

### Opgave 1.1

Der udskrives:

```
sum: 9 j: 9 k: -2
sum: 20 j: 11 k: 0
sum: 31 j: 11 k: 2
sum: 40 j: 9 k: 4
sum: 45 j: 5 k: 6
```

### Opgave 1.2

Der udskrives:

```
sum: 10 i: 6 j: 6
sum: 11 i: 6 j: 7
sum: 11 i: 4 j: 4
sum: 12 i: 4 j: 5
sum: 14 i: 4 j: 6
sum: 17 i: 4 j: 7
endelig sum: 17
```

### Opgave 1.3

```
public static void spiral(int n){
    Tegnemaskine tm = new Tegnemaskine();
    while (n > 2) {
        tm.move(n);
        tm.turn(90);
        n = n -2;
    }
}
```

### Opgave 1.4

```
public static void kasse(int n, int k, int s){
    Tegnemaskine tm = new Tegnemaskine();
    while (n>0){
        for (int side = 1; side <= 4; side++){
            tm.move(s);
            tm.turn(90);
        }
        s = s - 2*k;
        tm.jump(k); tm.turn(90); tm.jump(k); tm.turn(-90);
        n--;
    }
}
```

### Opgave 2.

Hele denne opgave går ud på at udfylde forskellige aspekter af klassen vandløb. Her er klassen vandløb i fuld udgave.

```
public class Vandløb
{
    private String navn;
    private double bredde; // målt i meter
    private double længde; // målt i kilometer
    private Udspring udspring;

    private Vandløb[] sideGrene = new Vandløb[20]; // maksimalt 10 sidegrene
    private int antalSideGrene = 0;

    public Vandløb(String n, double b, double l, Udspring u){
```

```

        navn = n; bredde = b; længde = l; udspring = u;
    }

    public int getAntalSideGrene(){ return antalSideGrene; }
    public int getNavn(){ return navn; }

    public void tilføjSidegren(Vandløb v){
        sideGrene[antalSideGrene] = v;
        antalSideGrene ++;
    }

    public double systemLængde(){
        double sum = længde;
        for (int sideGren = 0; sideGren < antalSideGrene; sideGren++)
            sum += sideGrene[sideGren].systemLængde();
        return sum;
    }

    public double vandMængde(){
        double sum = udspring.getLiterPrTime();
        for (int sideGren = 0; sideGren < antalSideGrene; sideGren++)
            sum += sideGrene[sideGren].vandMængde();
        return sum;
    }
}

```

Delopgave 2.3 handler om at lave en række erklæringer der repræsenterer Suså.

```

Vandløb ringstedÅ = new Vandløb("Ringsted å", 7, 13, new Udspring("Gyrstinge sø",
Vandløb vigerdalsÅ = new Vandløb("VigerdalsÅ", 3, 6, new Udspring(" Valsøllille sø"
ringstedÅ.tilføjSidegren( vigerdalsÅ);
Vandløb kongskildebæk = new Vandløb("Kongskildebæk", 1,4, new Udspring("Kongskilde
Vandløb sorøå = new Vandløb("Sorø å ", 2, 7, new Udspring("Sorø sø", 210) );
Vandløb suså = new Vandløb("Suså", 8, 50, new Udspring("Gøgsmose", 175) );
suså.tilføjSidegren(kongskildebæk);
suså.tilføjSidegren(sorøå);
suså.tilføjSidegren(ringstedå);

```

Delopgave 2.4.

a) Der kaldes kun videre på vandløb med side grene. Men ved alle vandløb vil man ende med så små vandløb at de ikke har sidegrene. Altså vil man altid komme til et vandløb som ikke har sidegrene, og dermed ikke kalde vandmængde inde fra forløkken - idet vi uden sidegrene ikke kommer ind i for løkken.

b) Der vil ske et kald for suså, og et pr sidegren, og for hver af sidegrenenes sidegrene osv. Suså systemet består i denne opgave af: **Suså, Kongskilde Bæk, Sorø Å, Ringsted Å og Vigerdalså**. Altså i alt 5 vandløb, og dermed i alt 5 kald.

Opgave 3.

Denne opgave handler om klassen Boligblok. Den bliver som:

```

public class BoligBlok
{
    Lejlighed[][] lejligheder;
    BoligBlok(int opgange, int etager){
        lejligheder = new Lejlighed[opgange][etager];
        for (int opgang = 1; opgang <= opgange; opgang++)
            for (int etage = 1; etage <= etager; etage++)
                lejligheder[opgang-1][etage-1] = new Lejlighed();
    }
    public Lejlighed getLejlighed(int opgang, int etage){
        return lejligheder[opgang -1] [etage -1];
    }
}

```

```

public void flytInd(Person[] personer, String tlf, int opgang, int etage){
    getLejlighed(opgang,etage).flytInd(personer, tlf);
}

public int antalPersoner(){
    int antal = 0;
    for (int opgang = 1; opgang < lejligheder.length; opgang++)
        for (int etage = 1; opgang < lejligheder[0].length; etage ++ )
            antal += getLejlighed(opgang,etage).beboere.length;
    return antal;
}

public Lejlighed flest(){
    Lejlighed medFlest = getLejlighed(1,1);
    for (int opgang = 1; opgang < lejligheder.length; opgang++)
        for (int etage = 1; opgang < lejligheder[0].length; etage ++ )
            if ( getLejlighed(opgang,etage).beboere.length > medF
                medFlest = getLejlighed(opgang,etage);
    return medFlest;
}

public int antalBørn(){
    int antal = 0;
    for (int opgang = 1; opgang < lejligheder.length; opgang++)
        for (int etage = 1; opgang < lejligheder[0].length; etage ++ )
            antal += getLejlighed(opgang,etage).antalBørn();
    return antal;
}
}

```

I delopgave 3.5 skal der også udfyldes en metode i klassen Lejlighed. Den bliver som følger:

```

public int antalBørn(){
    int antal = 0;
    for (int person = 0; person<beboere.length; person++)
        if (beboere[person].alder <= 12)
            antal ++;
    return antal;
}

```

#### Opgave 4.1

Der skal prøves om følgende situationer kan håndteres.

1. En blok med 1 lejlighed.
2. En blok med tomme lejligheder.
3. En blok hvor den med flest er i hver hjørne (4 tilfælde i alt).
4. En blok hvor alle lejligheder har det samme antal beboere.

#### Opgave 4.2

```

void tegnBoligblok(Boligblok blok, Graphics gr){
    final int SIZE = 25;
    for (int i = 1; i<=blok.lejligheder.length; i++)
        for (int j=1; j<= blok.lejligheder[0].length; j++) {
            switch ( blok.getLejlighed(i,j).beboere.length ) {
                case 0: gr.setColor(Color.green); break;
                case 1: case 2: gr.setColor(Color.blue); break;
                default: gr.setColor(Color.red); break;
            }
            gr.fillRect( (i-1)*SIZE, 150-(j*SIZE), SIZE-1, SIZE-1 );
        }
}
}

```

### Opgave 4.3

De to komponenter kan erklæres som:

```
SLabel tlfLabel = new SLabel("Telefon nummer: ");
STextField telefonFelt = new STextField("dummy", 9);
```

Metoden kan skrives som:

```
public void lejlighed_clicked(BoligBlok blok, int opgang, int etage){
    if (blok.getLejlighed(opgang, etage).telefonNr != null )
        telefonFelt.setText( blok.getLejlighed(opgang, etage).telefonNr );
    else
        telefonFelt.setText( "-----" );
}
```