Prof. Dr. Stefan Brass
Dipl.-Inform. Annett Thüring
Dipl.-Inform. Steffen Schiele
Institut für Informatik
MLU Halle-Wittenberg

28. Mai 2013

Vorlesung "Objektorientierte Programmierung" — Nachholklausur —

Name: _____

Matrikelnummer: _____

Studiengang: _____

Aufgabe	Punkte	von	Zeit
1 (Programmierung: Klasse)		8	15 min
2 (Programmierung: Array)		10	15 min
3 (Arithmetische Ausdrücke, Bedingung)		4	5 min
4 (Globale/Lokale Var., Call by Value/Ref.)		6	10 min
5 (Rekursion)		2	5 min
6 (Vererbung)		3	5 min
7 (static)		6	10 min
8 (Fehlersuche)		3	5 min
Summe		42	70 min

- Ich fühle mich gesundheitlich in der Lage, diese Prüfung abzulegen. (Bitte sprechen Sie mit dem Aufsichtspersonal, falls Sie sich krank fühlen. Eine Krankmeldung muss vor Ende der Prüfung erfolgen.)
- Ich erkläre, dass ich diese Prüfung nicht bereits endgültig nicht bestanden habe.
- Falls ich nicht korrekt zu dieser Prüfung angemeldet sein sollte, erkläre ich mich damit einverstanden, dass ich rückwirkend angemeldet werde.

Unterschrift: _____

Hinweise:

- Bearbeitungsdauer: 90 Minuten
- Skript, Bücher, Notizen sind erlaubt. Notebooks, PDAs, etc. dürfen nicht verwendet werden. Mobiltelefone bitte ausschalten (oder mit Aufsicht besprechen).
- Die Klausur hat 12 Seiten. Bitte prüfen Sie die Vollständigkeit.
- Bitte benutzen Sie den vorgegebenen Platz. Wenn Sie auf die Rückseite ausweichen müssen, markieren Sie klar, dass es eine Fortsetzung gibt.
- Tauschen Sie keinesfalls irgendwelche Dinge mit den Nachbarn aus. Notfalls rufen Sie eine Aufsichtsperson zur Kontrolle.
- Versuchen Sie, alle Aufgaben zu bearbeiten (notfalls raten Sie). Wenn Sie nichts hinschreiben, haben Sie den Punkt für die jeweilige Teilaufgabe auf jeden Fall verloren.
- Fragen Sie, wenn Ihnen eine Aufgabe nicht klar ist!
- Schreiben Sie lesbar! Verwenden Sie keinen roten Stift. Bleistift ist nicht grundsätzlich verboten, aber problematisch, wenn Sie sich nach der Klausureinsicht beschweren wollen.
- Zum (garantierten) Bestehen benötigen Sie 60% der Punkte. Die Grenze wird möglicherweise gesenkt.

Zum Nachschlagen:

- Tabelle mit den Prioritätsstufen der Operatoren (von hoch nach niedrig):

1	`o.a`, `i++`, `a[]`, `f()`, ...	Postfix-Operatoren
2	`-x`, `!`, `~`, `++i`, ...	Präfix-Operatoren
3	`new C()`, `(type) x`	Objekt-Erzeugung, Cast
4	`*`, `/`, `%`	Multiplikation etc.
5	`+`, `-`	Addition, Subtraktion
6	`<<`, `>>`, `>>>`	Shift
7	`<`, `<=`, `>`, `>=`, `instanceof`	kleiner etc.
8	`==`,`!=`	gleich, verschieden
9	`&`	Bit-und, logisches und
10	`^`	Bit-xor, logisches xor
11	`|`	Bit-oder, logisches oder
12	`&&`	logisches und (bedingt)
13	`||`	logisches oder (bedingt)
14	`?:`	Bedingter Ausdruck
15	`=`, `+=`, `-=`, `*=`, `/=`, ...	Zuweisungen

  Bei gleicher Priorität sind alle Operatoren (außer Präfixoperatoren, Zuweisungen, bedingter Ausdruck) implizit von links geklammert (linksassoziativ).

- Bei der Integer-Division wird immer in Richtung 0 gerundet, für positive Eingabewerte also abgerundet, z.B. liefert `8/3` das Ergebnis `2`.

Aufgabe 1 (Programmierung: Klasse) 8 Punkte

Programmieren Sie eine Klasse `Notsignalgeber`. Feuerwehrleute tragen bei Einsätzen mit Atemschutzgeräten einen Notsignalgeber, der Bewegungen des Trägers registriert. Bleibt der eine gewisse Zeit (z.B. 30s) ohne Bewegung, wird ein Voralarm ausgelöst (z.B. 50 dB). Ist der Träger bei Bewustsein, wird er sich dann leicht bewegen, um den Alarm auszuschalten. Ansonsten wird z.B. nach weiteren 10s der Hauptalarm ausgelöst (100 dB). Dadurch erkennen die Kollegen die Notsituation und können ggf. auch den betroffenen Feuerwehrmann leichter finden (es gibt zusätzlich noch einen optischen Alarm, den wir in dieser Aufgabe zur Vereinfachung nicht beachten).

Die Klasse `Notsignalgeber` soll die wesentliche Steuerungslogik enthalten. Informationen über die Bewegung und die Zeit kommen von außen (s.u.), das brauchen Sie nicht zu programmieren. Objekte der Klasse `Notsignalgeber` sollen einen Konstruktur und zwei Methoden haben:

- Der Konstruktor hat zwei Parameter für die Zeit bis zum Voralarm und die zusätzliche Zeit bis zum Hauptalarm. Ein Aufruf könnte so aussehen:

  ```
  Notsignalgeber notsig = new Notsignalgeber(30, 10);
  ```

- Die Methode `tick()` (ohne Parameter) wird von einem externen Zeitgeber jede Sekunde einmal aufgerufen. Sie soll die Lautstärke des Alarms zurückliefern (als Rückgabewert der Methode), also `0` für kein Alarm, `50` für Voralarm und `100` für Hauptalarm. Ein möglicher Aufruf könnte so aussehen:

  ```
  int lautstaerke = notsig.tick();
  ```

  Beim oben erzeugten Objekt soll bei den ersten 30 Aufrufen der Wert `0` geliefert werden, beim 31. bis 40. Aufruf der Wert `50`, und ab dem 41. Aufruf der Wert `100` (sofern nicht zwischendurch eine Bewegung festgestellt wurde).

- Eine Methode `bewegt()`, die aufgerufen wird, wenn der Bewegungsmelder eine Bewegung erkannt hat. Diese Methode hat keine Parameter und keinen Rückgabewert. Ein möglicher Aufruf könnte so aussehen:

  ```
  notsig.bewegt();
  ```

  Nachdem diese Methode aufgerufen wurde, sollen die nächsten 30 Aufrufe von `tick()` wieder den Wert `0` liefern.

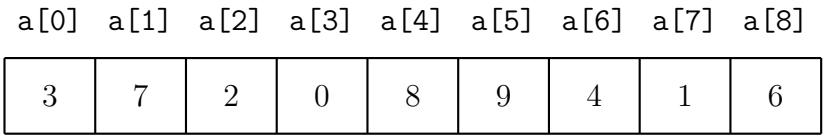
Der Konstruktor und die beiden Methoden sollen von außen aufrufbar sein, uns zwar auch von außerhalb des Paketes. Entsprechend muß die Klasse auch von außerhalb des Paketes aus benutztbar sein. Nichts sonst darf von außen zugreifbar sein (auch nicht von anderen Klassen des gleichen Paketes aus). Sie dürfen beliebige Attribute verwenden, die aber von von innerhalb dieser Klasse aus zugreifbar sein dürfen. Offenbar muß man sich die Parameter des Konstruktors abspeichern, und über die Zeit (Anzahl Aufrufe von `tick()`) seit dem letzten Aufruf von `bewegt()` Buch führen. Wie Sie das genau machen, ist aber Ihre Entscheidung.

Es ist Platz für die Lösung auf der nächsten Seite. Beachten Sie, dass auch für ein fehlendes oder falsches Semikolon ein halber Punkt abgezogen werden kann. Versuchen Sie also, Syntaxfehler zu vermeiden. Bemühen Sie sich außerdem um Verständlichkeit und guten Programmierstil. Es können auch für schlechten Stil Punkte abgezogen werden! Ebenso für unnötig komplizierte Lösungen.

Platz für die Lösung von Aufgabe 1 (Klasse `Vokabel`):

Aufgabe 2 (Programmierung: Array) 10 Punkte

Schreiben Sie eine Methode `fehlenderWert`, die in einem Array der Größe 9, das acht von den Zahlen 1–9 enthält, die fehlende Zahl an der Stelle ergänzt, an der die Zahl 0 steht. Dies wäre also eine extrem vereinfachte Version von Sudoku. Man könnte es eventuell als Unterprogramm in einem richtigen Sudoku nutzen: Wenn nur noch eine Ziffer in einer Zeile fehlt, soll diese ergänzt werden. Die freie Position ist hier durch die Zahl 0 markiert. Das Array könnte z.B. so aussehen:



Die Reihenfolge der Zahlen ist also beliebig. Ihre Methode müßte in diesem Beispiel an die Stelle `a[3]` den Wert `5` speichern.

```
public class Aufg2
{
    ... // Deklaration Ihrer Methode "fehlenderWert"

    public static void main(String[] args) {
        int[] a = new int[9];
        a[0] = 3;
        a[1] = 7;
        a[2] = 2;
        a[3] = 0;
        ...
        a[8] = 6;

        // Einfacher:
        // int[] a = {3, 7, 2, 0, 8, 9, 4, 1, 6 };

        fehlenderWert(a);
        System.out.println(a[3]);
    }
}
```

Ihre Methode soll keinen Rückgabewert haben. Sie können davon ausgehen, dass das übergebene Array `a` immer die Länge 9 hat, keine anderen als die Zahlen 0–9 enthält, und jede Zahl nur einmal. Die Zahl 0 kommt garantiert genau einmal vor. Sie dürfen das Array nur an der Stelle ändern, an der es den Wert 0 enthält (dies muss nicht `a[3]` sein, die Position ist beliebig).

Achten Sie darauf, dass der Aufruf so wie oben gezeigt funktioniert (aus der statischen Methode `main` heraus). Für unnötig umständliche Lösungen werden Punkte abgezogen (für schlechten Programmierstil auch).

Platz für die Lösung (Methode `fehlenderWert`):

Aufgabe 3 (Arithmetische Ausdrücke, Bedingung) 4 Punkte

Was gibt dieses Programm aus?

```
class Aufg3 {
    public static void main(String[] args) {

        int i = 5 + 3 * 4;
        System.out.println(i);          // a)

        int j = 13;
        j++;
        System.out.println(j % 3);      // b)

        int n = 27;
        int m = "abc".length();
        String s = "n";
        if(m < 3 || n == 27)
            s = "j";
        System.out.println(s);          // c)

        s = "1";
        System.out.println(s+2);        // d)
    }
}
```

a) _____

b) _____

c) _____

d) _____

Aufgabe 4 (Globale/lokale Var., Call by Value/Ref.) 6 Punkte

Was gibt dieses Programm aus?

```
public class Aufg4 {
    int n = 2;

    void f(int n) {
        this.n = 3;
        n = 5;
    }

    void g() {
        int n;
        n = 7;
    }

    static void h(Aufg4 o) {
        o.n = 11;
    }

    public static void main(String[] args) {
        Aufg4 x = new Aufg4();
        System.out.println(x.n);        // a)

        int n = 13;
        x.f(n);
        System.out.println(n);          // b)

        System.out.println(x.n);        // c)

        x.g();
        System.out.println(x.n);        // d)

        h(x);
        System.out.println(x.n);        // e)
        System.out.println(n);          // f)
    }
}
```

a) _____ d) _____

b) _____ e) _____

c) _____ f) _____

Aufgabe 5 (Rekursion) 2 Punkte

Was gibt dieses Programm aus?

```
public class Aufg5 {

    static String a = "";

    public static void f(int i)
    {
        System.out.println(i);
        if(i > 0) {
            a = a + "<";
            f(i-1);
            a = a + ">";
        }
        else {
            a = a + "+";
        }
    }

    public static void main(String[] args)
    {
        f(3);
        System.out.println(a);
    }
}
```

Ausgaben:

_____

_____

_____

_____

_____

Aufgabe 6 (Vererbung) 3 Punkte

Was gibt dieses Programm aus?

```
class O {
    int a = 1;
    void f() {
        System.out.println("O");
    }
    void g() {
        f();
    }
    void h() {
        System.out.println(a);
    }
}

class U extends O {
    int a = 2;
    void f() {
        System.out.println("U");
    }
}

public class Aufg6
{
    public static void main(String[] args)
    {
        O x = new U();

        x.f();  // a)
        x.g();  // b)
        x.h();  // c)
    }
}
```

a) _____ // x.f()

b) _____ // x.g()

c) _____ // x.h()

Aufgabe 7 (static) 6 Punkte

Gegeben sei folgendes Programm:

```
class Aufg7 {
    int i = 1;
    static int s = 2;

    int f() {
        int n = 1;
        n += i;   // a)
        n += s;   // b)
        n += g(); // c)
        return n;
    }
    static int g() {
        int n = 1;
        n += i;   // d)
        n += s;   // e)
        n += f(); // f)
        return n;
    }
}
```

Welche der Zugriff bzw. Aufrufe sind falsch? Kreuzen Sie die Konstrukte an, bei denen der Compiler einen Fehler melden wird. Begründen Sie Ihre Antwort in einen Satz, **auch bei den richtigen Konstrukten.** Es zählen nur Fehler, die der Compiler sicher findet, also z.B. keine Endlosrekursion.

a) ☐ `i` in `f()`
Begründung: _____

b) ☐ `s` in `f()`
Begründung: _____

c) ☐ `g()` in `f()`
Begründung: _____

d) ☐ `i` in `g()`
Begründung: _____

e) ☐ `s` in `g()`
Begründung: _____

f) ☐ `f()` in `g()`
Begründung: _____

Aufgabe 8 (Fehlersuche) 3 Punkte

Das folgende Programm enthält (mindestens) 4 Fehler. Bitte geben Sie drei dieser Fehler an (es ist möglicherweise schwierig, alle vier zu finden). Falls Sie mehr als drei Fehler angeben, werden nur die ersten drei gewertet (es gibt keine Extrapunkte). Geben Sie bitte jeweils die Zeilennummer mit an, in der sich der Fehler befindet. Es zählt auch nicht als Fehler, daß das Programm nichts Sinnvolles tut. Direkte Folgefehler zählen auch nicht.

```
 1 public class Aufg8 {
 2     public static void main(String[] args) {
 3         Aufg8 x = new C(5);
 4         int i;
 5         int n = i;
 6         C y = new C(3);
 7         y.f() = 2;
 8         System.out.println("eins" + 1);
 9         System.out.println(y.g());
10         if(y.f()) System.out.println("Hallo\n");
11     }
12 }
13
14 class C
15 {
16     private int a;
17     private int b;
18     public int f () { return a; }
19     public int g()  { return b; }
20     C(int n) { a = (n << 2); }
21 }
```

1. Fehler in Zeile: _____

   Begründung: _____

   _____

2. Fehler in Zeile: _____

   Begründung: _____

   _____

3. Fehler in Zeile: _____

   Begründung: _____

   _____