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 11
22. Januar 2007

Übungen zu Praktikum Grundlagen der Programmierung

Aufgabe 49 (Ü) Threads

a-1)

```
public class AusgabeThread extends Thread{
    public void run(){
    for(int i=0; i<20; i++){
        try{//try-catch-Statement fuer die sleep()-Anweisung
        //Wartezeit zwischen den einzelnen Ausgaben
        sleep(2000); //Anzahl in Millisekunden
        }catch(InterruptedException ie){}//Unterbrechung des Threads
        System.out.println("Hello, I am thread"+Thread.currentThread().getName()+" in round "+i+".");
    }
    }
}
```

a-2)

```
public class AusgabeThread1 extends Thread{
    String inhalt;
    int runden;
    public AusgabeThread1(String s, int r){
    inhalt = s;
    runden = r;
    }

    public void run(){
    for(int i=0; i<runden; i++){
        try{
        sleep((new java.util.Random()).nextInt(3000));
        }catch(InterruptedException ie){}
        System.out.println(inhalt+" in round "+i);
    }
    }
}
```

alternativ:

```
public class Ausgabe2 implements Runnable{
    String inhalt;
    int runden;
    public Ausgabe2(String s, int r){
    inhalt = s;
    runden = r;
    }

    public void run(){
    for(int i=0; i<runden; i++){
        try{
        Thread.sleep((new java.util.Random()).nextInt(3000));
        }catch(InterruptedException ie){}
```

```
        System.out.println(inhalt+" in round "+i);
    }
    }
}

public class Ausgabe{
    static int counter = 0;

    public static void main(String[] args){
    AusgabeThread at = new AusgabeThread(); //Threadobjekt erzeugen
    at.start();//startet Ausfuehrung von Thread (setzt ihn auf rechenwillig)
    AusgabeThread1 t1 = new AusgabeThread1(new String("ich bin fred!"),2);
    t1.start();
    AusgabeThread1 t2 = new AusgabeThread1(new String("ich bin caroline!"),10);
    t2.start();
    AusgabeThread1 t3 = new AusgabeThread1(new String("ich bin mia!"),5);
    t3.start();
    }
}
```

b)

```
public class Ablauf{
    static Object monitor = new Object();
    static int counter = 0;

    public static void main(String[] args) {
    Ausgeber a = new Ausgeber();
    Increment t1 = new Increment();
    Increment t2 = new Increment();
    a.start();
    t1.start();
    t2.start();
    }
}

public class Ausgeber extends Thread {

    public void run() {
    while (true){
        synchronized(Ablauf.monitor){
        System.out.println("The number is: " +Ablauf.counter);
        }
        try{
        sleep(1000);
        }catch(InterruptedException ie){}

    }
    }
}

public class Increment extends Thread{

    public void run(){
    while (true){
        synchronized(Ablauf.monitor){
        System.out.println("in Thread "+Thread.currentThread().getName());
        //System.out.println("It is increment by two thread for "+Ablauf.counter);
        Ablauf.counter +=2;
        //System.out.println("-> "+Ablauf.counter);
        try{
            sleep( Math.abs((new java.util.Random()).nextInt(3000)));
        }catch(InterruptedException ie){}
        //System.out.println("It is decrement by one thread for "+Ablauf.counter);
        Ablauf.counter -=1;
        //System.out.println("-> "+Ablauf.counter);
        }
        try{
        sleep( Math.abs((new java.util.Random()).nextInt(1000)));
        }catch(InterruptedException ie){}
    }
    }
}
```

Aufgabe 50 (Ü) Siedler-Verklemmung

```
public class Siedler extends Thread{
    private Object a;
    private Object b;

    public Siedler(Object a, Object b){
    this.a=a;
    this.b=b;
    }

    public void run(){
    synchronized(a){
        try{
        System.out.println(this.getName()+": hat Ressource "+a);
        sleep(2000);
        }catch(InterruptedException ie){}
        System.out.println("Thread "+this.getName()+":moechte gerne Ressource "+b);

        synchronized(b){
        try{
            System.out.println(this.getName()+": hat Ressource "+b);
            sleep(2000);
        }catch(InterruptedException ie){}
        }
    }
    }

    public static void main(String[] arg){
    Object x = new String("Lehm");
    Object y = new String("Holz");
    Siedler hugo = new Siedler(x,y);
    hugo.start();
    Siedler egon = new Siedler(y,x);
    egon.start();
    }


}
```

Aufgabe 51 (Ü) Javadoc

- Klassen, Attribute, Konstruktoren und Methoden. Nicht: Blöcke.
- Aufbau eines Javadoc-Kommentars:

```
/** [Nur Kommentare, die mit /** beginnen, werden als Javadoc-Kommentar erkannt
 * [Textuelle Beschreibung der Klasse / des Attributs / der Methode.
 * Der erste Satz gilt als Überschrift, es sind aber mehrere Sätze möglich.]
 *
 * [Es folgen die Tags (siehe nächste Teilaufgabe]
 * @param
 * @see
 * ...
 *
 **/ [Ende des Kommentars]
```

- Erlaube Tags: (unbedingt notwendige Tags sind mit * gekennzeichnet)
  @param*, @return*, @throws / @exception, @author, @version, @see, @since, @serial, @deprecated
- Die Dokumentation für eine Klasse A generiert man mit

  ```
  javadoc A.java
  ```

  Den Einstieg in die Dokumentation gibt dann die neu erzeugte Datei index.html.