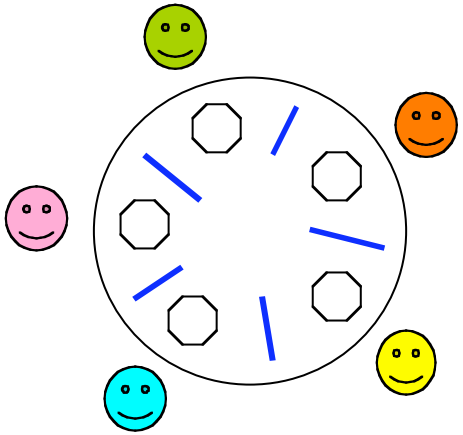
Ugeseddel 8

25. oktober - 1. november

- Læs kapitel 12 i lærebogen (side 587 - 643).
- Løs opgave 11.3.
- Løs opgaven på de næste sider.

Ekstraopgave 6

Fem filosoffer sidder omkring et rundt bord i dybe tanker. Udover at tænke må de en gang imellem spise. Foran hver af filosofferne er en skål med ris. Før en filosof kan spise, må han have to kinesiske spisepinde. Han tager én pind ad gangen – en fra venstre, eller en fra højre.



Filosofferne må finde en måde at deles om pindene, således at de alle får noget at spise.

På de følgende sider er vist et udkast til et program, der simulerer forløbet. Programmet stopper, når alle filosoffer har indtaget 100 måltider.

(a) Programmér klassen `Chopstick`.

I denne udgave af programmet kan der opstå baglås (deadlock). Det sker, hvis alle filosoffer tager den venstre spisepind samtidigt. Så bliver de fastlåst i deres forgæves forsøg på at få en højre pind.

Problemet kan løses på flere måder. En løsning er at lade ulige nummererede filosoffer tage den venstre spisepind først, og lade lige nummererede filosoffer tage den højre pind først.

(b) Programmér denne løsning.

En anden løsning er højst at give 4 af filosofferne adgang til bordet samtidigt.

(c) Programmér denne løsning.

En tredje løsning er kun at give en filosof lov til at tage pinde op, hvis begge pinde er fri. Begge pinde tages da op som en atomisk operation.

(d) Programmér denne løsning.

```
public class DiningPhilosophers {
    public static void main(String args[]) {
        ChopStick[] chopStick = new ChopStick[N];
        for (int i = 0; i < N; i++)
            chopStick[i] = new ChopStick();
        for (int i = 0; i < N; i++)
            new Philosopher(i, chopStick[(i - 1 + N) % N],
                            chopStick[i]).start();
    }

    static final int N = 5;
}
```

```
class Philosopher extends Thread {
    public Philosopher(int id, ChopStick left, ChopStick right) {
       this.id = id;
       leftChopStick = left;
       rightChopStick = right;
    }

    public void run() {
        for (int meals = 0; meals < 100; meals++) {
            think();
            leftChopStick.grab();
            rightChopStick.grab();
            eat();
            leftChopStick.release();
            rightChopStick.release();
        }
        System.out.println("Philosopher #" + id + " leaves the room");
    }

    ...
```

fortsættes

```
    void think() {
        System.out.println("Philosopher #" + id + " is thinking");
        try {
            sleep((long) (Math.random() * 10));
        } catch (InterruptedException e) {}
        System.out.println("Philosopher #" + id + " is hungry");
    }

    void eat() {
        System.out.println("Philosopher #" + id + " starts eating");
        try {
            sleep((long) (Math.random() * 20));
        } catch (InterruptedException e) {}
        System.out.println("Philosopher #" + id + " is stuffed");
    }

    int id;
    ChopStick leftChopStick, rightChopStick;
}

class ChopStick { /* spørgsmål (a) */ }
```