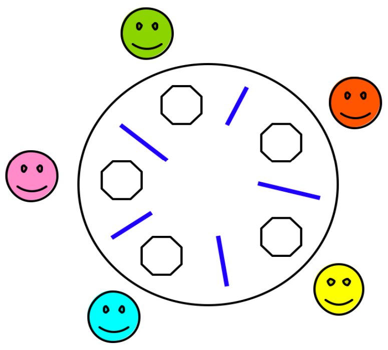
Exercise 1 (Dining philosophers)

Five philosophers are sitting around at a round table in deep thoughts. But of course, from time to time they must have something to eat. In front of each philosopher is a bowl of rice. Between each pair of philosophers is one chopstick. Before a philosopher can eat he must have two chopsticks, one taken from the left, and one taken from the right.



The philosophers must find some way to share chopsticks such that they all get to eat.

On the following page is shown an outline of a program, which simulates the system. The program terminates when each philosopher have eaten 100 meals.

(a) Program the Chopstick class.

This version of the program may result in deadlock. This happens if all philosophers take the left chopstick simultaneously. In this case they are all blocked in their attempt to take the right chopstick.

This problem may be solved in several ways. One solution is let odd-numbered philosophers take the left chopstick before the right chopstick, and let the even-numbered philosophers take the right chopstick before the left chopstick.

(b) Program this solution.

Another solution is to allow at most 4 philosophers to enter the table simultaneously.

(c) Program this solution.

A third solution is to allow a philosopher to take chopsticks only if both chopsticks are available. The two chopsticks are taken as an atomic operation.

(d) Program this solution.

```
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");
    }

    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 { /* Question (a) */ }
```