Opgaver tirsdag den 1. februar

Opgave 1

Lav en rekursiv metode der kan finde antallet af lige tal i en liste. Metoden skal have følgende signatur:

```
public static int ligeTal(ArrayList<Integer> list)
```

Lav både en version der ikke anvender hjælpemetoder og en der gør. Du kan tilføje metoden i den udleverede klasse HelperMethods, hvor du også kan finde eksempler på rekursive metoder med og uden hjælpemetoder.

Opgave 2

Et palindrom er en tekststreng der læses ens forfra og bagfra som f.eks ”ABBA” og ”radar”. Skriv en statisk rekursiv metode `public static boolean palindrom(String tekst)` der returnerer true hvis teksten er et palindrom og ellers falsk. Anvend hjælpe metoder, så der ikke skal laves substrings.

Opgave 3

Lav en metode der kan afgøre om et tal findes i et array af heltal (en søgning). Det kan antages at tallene i arrayet er sorteret i ikke aftagende orden og implementationen, skal være baseret på binær søgning. Implementationen skal anvende rekursion, idet den rekursive metode, skal være en hjælpe metode, så der ikke skal laves kopier af dele af arrayet i de rekursive kald.

Opgave 4

Betragt nedenstående rekursive funktion (Ackermann funktionen)

```
A(x,y) = y+1                 når x = 0
         A(x-1,1)            når y = 0
         A(x-1, A(x,y-1))    ellers
```

Programmer metoden i Java. Tegn rekursionstræet for kaldet til metoden med `x = 1 og y = 3`. Hvad bliver resultatet ? (Det skal være 5)

Opgave 5

Skriv en rekursiv metode `public static int binomial(int n, int m)` der beregner binomialkoefficienten for positive heltal hvor `0<=m<=n.`

Den rekursive definition er givet ved

Termineringsregel: $K_{n,m} = 1,\ m = 0$

$K_{n,m} = 1,\ m = n$

Rekurrensregel: $K_{n,m} = K_{n-1,m} + K_{n-1,m-1},\ 0<m<n$

$$K_{n,m} = \binom{n}{m}$$

Test metoden med nedenstående tal:

```
Table of K(n,m)
  m 0       1       2       3       4       5       6       7
n -----------------------------------------------------------------
0 | 1
1 | 1       1
2 | 1       2       1
3 | 1       3       3       1
4 | 1       4       6       4       1
5 | 1       5       10      10      5       1
6 | 1       6       15      20      15      6       1
7 | 1       7       21      35      35      21      7       1
```

Opgave 6

Beregn antal flytninger der skal laves for at løse Towers of Hanoi for de følgende antal ringe: 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20 og 25. (Lav først beregningen i hånden, udvid dernæst programmet, så det tæller antal flytninger.) Hvor mange flytninger laves hvis problemet er af størrelse n ?

Opgave 7
Betragt nedenstående rekursive definition af en talfølge

tal_0	=	2	
tal_1	=	1	
tal_2	=	5	
tal_n	=	$2*tal_{n-3} - tal_{n-1}$	når n>2 og n er lige
tal_n	=	$tal_{n-1} + tal_{n-2} + 3*tal_{n-3}$	når n>2 og n er ulige

Opgave 7.1
Find værdien af det tredje, fjerde, femte og sjette tal i talfølgen; mere præcist ønskes værdien af tal_3, tal_4, tal_5 og tal_6 beregnet.

Opgave 7.2
Lav en **rekursiv** metode, *talN*, der beregner det n'te tal i ovenstående talfølge. Mere præcist ønskes nedenstående metode realiseret

public int talN (int n)
/**
* ***Krav:*** $n \geq 0$
* ***Post:*** *talN' = n'te tal i talfølgen, givet ved ovenstående definition af* tal_n}
*/

Opgave 7.3 (Lidt drilsk)

Lav en **iterativ** metode der beregner det n'te tal i ovenstående talfølge. Metoden har den samme specifikation som i opgave 7.2, men metoden må ikke være rekursiv.

Opgave 8* Gennemlæsning af mappe på disken

Følgende metoder gennemløber en mappestruktur rekursivt

```
public static void ScanDir(String path) {
  System.out.println("[DIR]  "+path);

  // skab et File-objekt svarende til mappen path
     File file=new File(path);

     // få listen over alle filer og undermapper
  String[] names=file.list();
  for (String name: names) {
    File file2=new File(path+"/"+name);
    if (file2.isDirectory())
      ScanDir(path+"/"+name);
  }
}
```

Et gennemløb startes f.eks. med kaldet

```
String path="c:/programmer";
ScanDir(path);
```

a) Læs metoden igennem og forstå princippet i løsningen.

NB: Scan ikke med start-mappen `"c:/"`, dels tager det meget lang tid og nogle af dem er beskyttede af styresystemet.

b) Lav en ny rekursiv løsning der gennemløber en mappe og tæller antal directories i stedet for at udskrive dem, Signaturen for metoden skal være

```
public static int ScanDirCount(String path)
```

Kaldet skal f.eks. være

```
int n=ScanDirCount("c:/programmer");
System.out.println("Antal mapper: "+n);
```

c) Udskriften i a) er ikke overskuelig, den kan se ud som

```
[DIR]  c:/programmer
[DIR]  c:/programmer/Adobe
[DIR]  c:/programmer/Adobe/Acrobat 6.0
[DIR]  c:/programmer/Adobe/Acrobat 6.0/Esl
[DIR]  c:/programmer/Adobe/Acrobat 6.0/Help
[DIR]  c:/programmer/Adobe/Acrobat 6.0/Help/DAN
[DIR]  c:/programmer/Adobe/Acrobat 6.0/Help/ENU
[DIR]  c:/programmer/Adobe/Acrobat 6.0/Reader
[DIR]  c:/programmer/Adobe/Acrobat 6.0/Reader/ActiveX
[DIR]  c:/programmer/Adobe/Acrobat 6.0/Reader/Browser
[DIR]  c:/programmer/Adobe/Acrobat 6.0/Reader/HowTo
[DIR]  c:/programmer/Adobe/Acrobat 6.0/Reader/HowTo/DAN
```

Udvid signaturen for ScanDir, så du udskrive niveauet i strukturen foran mappestien:

```
1: [DIR]  c:/programmer
2: [DIR]  c:/programmer/Adobe
3: [DIR]  c:/programmer/Adobe/Acrobat 6.0
4: [DIR]  c:/programmer/Adobe/Acrobat 6.0/Esl
4: [DIR]  c:/programmer/Adobe/Acrobat 6.0/Help
5: [DIR]  c:/programmer/Adobe/Acrobat 6.0/Help/DAN
5: [DIR]  c:/programmer/Adobe/Acrobat 6.0/Help/ENU
4: [DIR]  c:/programmer/Adobe/Acrobat 6.0/Reader
5: [DIR]  c:/programmer/Adobe/Acrobat 6.0/Reader/ActiveX
5: [DIR]  c:/programmer/Adobe/Acrobat 6.0/Reader/Browser
5: [DIR]  c:/programmer/Adobe/Acrobat 6.0/Reader/HowTo
6: [DIR]  c:/programmer/Adobe/Acrobat 6.0/Reader/HowTo/DAN
```