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

19. März 2013

Vorlesung "Objektorientierte Programmierung" — Klausur —

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 `Vokabel` für einen sehr einfachen Vokabeltrainer. Objekte der Klasse `Vokabel` sollen drei Attribute haben:

- `deutsch` für das deutsche Wort (eine Zeichenkette),
- `englisch` für die englische Übersetzung (auch eine Zeichenkette),
- `gewusst` für die Anzahl von Trainings-Sitzungen, bei denen die Vokabel gefragt wurde und korrekt beantwortet wurde (eine ganze Zahl).

Dabei ist Folgendes zu beachten:

- Die Attribute `deutsch` und `englisch` sollen von außen zugreifbar sein (auch von außerhalb des Paketes — berücksichtigen Sie das auch bei der Klassendeklaration).
- Diese beiden Attribute sollen nach der Initialisierung im Konstruktor nicht mehr änderbar sein.
- Das Attribut `gewusst` soll von außerhalb der Klasse nicht zugreifbar sein (auch nicht von Klassen des gleichen Paketes).

Ihre Klasse soll einen Konstruktor und zwei Methoden haben:

- Der Konstruktor soll Parameter für das deutsche und das englische Wort haben. Das Attribut "`gewusst`" soll im Konstruktor auf `0` gesetzt werden. Zum Beispiel soll es möglich sein, ein Objekt der Klasse so zu erzeugen:

  ```
  Vokabel v1 = new Vokabel("Katze", "cat");
  ```

- Eine Methode `anzGewusst`, die den aktuellen Wert des Attributes "`gewusst`" liefert. Sie hat natürlich keine Parameter. Ein möglicher Aufruf könnte so aussehen:

  ```
  int n = v1.anzGewusst();
  ```

- Eine Methode `getestet` (mit einem Wahrheitswert "`korrekt`" als Parameter): Falls diese Methode mit dem Parameterwert "`true`" aufgerufen wird, soll sie "`gewusst`" um eins erhöhen (der Benutzer hat die Frage nach der Vokabel korrekt beantwortet). Falls diese Methode mit dem Parameterwert "`false`" aufgerufen wurde, soll sie "`gewusst`" auf `0` setzen (der Benutzer hat die Vokabel nicht gewusst, und muss wieder ganz von vorne anfangen). Diese Methode hat keinen Rückgabewert. Ein möglicher Aufruf könnte so aussehen:

  ```
  v1.getestet(true);
  ```

Die zwei Attribute, der Konstruktor und die beiden Methoden sollen von außen (auch außerhalb des Paketes) zugreifbar sein. Nichts sonst darf von außen zugreifbar sein.

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

Ein bekannter Rätseltyp ist, gebene Zahlenfolgen fortzusetzen. Z.B. $5, 8, 11, \ldots$. Im Beispiel ist die Differenz zweier aufeinander folgender Glieder der Zahlenfolge immer gleich, nämlich 3. Daher kann man vermuten, dass die nächste Zahl 14 ist. Allgemein gilt für eine arithmetische Folge: $a_i = a_0 + i * d$. In dieser Aufgabe sollen Sie eine (statische) Methode `loese` schreiben, die solche Rätsel löst, falls sie vom Typ "arithmetische Folge" sind. Die Methode soll ein Array von `int`-Werten als Parameter bekommen. Falls das Array nicht mindestens zwei Elemente enthält (d.h. die Array-Länge ist kleiner als 2), soll der Text "`Was soll das denn?`" ausgegeben werden (und nichts weiter). Ansonsten soll sie aus den ersten beiden Array-Elementen die Differenz d bestimmen, und dann für alle übrigen Array-Elemente prüfen, ob sie wirklich nach obiger Formel berechnet werden können. Falls das für alle Array-Elemente der Fall ist, soll das nächste Element der Folge ausgegeben werden (was nach dem letzten Element des Arrays folgen würde). Falls mindestens ein Array-Element nicht obiger Formel entspricht, soll der Text "`Ich weiss nicht.`" ausgegeben werden. Dieser Text soll nur ein Mal ausgegeben werden, auch wenn mehrere Array-Elemente die Formel verletzen. Die Methode soll keinen Rückgabewert haben. Den Zugriffsschutz und andere Modifizierer wie `static` können Sie selbst wählen, aber das folgende Test-Programm muss funktionieren:

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

    public static void main(String[] args) {
        int[] a = new int[3];
        a[0] = 5;
        a[1] = 8;
        a[2] = 11;
        loese(a);
    }
}
```

Der Aufruf müsste die Zahl 14 ausgeben (mit `System.out.println(...)`).

Sie können davon ausgehen, dass das übergebene Array `a` immer vollständig initialisiert ist, d.h. dass an Position `a.length-1` die letzte zu berücksichtigende Zahl steht. Die Array-Größe kann beliebig sein, sie muss nicht `3` wie im Beispiel sein.

Platz für die Lösung (Methode `loese`):

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: _____

   _____