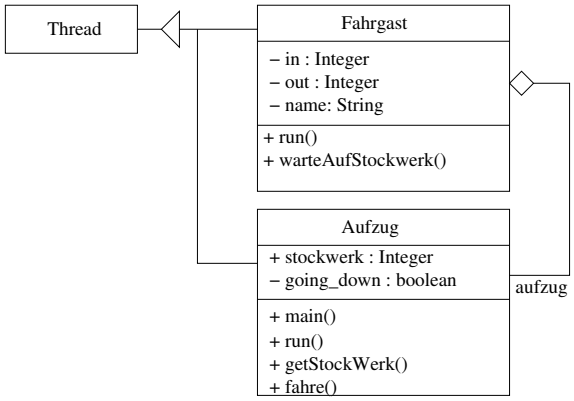
Technische Universität München
Fakultät für Informatik
Prof. Dr. A. Knoll

SS 2006
Übungsblatt 6
14. Juli 2006

Übungen zu Einführung in die Informatik II

Aufgabe 19 Pater-Noster

a) Eine mögliche Modellierung sieht folgendermaßen aus:



b)
```
public class Aufzug extends Thread {
    private int stockwerk = -1;
    private boolean going_down;

    public Aufzug() {
        start();
    }

    public void run() {
        while (true) {
            try {
                sleep(1000);
            } catch (InterruptedException e) {
            }
            fahre();
        }
    }

    public synchronized int getStockwerk() {
        return stockwerk;
    }

    private synchronized void fahre() {
        if (going_down) {
            stockwerk--;
            if (stockwerk == 0)
```

```
                going_down = false;
            } else {
                stockwerk++;
                if (stockwerk == 10)
                    going_down = true;
            }
            System.out.println("Stockwerk " + stockwerk);
            notifyAll();
        }

        public static void main(String[] args) {
            Aufzug a = new Aufzug();
            Fahrgast f1 = new Fahrgast("Thomas", a, 3, 5);
            Fahrgast f2 = new Fahrgast("Martin", a, 8, 3);
            Fahrgast f3 = new Fahrgast("Edgar", a, 10, 1);
        }
    }

    public class Fahrgast extends Thread {
        int in, out;
        Aufzug aufzug;
        String name;

        public Fahrgast(String name, Aufzug aufzug, int in, int out) {
            this.name = name;
            this.aufzug = aufzug;
            this.in = in;
            this.out = out;
            start();
        }

        public void run() {
            System.out.println(name + " wartet im " + in + ". Stock");
            warteAufStockwerk(in);
            System.out.println(name + " betritt Aufzug im " + in + ". Stock");
            warteAufStockwerk(out);
            System.out.println(name + " verlässt Aufzug im " + out + ". Stock");
        }

        synchronized void warteAufStockwerk(int stockwerk) {
            synchronized (aufzug) {
                while (stockwerk != aufzug.getStockwerk())
                    try {
                       aufzug.wait();
                    } catch (InterruptedException e) {
                    }
            }
        }
    }
```

Aufgabe 20 Speisende Philosophen (Lösungsvorschlag)

a) DiningPhilosophers.java

```
public class DiningPhilosophers {
  public static void main(String[] args) {
    int count = 5;

    Fork[] forks = new Fork[count];

    for (int i = 0; i < count; i++) {
      forks[i] = new Fork(i);
    }

    Philosopher[] philosophers = new Philosopher[count];

    for (int i = 0; i < count - 1; i++) {
      philosophers[i] = new Philosopher(i, forks[i], forks[i
          + 1]);
    }

    philosophers[count - 1] = new Philosopher(count - 1,
       forks[count - 1], forks[0]);

    for (int i = 0; i < count; i++) {
      philosophers[i].start();
    }
  }
}
```

Fork.java

```
public class Fork {
  public int id;
  private boolean taken;

  Fork(int id) {
    this.id = id;
    this.taken = false;
  }

  public synchronized void put(int philosopher) {
    this.taken = false;
    System.out.println("Fork␣" + this.id + "␣has␣been␣put␣
       down␣by␣philosopher␣" + philosopher);
    this.notifyAll();
  }

  public synchronized void take(int philosopher) {
    while (this.taken) {
      try {
        wait();
      } catch (InterruptedException e) {
        break;
      }
```

```
    }

    try {
      Thread.sleep((int) (Math.random() * 2000));
    } catch (InterruptedException e) {
    }

    this.taken = true;
    System.out.println("Fork " + this.id + " has been taken
       by philosopher " + philosopher);
    this.notifyAll();
  }
}
```

Philosopher.java

```
public class Philosopher extends Thread {
  private Fork f0;
  private Fork f1;
  private int id;

  Philosopher(int id, Fork f0, Fork f1) {
    this.f0 = f0;
    this.f1 = f1;
    this.id = id;
  }

  public void eat() {
    this.f0.take(this.id);
    this.f1.take(this.id);

    System.out.println("Philosopher " + this.id + " is
       eating with forks " + this.f0.id + " and " + this.f1.
       id + "...");

    try {
      this.sleep((int) (Math.random() * 100));
    } catch (InterruptedException e) {
    }

    this.f0.put(this.id);
    this.f1.put(this.id);
  }

  public void run() {
    while (true) {
      this.think();
      this.eat();
    }
  }

  public void think() {
```

```
      System.out.println("Philosopher " + this.id + " is
         thinking...");

      try {
        this.sleep((int) (Math.random() * 100));
      } catch (InterruptedException e) {
      }
    }
  }
```

b) DiningPhilosophers.java

```
import java.util.concurrent.Semaphore;

public class DiningPhilosophers {
  public static void main(String[] args) {
    int count = 5;

    Semaphore semaphore = new Semaphore(count - 1, true);

    Fork[] forks = new Fork[count];

    for (int i = 0; i < count; i++) {
      forks[i] = new Fork(i);
    }

    Philosopher[] philosophers = new Philosopher[count];

    for (int i = 0; i < count - 1; i++) {
      philosophers[i] = new Philosopher(i, forks[i], forks[i
          + 1], semaphore);
    }

    philosophers[count - 1] = new Philosopher(count - 1,
       forks[count - 1], forks[0], semaphore);

    for (int i = 0; i < count; i++) {
      philosophers[i].start();
    }
  }
}
```

Fork.java (s. vorherige Aufgabe)
Philosopher.java

```
import java.util.concurrent.Semaphore;

public class Philosopher extends Thread {
  private Fork f0;
  private Fork f1;
  private int id;
  private Semaphore semaphore;
```

```
  Philosopher(int id, Fork f0, Fork f1, Semaphore semaphore)
      {
    this.f0 = f0;
    this.f1 = f1;
    this.id = id;
    this.semaphore = semaphore;
  }

  public void eat() {
    this.f0.take(this.id);
    this.f1.take(this.id);

    System.out.println("Philosopher " + this.id + " is
       eating with forks " + this.f0.id + " and " + this.f1.
       id + "...");

    try {
      this.sleep((int) (Math.random() * 100));
    } catch (InterruptedException e) {
    }

    this.f0.put(this.id);
    this.f1.put(this.id);
  }

  public void run() {
    while (true) {
      this.think();

      try {
        this.semaphore.acquire();
      } catch (InterruptedException e) {
      }

      this.eat();
      this.semaphore.release();
    }
  }

  public void think() {
    System.out.println("Philosopher " + this.id + " is
       thinking...");

    try {
      this.sleep((int) (Math.random() * 100));
    } catch (InterruptedException e) {
    }
  }
}
```