TECHNISCHE UNIVERSITÄT MÜNCHEN
FAKULTÄT FÜR INFORMATIK

Lehrstuhl für Sprachen und Beschreibungsstrukturen
Praktikum Grundlagen der Programmierung
F. Forster, T. Gawlitza, A. Flexeder

WS 2007/2008
Lösungsvorschläge zu Blatt 12
18. Januar 2008

Übungen zu Praktikum Grundlagen der Programmierung

Aufgabe 54 Appletspielereien (Lösungsvorschlag)

a) HTML-Code: `simpleapplet.html`

```
<html>
  <head>
    <title>Applet-Spielereien</title>
  </head>
  <body>
    <h1>Applet-Spielereien</h1>
    <applet codebase="." code="SimpleApplet.class" width=600
            height=400 alt="Your␣browser␣has␣to␣be␣Java-enabled␣to␣see␣the␣applet!">
    </applet>
  </body>
</html>
```

b) Applet-Code: `SimpleApplet.java`

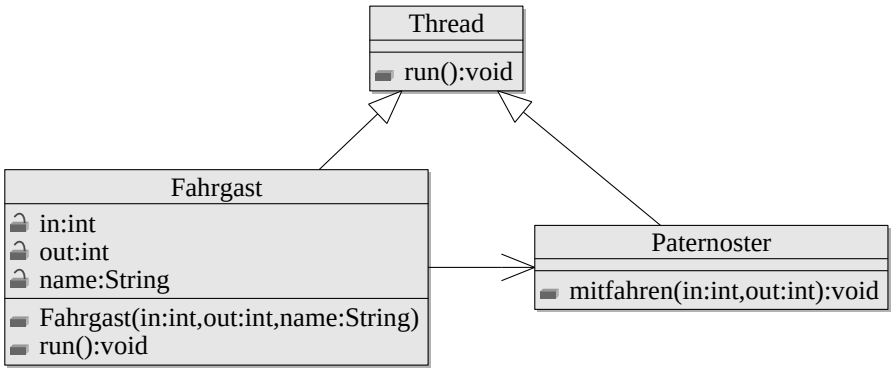
```
import java.applet.Applet;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Polygon;

public class SimpleApplet extends Applet {

    public void paint (Graphics page) {

        setBackground(Color.gray);
        Font big = new Font("SansSerif",Font.BOLD,30);
        Font normal = new Font("SansSerif",Font.BOLD,10);

        page.setFont(big);
        page.setColor(Color.black);
        page.drawString("Applet-Spielereien", 50, 50);
        page.setFont(normal);
        page.setColor(Color.white);
        page.drawString("im␣Praktikum␣Grundlagen␣der␣Programmierung", 50, 70);
        page.drawRoundRect(30,20, 340, 60,10,10);

        page.setColor(Color.blue);
        page.drawOval(200, 200, 100, 100);
        page.drawOval(400, 200, 100, 100);

        page.setColor(Color.black);
        page.fillRect(235, 235, 30, 30);
        page.fillRect(435, 235, 30, 30);

        Polygon p = new Polygon();
        p.addPoint(350, 240);
        p.addPoint(330, 280);
        p.addPoint(370, 280);

        page.fillPolygon(p);

        page.setColor(Color.red);
```

```
            page.drawLine(250, 350, 450, 350);
            page.drawLine(250, 350, 220, 320);
            page.drawLine(450, 350, 480, 320);

        }


    }
```

Test durch Aufrufen der HTML-Datei im WWW-Browser oder mit `appletviewer`.

Aufgabe 55 (Ü) Pater-Noster

Eine mögliche Modellierung sieht folgendermassen aus:




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

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

    public void run() {
        System.out.println("␣␣␣␣␣"+name + "␣wartet␣in␣[" + in + "]:␣nach␣[" + out + "]");
        aufzug.mitfahren(name, in, out);
        System.out.println("␣␣␣␣␣"+name + "␣verlaesst␣Aufzug␣in␣[" + out + "]");
    }
}
```

Einfache Version (keine Betrachtung der Platzkapazitäet):

```
public class Paternoster extends Thread {
    private int stockwerke;
    private int stockwerk = 0;
    private int passagiermaximum;
    private int passagierstand = 0;
    // Ein Monitor fuer Einstieg, einer fuer Ausstieg
    private Object aufzugRein;
    private Object aufzugRaus;

    public Paternoster (int stockwerke, int kapazitaet) {
        this.stockwerke = stockwerke;
        passagiermaximum = kapazitaet;

        // nur ein Monitor
        aufzugRein = new Object();
        aufzugRaus = new Object();
    }

    public void run() {
        boolean aufwaerts = true;
        while (true) {
```

```
            //fahren
            try {sleep(1000);} catch (InterruptedException e) { }
            if (aufwaerts) stockwerk ++;
            else stockwerk --;

            //umkehren
            if ((stockwerk ==0)||(stockwerk==stockwerke-1))
                aufwaerts = !aufwaerts;

            System.out.println("----------------------------------------");
            System.out.println("Kabine in ["+stockwerk +"]:");

            // Leute aussteigen lassen
            System.out.println("* Lasse Leute AUSsteigen: ");
            synchronized(aufzugRaus) {aufzugRaus.notifyAll(); }
            try {sleep(500);} catch (InterruptedException e) {}

            //Leute einsteigen lassen
            System.out.println("* Lasse Leute EINsteigen: ");
            synchronized(aufzugRein) {aufzugRein.notifyAll(); }
            try {sleep(500);} catch (InterruptedException e) {}

            //weiterfahren
        }
    }

    // Zugriff auf Plaetze in der Kabine ueber einen Monitor
    private Object platzmonitor = new Object();

    private boolean nehmePlatz(){
        synchronized (platzmonitor){
            if (passagierstand < passagiermaximum) {
                passagierstand++;
                return true;
            }
            else return false;
        }
    }

    private void gebePlatzFrei(){
        synchronized (platzmonitor){
            passagierstand --;
        }
    }

    public void mitfahren(String name, int einstieg, int ziel){
        // anstellen solange noch kein Platz in der Kabine ist, und solange ein Platz frei ist
        while ((stockwerk!=einstieg)||(!nehmePlatz()))
            synchronized (aufzugRein){
                try {aufzugRein.wait();} catch(InterruptedException ie){}
            }
        System.out.println("     "+name + " betritt Aufzug   in [" + stockwerk + "]");

        // warten auf das Erreichen des Zielgeschoss
        while (stockwerk!=ziel)
            synchronized (aufzugRaus){
                try {aufzugRaus.wait();} catch(InterruptedException ie){}
            }
        gebePlatzFrei();
    }
}
```

Komplexere Version:

```
public class Paternoster2 extends Thread {
    private int stockwerke;
    private int stockwerk = 0;
    private int passagiermaximum;
    private int passagierstand = 0;
    private Object[][] etagen;

    public Paternoster2 (int stockwerke, int kapazitaet) {
        this.stockwerke = stockwerke;
        passagiermaximum = kapazitaet;
        etagen = new Object[stockwerke][2];

        for (int i=0; i<etagen.length; i++)
            for (int j=0; j<2; j++)
                etagen[i][j]=new Object();
    }
```

```
    public void run() {
        boolean aufwaerts = true;
        while (true) {

            //fahren
            try {sleep(1000);} catch (InterruptedException e) { }
            if (aufwaerts) stockwerk ++;
            else stockwerk --;

            //umkehren
            if ((stockwerk ==0)||(stockwerk==stockwerke-1))
                aufwaerts = !aufwaerts;

            System.out.println("----------------------------------------");
            System.out.println("Kabine in ["+stockwerk +"]:");

            // Leute aussteigen lassen
            System.out.println("* Lasse Leute AUSsteigen: ");
            synchronized(etagen[stockwerk][0]) {etagen[stockwerk][0].notifyAll(); }
            try {sleep(500);} catch (InterruptedException e) {}

            //Leute einsteigen lassen
            System.out.println("* Lasse Leute EINsteigen: ");
            synchronized(etagen[stockwerk][1]) {etagen[stockwerk][1].notifyAll(); }
            try {sleep(500);} catch (InterruptedException e) {}

            //weiterfahren
        }
    }

    // Zugriff auf Plaetze in der Kabine ueber einen Monitor
    private Object platzmonitor = new Object();

    private boolean nehmePlatz(){
        synchronized (platzmonitor){
            if (passagierstand < passagiermaximum) {
                passagierstand++;
                return true;
            }
            else return false;
        }
    }

    private void gebePlatzFrei(){
        synchronized (platzmonitor){
            passagierstand --;
        }
    }

    public void mitfahren(String name, int einstieg, int ziel){
        // anstellen solange noch kein Platz in der Kabine ist, und solange ein Platz frei ist
        while ((stockwerk!=einstieg)||(!nehmePlatz()))
            synchronized (etagen[einstieg][1]){
                try {etagen[einstieg][1].wait();} catch(InterruptedException ie){}
            }
        System.out.println("     "+name + " betritt Aufzug   in [" + stockwerk + "]");

        // warten auf das Erreichen des Zielgeschoss
        synchronized (etagen[ziel][0]){
            try {etagen[ziel][0].wait();} catch(InterruptedException ie){}
        }
        gebePlatzFrei();
    }

    public static void main(String[] args){
        Paternoster p = new Paternoster(6, 2);
        Fahrgast f1 = new Fahrgast("Thomas ", p, 3, 5);
        Fahrgast f2 = new Fahrgast("Alex   ", p, 2, 3);
        Fahrgast f3 = new Fahrgast("Peter  ", p, 2, 1);
        Fahrgast f4 = new Fahrgast("Sylvia ", p, 2, 1);
        Fahrgast f5 = new Fahrgast("Andrea ", p, 2, 1);
        Fahrgast f6 = new Fahrgast("Michael", p, 2, 1);
        f1.start();
        f2.start();
        f3.start();
        f4.start();
        f5.start();
        f6.start();
        p.start();
    }
}
```