

Programmering 1999

Forelæsning 6, fredag 17. september 1999

- Tabeller (arrays)
- Erklæring, oprettelse, og indeksering af tabeller
- Eksempler med tabeller af heltal
- Eksempler med tabeller af tegnstreng
- Opbygning af søjlediagrammer (histogrammer)
- Primitiv krydning ved substitution
- StringBuffer: effektivt udvidelige tegnstreng

Længden af en tabel

Længden af en tabel beregnes med:

```
days.length
```

I eksemplet er `days.length` lig med 12.

Indeksering (opslag) i tabellen

Skal indholdet af plads 3:

```
days[3]
```

Sæt plads nummer 1 til 29:

```
days[1] = 29;
```

Udtrykket `[...]]` kaldes for *indekser*.

Lovlige indekxsværdier er 0, 1, 2, ..., `days.length-1`.

Bruges et ulovligt indeks afbrydes programmet med en fejlmedling:

```
java.lang.ArrayIndexOutOfBoundsException: ...
```

Tabeller af heltal

En tabelvariabel er en variabel med plads til flere værdier, f.eks.:

0	1	2	3	4	5	6	7	8	9	10	11
---	---	---	---	---	---	---	---	---	---	----	----

Værdierne (elementerne) er nummereret 0, 1, 2, ...

Erklæring af tabelvariabel (opretter ikke selve tabellen):

```
int[] days;
```

Erklæring, oprettelse og initialisering af tabel:

```
int[] days = { 31, 28, 31, 30, 31, 31, 31, 30, 31, 30, 31 };
```

Oprettelse af tabel med 12 pladser, alle initialiseret til 0:

```
days = new int[12];
```

Pladserne er nummereret 0, 1, 2, ..., 11:

Samtidig erklæring og oprettelse af tabel:

```
int[] days = new int[12];
```

Givet månedsnummeret (1–12), find antal dage i måneden

Lav en tabel over månedernes længde.

Slå op i tabellen med månedsnummeret minus én:

```
static int monthLen(int mth) {
    int[] days = { 31, 28, 31, 30, 31, 31, 30, 31, 31, 30, 31, 31 };
    return days[mth-1];
}
```

En tabel kan initialiseres med en tabelkonstant { 31, 28, ..., 31 }.

Et program til at finde månedslængder

```
public class Days1 {  
  
    public static void main(String[] args) {  
        int mth = Integer.parseInt(args[0]);  
        System.out.println("Måned " + mth + " har længde " + monthlen(mth));  
    }  
  
    static int monthlen(int mth) {  
        int[] days = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };  
        return days[month-1];  
    }  
}
```

Hvad sker der hvis man kører java Days1 13?

Programmering 1999

KVL

Side 6-5

Givet månedsnavn, find månedens nummer (1-12) i året
Lav en tabel over månedsnavnene (som tegnstreng).

Gennemløb tabellen fra en ende at, indtil det givne navn findes.

```
public class Days3 {  
  
    public static void main(String[] args) {  
        String monthname = args[0];  
        System.out.println(monthname + " har nummer " + monthno(monthname));  
    }  
  
    static int monthno(String monthname) {  
        String[] name = { "januar", "februar", "marts", "april", "maj",  
            "juni", "juli", "august", "september",  
            "oktober", "november", "december" };  
        int i = 0;  
        while (!monthname.equals(name[i]))  
            i=i+1;  
        return i+1;  
    }  
}
```

Husk: s.equals(t) er true hvis s og t indeholder samme tegnfølge.

Hvad sker der hvis man kører java Days3 januar?

Programmering 1999

KVL

Side 6-7

Givet månedsnummer (1-12) og dato, find dagens nummer i året

Summér antal dage i månederne forud for den givne måned, og læg datoen til.

```
public class Days2 {  
  
    public static void main(String[] args) {  
        int day = Integer.parseInt(args[0]);  
        int mth = Integer.parseInt(args[1]);  
        System.out.println(day + "/" + mth + " er dag nummer "  
            + dayinyear(mth,day));  
    }  
  
    static int dayinyear(int mth, int day) {  
        int[] days = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };  
        int dayno = 0;  
        for (int i=0; i < mth-1; i=i+1)  
            dayno = dayno + days[i];  
        return dayno + day;  
    }  
}
```

Hvad sker der hvis man kører java Days2 32 1?

Programmering 1999

KVL

Side 6-6

Parametereen args i main-metoden

I main-metoden public static void main(String[] args) er args en tabel af argumenterne fra kommandolinien.

Så args.length er antallet af argumenter.

Og args[0] er det første argument, args[1] er det andet argument, osv

```
public class Args {  
  
    public static void main(String[] args) {  
        if (args.length != 2)  
            System.out.println("Fejl: Dette program kræver to argumenter");  
        else {  
            int x1 = Integer.parseInt(args[0]);  
            int x2 = Integer.parseInt(args[1]);  
            System.out.println("Summen er " + (x1 + x2));  
        }  
    }  
}
```

Opgave: et program der kan tage vilkårlig mange argumenter og lægge dem sammen.

Programmering 1999

KVL

Side 6-8

Optælling af søjlediagram (histogram) ved hjælp af tabel

Med en tilfældigstagsgenerator kan en metode kaste en terning:

```
static Random rnd = new Random();

static int rolldie() {
    return 1 + (int)(6 * rnd.nextDouble());
}
```

Hvordan fordeler udfaldene sig ved 6000 kast med en terning?

Dvs. hvor tilfældig er tilfældigstagsgeneratoren?

Brug en tabel `freq` til at indeholde antallet af forekomster af 1, 2, 3, 4, 5, 6.

Element `freq[1]` er antallet af enere, `freq[2]` er antallet af toere, osv.

Til at begynde med er alle antal 0.

Når terningen viser 4, så forøg `freq[4]` med 1. Dvs., udfør

```
freq[4] = freq[4] + 1;

I almindelighed, når terningen viser res, udfør
freq[res] = freq[res] + 1;
```

Programmering 1999

KVL

Side 6-9

Primitiv kryptering

Kryptering ved Caesar-substitution: udskift `a` med `n`, `b` med `o`, `c` med `p`, osv.

```
public class Caesar1 {
    public static void main(String[] args) {
        String orig = args[0];
        System.out.println(encrypt(orig));
    }

    static char[] codes = { 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v',
        'w', 'x', 'y', 'z', 'a', 'b', 'c', 'd', 'e',
        'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm' };

    static String encrypt(String s) {
        String res = "";
        for (int i=0; i<s.length(); i=i+1)
            res = res + codes[s.charAt(i) - 'a'];
        return res;
    }
}
```

Tegn lægges som tal i maskinen. F.eks. lægges 'c' som 99 og 'a' som 97, så 'c' - 'a' er 2.

Tabellen `codes` omsætter 0 til 'n', 1 til 'o', 2 til 'p', osv.

Programmering 1999

KVL

Side 6-11

Opbygning af histogram

```
import java.util.Random;

class Histogram {

    public static void main(String[] args) {
        int count = Integer.parseInt(args[0]);
        int[] freq = new int[1 + 6];

        for (int i=0; i<count; i=i+1) {
            int res = rolldie();
            freq[res] = freq[res] + 1;
        }

        for (int c=1; c<=6; c=c+1)
            System.out.println("Antal " + c + " ere: " + freq[c]);

        static Random rnd = new Random();

        static int rolldie() {
            return 1 + (int)(6 * rnd.nextDouble());
        }
    }
}
```

Programmering 1999

KVL

Side 6-10

Problem: gradvis udvidelse af en String er langsom

Eksempel: Udkørselse af

```
String s = "";
s = s + 'A';
s = s + 'B';
s = s + 'C';
...
s = s + 'Z';
```

skaber 26 midlertidige strenge A, AB, ABC, ..., ABC...Z.

Det endelige resultat har længde 26.

Der er kopieret $1 + 2 + 3 + \dots + 26 = \frac{26 \cdot 27}{2} = 351$ tegn for at danne resultatet.

Generelt: hvis resultatet har længde n , så er der blevet kopieret $\frac{n(n+1)}{2}$ tegn.

Eksempel: Når n er 300, er der kopieret ca. 45 000 tegn.

Med `StringBuffer` kan det meste af denne kopiering undgås.

Programmering 1999

KVL

Side 6-12

StringBuffer: effektivt udvidelige tegnstreng

En String forbliver for altid den samme, når den én gang er skabt.

Strengsammensætning (konkaterering) skaber en helt ny streng.

Derfor er gradvis udvidelse af en String meget langsom og ineffektiv.

En StringBuffer er en streng som kan udvides effektivt.

Konstruktorer i StringBuffer

```
new StringBuffer()
```

```
new StringBuffer(117)
```

Det er mest effektivt at lave StringBufferen stor nok fra begyndelsen.

Metoder i StringBuffer

```
sb.append(c)
```

```
sb.length()
```

```
sb.charAt(i)
```

```
sb.setCharAt(i, c)
```

```
sb.toString()
```

Sammenfatning

Læs Lewis og Loftus kapitel 6, siderne 207–227 og 242–243.

- En tabel (et array) er en indekseret samling værdier af en given type.
 - Tabeller bruges til
 - at lagre foruddefineret information (Days)
 - at beregne et resultat gradvis (Histogram)
- Gradvis udvidelse af en String er tidskrævende.
- En StringBuffer kan udvides effektivt.

Effektiv version af primitiv kryptering

Metoden encrypt burde bruge StringBuffer i stedet, sådan her:

```
static String encrypt(String s) {
    StringBuffer res = new StringBuffer();
    for (int i=0; i<s.length(); i=i+1)
        res.append(codes[s.charAt(i) - 'a']);
    return res.toString();
}
```

I hvert gennemløb af for-løkken følges et tegn til res med kaldet res.append(...).

Stringbufferen res omdannes til en String inden metoden returnerer.