FernUniversität in Hagen

FAKULTÄT FÜR
MATHEMATIK UND INFORMATIK

Lehrgebiet Programmiersysteme
Prof. Dr. Friedrich Steimann

_____
(Name, Vorname)
_____
_____
(Straße, Nr.)
_____ _____
(PLZ) (Wohnort)
_____
(Land, falls außerhalb Deutschlands)

Kurs 1618 SS 2010

„Einführung in die objektorientierte Programmierung“

Nachklausur am 12.2.2011

Dauer: 3 Std., 10 – 13 Uhr

Lesen Sie zuerst die Hinweise auf der folgenden Seite!

Matrikelnummer: | | | | | | | | **Geburtsdatum:** | | | . | | . | | |

Klausurort: _____

Aufgabe	1	2	3	4	5	6	7	8	9	10	Summe
habe bear-beitet											
maximal	10	10	10	10	10	10	10	10	10	10	100
erreicht											
Korrektur											

O Herzlichen Glückwunsch, Sie haben die Klausur bestanden. Note: .....

O Sie haben die Klausur leider nicht bestanden. Für den nächsten Versuch wünschen wir Ihnen viel Erfolg. Die nächste Klausur findet im Sommersemester 2011 statt.

Hagen, den 23.2.2011

Im Auftrag

Hinweise zur Bearbeitung

1. Prüfen Sie die Vollständigkeit Ihrer Unterlagen. Die Klausur umfasst auf insgesamt 17 Seiten :
   – 1 Deckblatt
   – Diese Hinweise zur Bearbeitung
   – 10 Aufgaben auf Seite 3-15
   – Zwei zusätzliche Seiten 16 und 17 für weitere Lösungen

2. Füllen Sie jetzt bitte zuerst das Deckblatt aus:
   – Name, Vorname und Adresse,
   – Matrikelnummer, Geburtsdatum und Klausurort.

3. Schreiben Sie Ihre Lösungen mit Kugelschreiber oder Füllfederhalter (*kein Bleistift*) direkt in den bei den jeweiligen Aufgaben gegebenen, umrahmten Leerraum. Benutzen Sie auf keinem Fall die Rückseiten der Aufgabenblätter. Versuchen Sie, mit dem vorhandenen Platz auszukommen, sie dürfen auch stichwortartig antworten. Sollten Sie wider Erwarten nicht mit dem vorgegebenen Platz auskommen, benutzen Sie bitte die beiden an dieser Klausur anhängenden Leerseiten. **Es werden nur Aufgaben gewertet, die sich auf dem offiziellen Klausurpapier befinden.** Eigenes Papier ist nur für Ihre persönlichen Notizen erlaubt.

4. Kreuzen Sie die bearbeiteten Aufgaben auf dem Deckblatt an. Schreiben Sie unbedingt *auf jedes Blatt* Ihrer Klausur Ihren Namen und Ihre Matrikelnummer, auf die Zusatzblätter auch die Nummer der Aufgabe.

5. Geben Sie die gesamte Klausur ab. Lösen Sie die Blätter nicht voneinander.

6. Es sind *keine Hilfsmittel* zugelassen.

7. Lesen Sie vor der Bearbeitung einer Aufgabe den *gesamten* Aufgabentext sorgfältig durch.

8. Es sind maximal **100 Punkte** erreichbar. Wenn Sie mindestens **45 Punkte** erreichen, haben Sie die Klausur bestanden.

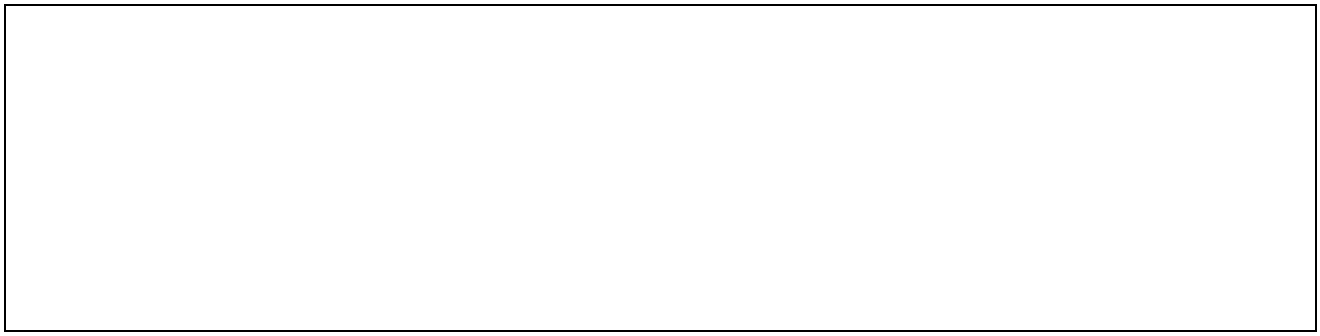
9. Sie erhalten die korrigierte Klausur zurück zusammen mit einer Bescheinigung für das Finanzamt und ggf. dem Übungsschein.

10. Legen Sie jetzt noch Ihren Studierendenausweis und einen amtlichen Lichtbildausweis bereit, dann kann die Arbeit beginnen. Viel Erfolg!

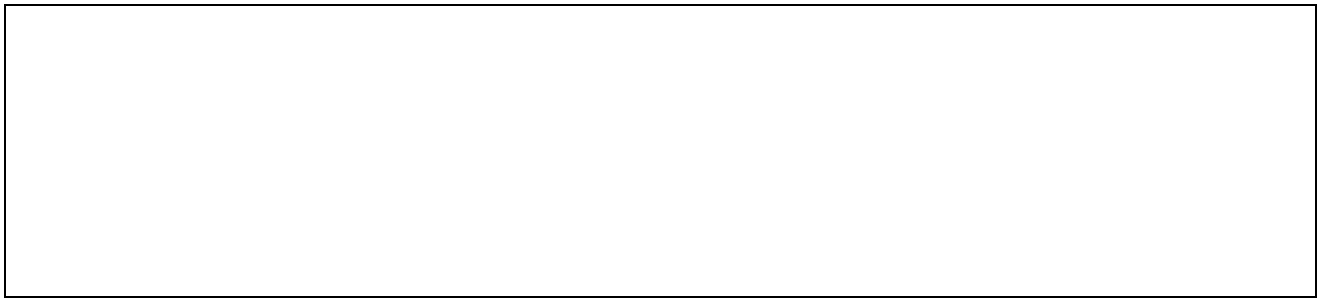
Aufgabe 1: Schleifen (10 Punkte)

Ergänzen Sie folgendes Java-Programm, so dass die Main-Methode die übergebenen Parameter nacheinander mittels `System.out.println` insgesamt viermal ausgibt. Jeweils einmal mit einer **while-Schleife**, einer **do-Schleife** und den **beiden Varianten** der **for-Schleife**.

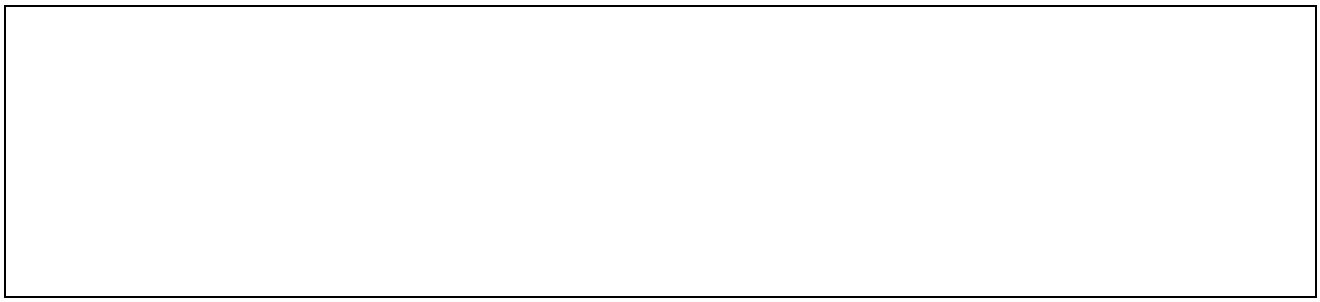
```
public class Loops {
      public static void main(String[] args) {

            // mit while-Schleife über args iterieren
```



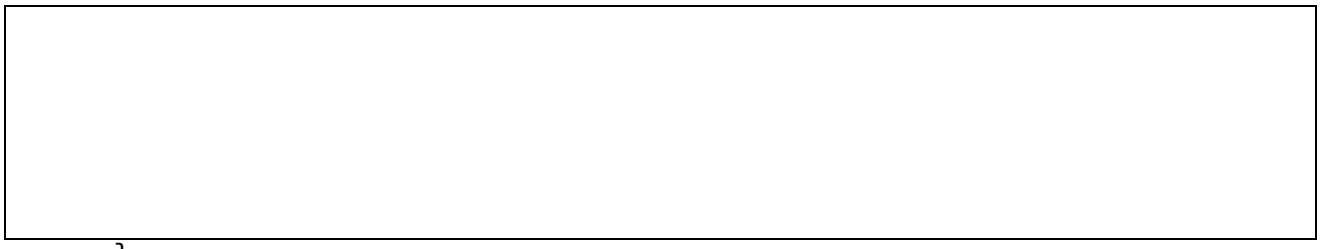
```
            // mit do-Schleife über args iterieren
```



```
            // mit erster Variante der for-Schleife über args iterieren
```



```
            // mit zweiter Variante der for-Schleife über args iterieren
```



```
      }
}
```

Aufgabe 2: Abstrakte Klassen & Interfaces (10 Punkte)

Nennen Sie **zwei** Gemeinsamkeiten von abstrakten Klassen und Interfaces

Nennen Sie **einen** Vorteil abstrakter Klassen gegenüber Interfaces

Nennen Sie **einen** Vorteil von Interfaces gegenüber abstrakten Klassen

Welche **drei** Compilerfehler erhalten Sie für folgendes Programm?

```
interface I { void m(); }
abstract class A { void n(); }
class B extends A implements I {
      void n(){}
      abstract void k();
}
```

Aufgabe 3: Polymorphie (10 Punkte)

Im folgenden Codebeispiel sind **vier Arten der Polymorphie** zu finden. Benennen Sie die vier Arten der Polymorphie und beschreiben Sie kurz, wo diese im Programm erkennbar werden.

```
class Polymorph<T extends Exception> {
      public void doSomethingWith(String s) {
            List<String> list = new LinkedList<String>();
            list.add(s);
            list.get(0).charAt(2);
      }
      public void doSomethingWith(T t) {
            List<T> list = new LinkedList<T>();
            list.add(t);
            list.get(0).printStackTrace();
      }
      public void doSomethingWith(Double d) {
            List list = new LinkedList();
            list.add(d);
            System.out.println(((Double)list.get(0)).isNaN());
      }
}
```

Aufgabe 4: static & innere Klassen (10 Punkte)

Statischen Methoden fehlt ebenso wie statischen inneren Klassen eine Eigenschaft, die den nichtstatischen Methoden bzw. nichtstatischen inneren Klassen implizit gegeben ist. Welche ist das?

Geben Sie ein Java-Programm an, welches (neben beliebigen weiteren Deklarationen) eine nichtstatische Methode und eine nichtstatische innere Klasse enthält und die Verwendung dieser impliziten Eigenschaft zeigt. Genauer: Ihr Programm soll zu Compilerfehlern führen, wenn Ihre nichtstatische Methode oder Ihre nichtstatische innere Klasse mit dem Modifizierer `static` versehen wird. Markieren Sie die kritischen Stellen, die die Compilerfehler auslösen.

Aufgabe 5: Objektgeflechte & Serialisierung (10 Punkte)

Zeichnen Sie das Objektgeflecht, das bei Beendigung der Main-Methode entstanden ist.

```
class Tripel {
      Tripel a = null;
      Tripel b = null;
      Tripel c = null;
}
```

```
class Main {
      Tripel x = new Tripel();
      Tripel y = new Tripel();
      Tripel z = new Tripel();
      void verbinde() {
            x.a = x.b = z.a = y;
            y.a = y.b = y.c = z.c = z;
            z.a = z.b = x.c = x;
      }
      public static void main(String[] args) {
            new Main().verbinde();
      }
}
```

Geben Sie **einen** Grund an, warum Serialisierung eine nichttriviale Aufgabe ist. Erläutern Sie dies anhand Ihrer Zeichnung.

Aufgabe 6: AWT (10 Punkte)

Gegeben sei folgendes Programm, welches in einem kleinen Fenster zwei Buttons anzeigen und bei Anklicken eines der beiden Buttons die Methode `buttonPressed` ausführen soll.

```
public class MyApplication extends java.awt.Frame {
      Button button1 = new Button("Button 1!");
      Button button2 = new Button("Button 2!");
      public MyApplication() {
            setSize(200,124);
            setTitle("My Application");
            add(button1);
            add(button2);
      }
      private void buttonPressed() {
            System.out.println("Button pressed!");
      }
      public static void main(String[] args) {
            MyApplication app = new MyApplication();
            app.addWindowListener(new WindowAdapter() {
                  public void windowClosing(WindowEvent e) {
                        System.exit(NORMAL);
                  }
            });
            app.setVisible(true);
      }
}
```

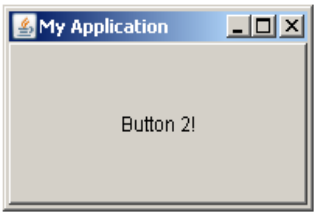


Abbildung 1:
Die resultierende Anzeige

Das Programm beinhaltet zwei Fehler. Einerseits wird nur einer der beiden Buttons angezeigt, andererseits haben Klicks auf den Button keinen Effekt.

Wo im Programm müssen welche Anweisungen ergänzt werden, damit beide Buttons sichtbar werden? (Größe und Anordnung der Buttons spielen dabei keine Rolle.) Sollten Ihnen konkrete Java-Anweisungen entfallen sein: beschreiben Sie in Worten, was fehlt.

(Fortsetzung der Aufgabe auf folgender Seite)

(Fortsetzung von Aufgabe 6)

Wie müssen Sie das Programm ergänzen, damit bei einem Klick auf `button1` bzw. `button2` die Methode `buttonPressed()` ausgeführt wird?

Nennen Sie drei Eigenschaften des AWT, die dieses als Programmgerüst kennzeichnen.

Aufgabe 7: Methodenbindung & Ausnahmen (10 Punkte)

Methoden können in ihrer Signatur `throws`-Deklarationen enthalten. Was verlangt Ihnen der Compiler ab, wenn Sie eine solche Methode aufrufen? Gilt dies für jede Ausnahme?

Gegeben sei folgendes Java-Programm.

```
public class Test {
      public static void main(String[] args) {
            Object o1 = new Object();
            Object o2 = new String();
            Double d = new Double(0);
            String s = new String();
            A a = new B();
            a.m(o1);
            a.m(o2);
            a.m(d);
            a.m(s);
      }
}

class A {
      void m(String s) {System.out.println("A.m(String s)");}
      void m(Object o) {System.out.println("A.m(Object o)");}
      private void m(Double d) {System.out.println("A.m(Double d)");}
}

class B extends A {
      void m(String s) {System.out.println("B.m(String s)");}
      void m(Object o) {System.out.println("B.m(Object o)");}
}
```

(Fortsetzung der Aufgabe auf folgender Seite)

(Fortsetzung von Aufgabe 7)

Welche Ausgabe erzeugt es? Geben Sie für jeden der vier `m`-Aufrufe genau an, warum die Ausgabe wie von Ihnen vorhergesagt lautet.

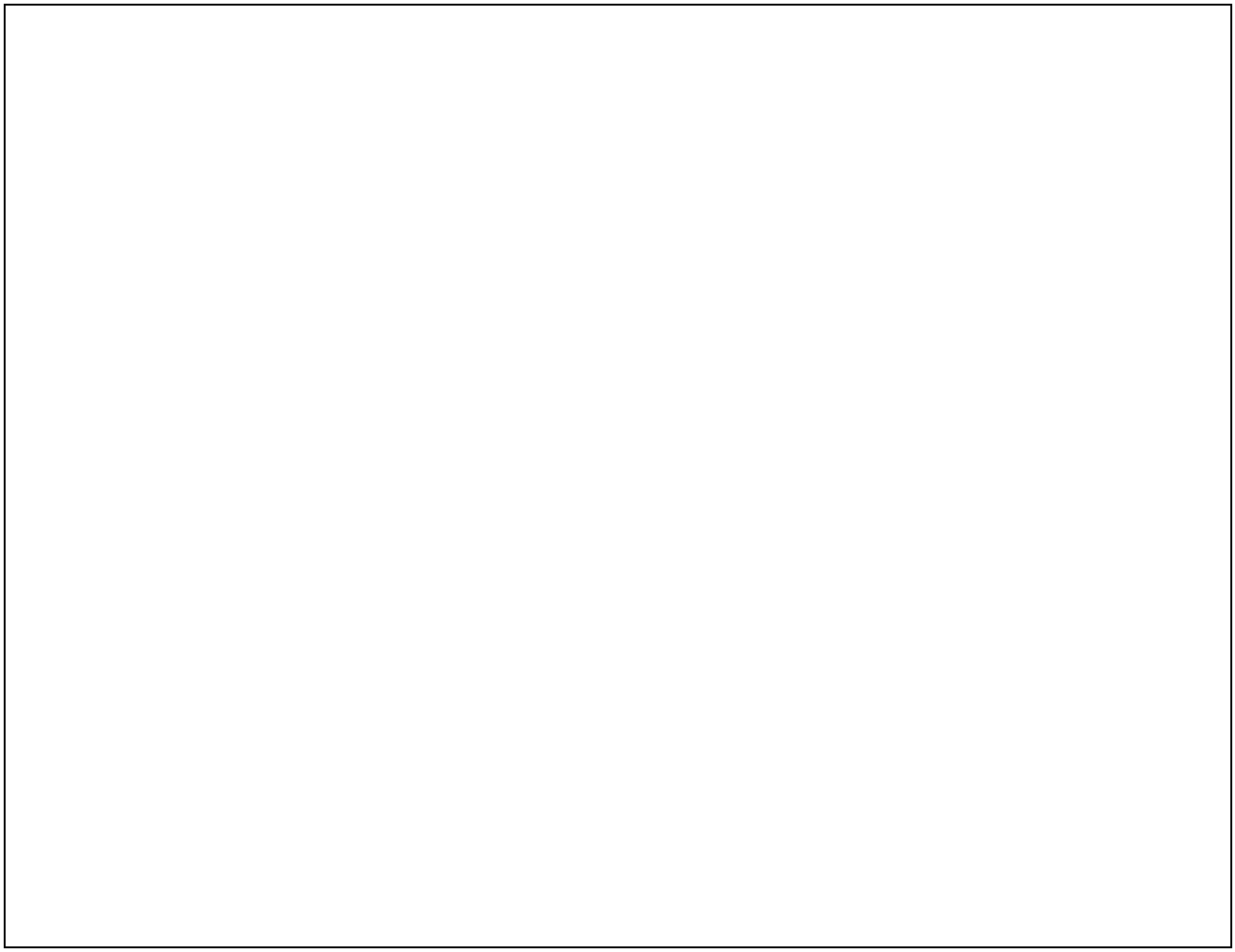
(Hinweis: Hilfreiche Stichworte in Ihrer Erklärung könnten *Compiler*, *Laufzeit*, *überladen*, *überschreiben* und *nicht zugreifbar* sein).

Aufgabe 8: Aufzählungstypen (10 Punkte)

Es sollen mit einem Aufzählungstyp verschiedene Tiere modelliert werden: Mäuse, Katzen und Elefanten. Ebenso soll auf den Elementen des Aufzählungstyps anhand von geeigneten Methoden feststellbar sein, ob ein Tier vor einem anderen Tier flieht, mit ihm Freundschaft schließt oder es verjagt. Elefanten verjagen Katzen, Katzen verjagen Mäuse und Mäuse verjagen Elefanten. Tiere gleicher Art schließen Freundschaft, und ein Tier flieht, wenn es verjagt wird.

Ergänzen Sie folgendes Programmfragment, damit die Main-Methode die in den Kommentaren angegebenen Ausgaben liefert.

```
public enum Tier {
```



```
    public static void main(String[] args) {
        System.out.println(ELEFANT.verscheucht(KATZE));   // true
        System.out.println(ELEFANT.verscheucht(ELEFANT));    // false
        System.out.println(ELEFANT.schliesstFreudschaftMit(ELEFANT)); // true
        System.out.println(ELEFANT.schliesstFreudschaftMit(KATZE));// false
        System.out.println(ELEFANT.wirdVerscheuchtVon(MAUS)); // true
        System.out.println(ELEFANT.wirdVerscheuchtVon(ELEFANT));  // false
    }
}
```

Aufgabe 9: Threads (10 Punkte)

Gegeben sei das folgende Programm

```
class Wert {
      int wert;
      Wert(int wert) {
            this.wert = wert;
      }
}

class Tausche extends Thread {
      Wert a,b;

      Tausche(Wert a, Wert b) {
            this.a = a;
            this.b = b;
      }
      public void run() {
            int h = a.wert;
            a.wert = b.wert;
            b.wert = h;
      }
      public static void main(String[] args) {
            Wert x = new Wert(0);
            Wert y = new Wert(1);
            Tausche tom = new Tausche(x,y);
            Tausche jerry = new Tausche(y, x);
            tom.start();
            jerry.start();
      }
}
```

Argumentieren Sie, warum nach Beendigung beider Threads – also dem zweifachen Tauschen der Werte – diese nicht zwangsläufig wieder ihre Ursprungsbelegung haben.

(Fortsetzung der Aufgabe auf folgender Seite)

(Fortsetzung von Aufgabe 9)

Erklären Sie, warum es keine gute Idee ist, das Problem durch folgende Änderung in der `run`-Methode zu beheben:

```
public void run() {
      synchronized (a) {
            synchronized (b) {
                  int h = a.wert;
                  a.wert = b.wert;
                  b.wert = h;
            }
      }
}
```

Welche Lösung schlagen Sie vor, damit keines der beiden erwähnten Probleme auftritt?

Aufgabe 10: Verteilte Systeme (10 Punkte)

Der Kurstext nennt **vier** verschiedene Kommunikationsmittel für verteilte Systeme, wovon zwei primär zur asynchronen und zwei primär zur synchronen Kommunikation eingesetzt werden. Nennen Sie diese und ordnen Sie sie der asynchronen bzw. synchronen Kommunikation zu.

Nennen Sie **zwei** Gründe, warum sich das objektorientierte Grundmodell besonders gut für die Programmierung verteilter Systeme eignet.

Zusätzlicher Platz für Ihre Lösungen

Ergänzung zu Aufgabe Nr.

Ergänzung zu Aufgabe Nr.

Zusätzlicher Platz für Ihre Lösungen

Ergänzung zu Aufgabe Nr.

Ergänzung zu Aufgabe Nr.