme format, så vi kan lage en løkke hvor hvert gjennomløp av løkken leser de tre itemene på en linje. Vi bruker da henholdsvis `inInt()`, `inDouble()` og `inWord()`.
- For å vite når filen er slutt, kan vi enten bruke `endOfFile()` eller `lastItem()`. Siden vi leser filen itemvis, er det mest naturlig å bruke `lastItem()`. Da får vi heller ikke problemer dersom det skulle ligge noen blanke helt på slutten av filen – og det gjør det som oftest (ofte et siste linjeskift på siste linje).
- Vi hopper over detaljene.

20

Eksempel

Program som leser en fil med tre kolonner: en kolonne med int, en kolonne med desimaltall, og en kolonne med tekst.

```
import easyIO.*;
class LesVarer {
    public static void main (String[] args) {
        In innfil = new In("varer.txt");
        int [] t = new int[100];
        double[] x = new double[100];
        String[] s = new String[100];
        int ant = 0;
        innfil.readline();
        while (!innfil.lastItem()) {
            t[ant] = innfil.inInt();
            x[ant] = innfil.inDouble();
            s[ant] = innfil.inWord("\n");
            ant = ant + 1;
        }
        for (int i=0; i<ant; i++) {
            System.out.println(t[i] + " " + x[i] + "-" +
                s[i]);
        }
    }
}
```

21

Eksempel på å gi skilletegn ved innlesing

- Vi kan ved innlesing i easyIO spesifisere hvilke tegn vi vil hoppe over ved innlesing – i inInt, inWord, inDouble,.. kan skilletegn gis
- Anta at kollonne k og rad r skal gis som: S(r,k) – eks: S(0,4)

```
import easyIO.*;
class Skilletegn {
    public static void main (String[] args) {
        int r,k;
        String skille = " S(,)";
        In tast = new In(); Out skjerm = new Out();

        skjerm.out("Gi rad r og kollonne k som S(r,k):");
        r = tast.inInt(skille);
        k = tast.inInt(skille);

        skjerm.outln("Du ga r=" +r+", og k=" +k);
    }
}
```

22

Noen nyttige hjelpemidler (ikke pensum)

- Sjekk om det finnes en fil med et bestemt navn:

```
if (new File("filnavn").exists()) {
    System.out.println("Filen finnes");
}
```

- Slette en fil:

```
if (new File("filnavn").delete()) {
    System.out.println("Filen ble slettet");
}
```

- Avgjøre hvilket filområde programmet ble startet fra:

```
String curDir = System.getProperty("user.dir");
```

- Lage liste over alle filer og kataloger på et filområde:

```
String [] allefiler = new File("filområdenavn").list();
```

23

Skrive til fil

Vi importerer pakken easyIO.

Vi åpner filen for Skrivning

```
import easyIO.*;
class SkrivTilFil {
    public static void main (String [] args) {

        Out fil = new Out("nyfilnavn");

        fil.outln("Dette er første linje");

        fil.close();
    }
}
```

Vi må huske å lukke filen til slutt

Her skrives en linje med tekst til filen

24

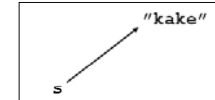
EasyIO: Hvilke skrivemetoder finnes?

Datatype	Eksempel	Beskrivelse
int	fil.out(x); fil.out(x, 6);	Skriv x Skriv x høyrejustert på 6 plasser
double	fil.out(x, 2); fil.out(x, 2, 6);	Skriv x med 2 desimaler Skriv x med 2 desimaler på 6 plasser
char	fil.out(c);	Skriv c
String	fil.out(s); fil.out(s, 6);	Skriv s Skriv s på 6 plasser (venstrejustert)
	fil.outln();	Skriv en linjeskift
	fil.close();	Lukk filen

Merk: dersom antall plasser spesifiseres og det ikke er plass til det som skal skrives ut, vil det som skrives ut avsluttes med tre punkter: ...

Tekster og klassen String

- En tekststreng er en sekvens av tegn (null, en eller flere), f.eks.
 - ""
 - "&"
 - "Arne er student"
- Hver tekststreng vi lager er et objekt av typen String
- String-objektet kan i seg selv ikke endre (Immutable).
- En String-variabel (f.eks. **String s**) er en referanse til et slikt objekt
 - Resultatet av å utføre **String s = "kake"** ;
- For å finne lengden (dvs antall tegn i) en tekst:

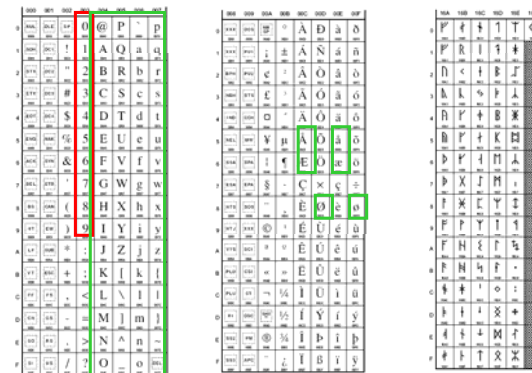


```
int lengde = s.length();
```

Bruk av spesialtegn

- Både i char-uttrykk og String-uttrykk kan vi ha mange ulike typer tegn
- Alle Unicode-tegn er tillatt
- Unicode er en standard som tillater tusenvis av tegn (ulike varianter fins; den som støttes av Java tillater 65536 ulike tegn)
- Alle tegnene kan angis som 'uxxxx' hvor hver x er en av 0, 1, 2, ..., 9, A, B, C, D, E, F
- Eksempel: 'u0041' er tegnet 'A'
- Noen spesialtegn har egen forkortelse:
 - \t tabulator
 - \r vognretur (skrivning starter først på linja)
 - \n linjeskift
 - \" dobbelt anførselstegn
 - \' enkelt anførselstegn
 - \\ bakslask

Unicode (<http://www.unicode.org>)



Teste om to tekster er like

- For å teste om to tekststrenger er like, brukes equals:
// Anta at s og t er tekstvariable (og at s ikke har verdien **null**)

```
if (s.equals(t)) {
    System.out.println("Tekstene er like");
} else {
    System.out.println("Teksten er forskjellige");
}
```

- Bruk av == virker av og til, men ikke alltid:

```
String s = "abc";
String t = "def";
String tekst1 = s + t;
String tekst2 = s + t;
```

Nå er tekst1.equals(tekst2) **true**, mens tekst1 == tekst2 er **false**.

29

De enkelte tegnene i en tekststreng

- Tegnene i en tekststreng har posisjoner indeksert fra 0 og oppover

0	1	2	3
'k'	'a'	'k'	'e'

- Vi kan få tak i tegnet i en bestemt posisjon:

```
String s = "kake";
char c = s.charAt(1);
// Nå er c == 'a'
```

- Vi kan erstatte alle forekomster av et tegn med et annet tegn:

```
String s1 = "kake";
String s2 = s1.replace('k', 'r');
// Nå er s2 en referanse til tekststrengen "rare"
```

30

Deler av en tekststreng

- Vi kan trekke ut en del av en tekststreng:

```
String s = "Paris";
String s1 = s.substring(1,4);
// Nå er s1 tekststrengen "ari"
```

0	1	2	3	4
'P'	'a'	'r'	'i'	's'

s.substring(1,4)

- Generelt:

```
s.substring(index1, index2)
```

Første posisjon som skal være med

Første posisjon som *ikke* skal være med

- Siste del av en tekststreng:

```
String s = "Paris er hovedstaden i Frankrike";
String s1 = s.substring(6);
// Nå er s1 tekststrengen "er hovedstaden i Frankrike"
```

31

Alfabetisk ordning

- Anta at s og t er tekstvariable (og at s ikke har verdien null)
- Er s foran t i alfabetet?

```
int k = s.compareTo(t);
if (k < 0) {
    System.out.println("s er alfabetisk foran t");
} else if (k == 0) {
    System.out.println("s og t er like");
} else {
    System.out.println("s er alfabetisk bak t");
}
```

Husk at det er Unicode-verdien som brukes her og at det kan gi uventet resultat!

32

Oppgave 1

- Hva skriver programmet ut?

```
import easyIO.*;
class Alfabetisk {
    public static void main (String [] args) {
        String sString = "abCDøÅ";
        String tString = "bCDøÆ";

        for(int i=0; i < sString.length();i++){
            String s = sString.substring(i, i+1);
            String t = tString.substring(i, i+1);

            int k = s.compareTo(t);

            if (k < 0) {
                System.out.print(s + " er alfabetisk foran " + t);
            } else if (k == 0) {
                System.out.print(s + " og " + t + " er like");
            } else {
                System.out.print(s + " er alfabetisk bak " + t);
            }

            System.out.println("\t\t k er " + k);
        }
    }
}
```

33

```
arnem@sviurr ~/INF1000/Programmer> java Alfabetisk
a er alfabetisk foran b      k er -1
b er alfabetisk bak C       k er 31
C er alfabetisk foran d     k er -33
D og D er like              k er 0
ø er alfabetisk bak Ø       k er 32
A er alfabetisk foran Æ     k er -1
```

Inneholder en tekst en annen?

- Anta at s og t er tekstvariable (og at s ikke har verdien null)
- Inneholder s teksten t?

```
int k = s.indexOf(t);
if (k < 0) {
    System.out.println("s inneholder ikke t");
} else {
    System.out.println("s inneholder t");
    System.out.println("Posisjon i s: " + k);
}
```

34

Starter en tekst med en annen?

- Anta at s og t er tekstvariable (og at s ikke har verdien null)
- Starter s med teksten t?

```
boolean b = s.startsWith(t);
if (b) {
    System.out.println("s starter med t");
} else {
    System.out.println("s starter ikke med t");
}
```

35

Slutter en tekst med en annen?

- Anta at s og t er tekstvariable (og at s ikke har verdien null)
- Slutter s med teksten t?

```
boolean b = s.endsWith(t);
if (b) {
    System.out.println("s ender med t");
} else {
    System.out.println("s ender ikke med t");
}
```

36

Fra tall til tekst og omvendt

- For å konvertere fra tall til tekst:

```
String s1 = String.valueOf(3.14);
String s2 = String.valueOf('a');
String s3 = String.valueOf(false);
String s4 = "" + 3.14;
String s5 = "" + 'a';
String s6 = "" + false;
```

- For å konvertere fra tekst til tall:

```
int k = Integer.parseInt(s);
double x = Double.parseDouble(s);
```

og tilsvarende for de andre numeriske datatypene...

37

Oppgave 2

- Hva skriver programmet ut?

```
import easyIO.*;
class SlutterMedOppgave {
    public static void main (String [] args) {
        String s = "julenisse";
        String t = s.substring(4);
        String v = s.substring(0,4);
        if(s.startsWith("jule")){
            System.out.println("A");
        }
        if(t.startsWith("jule")){
            System.out.println("B");
        }
        if(v.startsWith("jule")){
            System.out.println("C");
        }
    }
}
```

```
$ java SlutterMedOppgave
A
C
```

38

Oppgave 3

- Hva skriver programmet ut?

```
import easyIO.*;
class ReplaceOppgave {
    public static void main (String [] args) {
        String s = "javaprogram";
        String l = s;
        s.replace('a', 'i');
        l.replace('a', 'o');
        System.out.println(s);
        System.out.println(l);

        l = "jp";
        System.out.println(s);
    }
}
```

```
$ java ReplaceOppgave
javaprogram
javaprogram
javaprogram
```

39