Ekstraopgave 6

(a)

```
class ChopStick {
    boolean taken;

    synchronized void grab() {
        try {
            while (taken)
                wait();
        } catch (InterruptedException e) {}
        taken = true;
    }

    synchronized void release() {
        taken = false;
        notify();
    }
}
```

(b)

```
public void run() {
    for (int meals = 0; meals < 100; meals++) {
        think();
        if (id % 2 == 0) {
            leftChopStick.grab();
            rightChopStick.grab();
        } else {
            rightChopStick.grab();
            leftChopStick.grab();
        }
        eat();
        leftChopStick.release();
        rightChopStick.release();
    }
    System.out.println("Philosopher #" + id + " leaves the room");
}
```

(c)

```
class Table {
    int count = 0;

    synchronized void enter() {
        try {
            while (count == DiningPhilosophers2.N - 1)
                wait();
        } catch (InterruptedException e) {}
        count++;
    }

    synchronized void leave() {
        count--;
        notify();
    }
}

public class DiningPhilosophers2 extends DiningPhilosophers {
    static Table table = new Table();
    ...
}

class Philosopher extends Thread {
    ...

    public void run() {
        for (int meals = 0; meals < 100; meals++) {
            think();
            DiningPhilosophers2.table.enter();
            leftChopStick.grab();
            rightChopStick.grab();
            eat();
            leftChopStick.release();
            rightChopStick.release();
            DiningPhilosophers2.table.leave();
        }
        System.out.println("Philosopher #" + id + " leaves the room");
    }

    ...
}
```

(c) Alternativ løsning med statiske metoder i class Table

```
class Table {
    static int count = 0;

    static synchronized void enter() {
        while (count == DiningPhilosophers.N - 1) {
            try {
                Table.class.wait();
            } catch (InterruptedException e) {}
        }
        count++;
    }

    static synchronized void leave() {
        count--;
        Table.class.notifyAll();
    }
}

class Philosopher extends Thread {
    ...

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

    ...
}
```

(d)

```
public void run() {
    for (int meals = 0; meals < 100; meals++) {
        think();
        grabChopSticks(this);
        eat();
        releaseChopSticks(this);
    }
    System.out.println("Philosopher #" + id + " leaves the room");
}

static synchronized void grabChopSticks(Philosopher phil) {
    while (phil.leftChopStick.taken || phil.rightChopStick.taken) {
        try {
            Philosopher.class.wait();
        } catch (InterruptedException e) {}
    }
    phil.leftChopStick.taken = true;
    phil.rightChopStick.taken = true;
}

static synchronized void releaseChopSticks(Philosopher phil) {
    phil.leftChopStick.taken = false;
    phil.rightChopStick.taken = false;
    Philosopher.class.notifyAll();
}
```

Bemærk: `grabChopSticks` er en statisk metode i klassen `Philosopher`. Den anvendte lås er den, der hører til klassen.