



INF1000 – Uke 5

Litt om objekter, filer med easyIO, tekst

1



3 emner i dag!

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

2



Objekter og pekere

- 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 er en peker som:
 - Inneholder adressen til hvor objektet er i hukommelsen
 - Tegnes som en pil

3



null

- Hva om vi *ikke* har laget et objekt ?
 - `int [] b ;`
 - `Student s;`
 - `String navn;`
- Hva peker de på? Svar: *ingenting!*
- Denne 'ingenting'-verdien heter `null`

4

null igjen

`null` kan testes på og brukes i tilordninger:

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

5

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
 - Det er flere metoder enn de som gjennomgås
- easyIO ble laget fordi Javas innebygde IO-metoder var for kompliserte
 - Java bedre nå
 - men fortsatt noe vanskeligere enn easyIO

6

Eksempel

Vi importerer pakken easyIO.

```
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 åpner filen for lesing

Her leses hele første linje av filen

7

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);
}
```

8

Lese ord for ord: skilletegn

Hopper over skilletegn 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:
 - `String egneSkilletegn = "F(,)"`;
 - `i = inInt(egneSkilletegn)`;
 - **før** det leses neste 'ord' nå leses ignoreres kun linjeskift samt tegnene i 'egneSkilletegn'

9

Lese 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 kopiere en fil tegn for tegn uansett hva den inneholder

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:

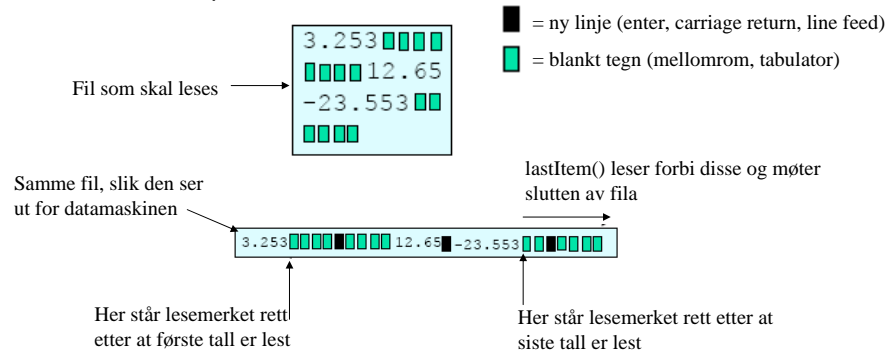
```
import easyIO.*;
class LinjeForLinje {
    public static void main (String[] args) {
        In innfil = new In("filnavn");
        String[] s = new String[100];
        int ant = 0;
        while (!innfil.endOfFile()) {
            s[ant] = innfil.readLine();
            ant = ant + 1;
        }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

- Vi må ha 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

- Lag program som leser en fil med
 - 10 desimaltall
 - tallene er atskilt med blanke tegn og/eller linjeskift
- Algoritme:
 - For-løkke med 10 gjennomløp
 - Bruk **inDouble**-metoden fra **easyIO**

15

Eksempel: fil med kjent lengde

```
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++)
        {
```

16

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
```

17

Eksempel: fil med lengde-info

- Lag program som leser en fil med
 - på forhånd ukjent antall desimaltall
 - tallene er atskilt med blanke tegn og/eller linjeskift
 - antall tall som skal leses ligger øverst i filen
- Algoritme:
 - Les antall tall fra fila
 - For-løkke med så mange gjennomløp som det er tall
 - Bruk inDouble-metoden fra easyIO

18

Eksempel: fil med lengde-info

```
import easyIO.*;
class LesTallMedLengde {
    public static void main (String[] args) {
        // vet ikke lengden ennå
        double[] x;
        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]);
        }
    }
}
```

19

Eksempel: fil med sluttmerke

- Lag program som leser en fil med
 - på forhånd ukjent antall desimaltall
 - tallene er atskilt med blanke tegn og/eller linjeskift
 - slutten av filen er markert med tallet -999
 - antall tall er max 100
- Algoritme:
 - while-løkke inntil -999 leses
 - Bruk inDouble-metoden fra easyIO

20

Eksempel: fil med sluttmerke

```
import easyIO.*;
class LesTallMedMerke {
    public static void main (String [] args) {
        // antar max 100 tall på fil
        double [] x = new double[100];
        In innfil = new In("tall-med-merke.txt");
        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]
    }
}
```

21

Lese en fil med mer komplisert format

Anta at vi skal lese en fil med følgende format:

- Først en linje med 3 overskrifter
- Deretter en eller flere linjer på formen:
 - heltall, desimaltall, tekststreng
- Alle felt er separert av blanke tegn

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

22

Fremgangsmåte

- Den første linja er spesiell
 - vi tenker oss her at den ikke er så interessant
 - vi leser forbi den med `readLine()` eller `inLine()`
- De andre linjene har samme format,
 - løkke hvor hvert gjennomløp av løkken leser de tre itemene
 - bruker da henholdsvis `inInt()`, `inDouble()` og `inWord()`.
- For å vite når filen er slutt:
 - kan enten bruke `endOfFile()` eller `lastItem()`
 - vi leser filen itemvis, bruker derfor `lastItem()`
 - da får vi ikke problemer med blanke helt på slutten av filen
 - ofte et siste linjeskift på siste linje!

23

```
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]);
        }
    }
}
```

24

Eksempel på å gi skilletegn ved innlesing

Anta at kolonne k og rad r på form: 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 kolumnen k som
                    S(r,k):");
        r = tast.inInt(skille);
        k = tast.inInt(skille);

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

Noen nyttige hjelpemidler (ikke pensum)

- Sjekke 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();
```

Skrive til fil

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

        Out fil = new Out("nyfilnavn");

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

        fil.close();
    }
}
```

Vi importerer pakken easyIO.

Vi åpner filen for Skrivning

Vi må huske å lukke filen til slutt

Her skrives en linje med tekst til filen

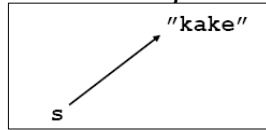
EasyIO: Hvilke skrivemetoder finnes?

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 punktumer: ...

Tekster og klassen String

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



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

29

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)

30

Bruk av spesialtegn

- 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

31

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

The image displays three sections of the Unicode character set grid. The first section shows the range 000-07F, highlighting the first row (0) with a red box. The second section shows the range 080-0FF, highlighting the first row (0) with a green box. The third section shows the range 100-10F, highlighting the first row (0) with a red box.

32

equals() tester om to tekster er like

- Anta at s og t er tekstvariable (og s ikke er **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**.

33

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"
```

34

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"
```

35

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!

36

Oppgave 1

- Hva skriver programmet?

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

        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();
        }
    }
}
```

```
M:\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 Æ     k er 32
Å 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);
}
```

38

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");
}
```

39

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");
}
```

40

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...

41

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");  
        }  
    }  
}
```

42

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);  
    }  
}
```

43