FernUniversität in Hagen

Fakultät für Mathematik und Informatik

Lehrgebiet Programmiersysteme
Prof. Dr. Friedrich Steimann

(Name, Vorname)

(Straße, Nr.)

(PLZ) (Wohnort)

(Land, falls außerhalb Deutschlands)

FernUniversität in Hagen D-58084 Hagen

Kurs 1618 SS 2008

„Einführung in die objektorientierte Programmierung"

Klausur am 13.09.2008

Dauer: 3 Std., 10 – 13 Uhr

Lesen Sie zuerst die Hinweise auf der Rückseite!

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

Klausurort: .....

Aufgabe	1	2	3	4	5	6	7	8	9	10	Summe
habe bearbeitet											
maximal	8	11	10	10	8	10	12	10	12	12	103
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 Nachklausur findet am 07.02.2009 statt; wenn Sie teilnehmen möchten, melden Sie sich bitte bis zum 07.01.2009 an (genauere Informationen s. Webseiten des Lehrgebietes).

Hagen, den 20.09.2008 im Auftrag

1. Prüfen Sie die Vollständigkeit Ihrer Unterlagen. Die Klausur umfasst:
   - 1 Deckblatt,
   - 10 Aufgaben auf Seite 1 bis Seite 14.

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 Leerraum. Benutzen Sie auf keinen Fall die Rückseiten der Aufgabenblätter. Versuchen Sie, mit dem vorhandenen Platz auszukommen, sie dürfen auch stichwortartig antworten. Für den Notfall können Sie gekennzeichnetes Papier bei der Klausuraufsicht bekommen. **Es werden nur Aufgaben gewertet, die sich auf dem offiziellen Klausurpapier (Klausur + gekennzeichnetes Papier) befinden.** Eigenes Papier ist nur für Ihre persönlichen Notizen erlaubt und von der Benotung ausgeschlossen.

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

5. Geben Sie die gesamte Klausur und, falls vorhanden, gekennzeichnete Zusatzblätter ab.

6. Es sind *keine Hilfsmittel* zugelassen.

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

8. Achten Sie darauf, dass Sie bei Programmieraufgaben Ihre Lösungen sinnvoll kommentieren; es könnten Ihnen sonst Punkte abgezogen werden.

9. Es sind maximal 103 Punkte erreichbar. Wenn Sie mindestens 42 Punkte erreichen, haben Sie die Klausur mit Sicherheit bestanden.

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

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

Aufgabe 1: Schleifen (8 Punkte)

Das folgende Java-Programm soll alle beim Programmaufruf mitgegebenen Programmparameter auf der Standardausgabe jeweils in einer eigenen Zeile ausgeben.
Wie sehen die zugehörigen Befehle für die vier verschiedenen Schleifentypen aus? Tragen Sie sie bitte in das hier stehende Programmskelett ein!

```
public class Schleifen {
  public static void main(String[] args) {
    //Mit while-Schleife









    //Mit do-Schleife









    // Mit erster Variante der for-Schleife









    // Mit zweiter Variante der for-Schleife









  }
}
```

Aufgabe 2: Sichtbarkeit (11 Punkte)

Gegeben ist der folgende Programmausschnitt:

```
package a;
public class A {
   public int i;
   int j;
   protected int k;
   private int l;
   public int m() {...}
   int n() {...}
   protected int o() {...}
   private int p() {...}
}

package a;
public class B {...}

package a.a;
import a.A;
public class C {...}

package a.a;
import a.A;
public class D extends A {...}
```

Auf einer per `A a` deklarierten Variable sind abhängig davon, innerhalb welcher Klasse die Deklaration steht, verschiedene Elemente zugreifbar.

a) Tragen Sie bitte in die folgende Liste ein, um welche Elemente es sich jeweils handelt:

   - a ist in Klasse A selbst deklariert:

   - a ist in Klasse B deklariert:

   - a ist in Klasse C deklariert:

   - a ist in Klasse D deklariert:

b) Auf welche in A deklarierten Attribute und Methoden kann man über die Variable a zugreifen, wenn a in der Klasse D mit Typ D statt A deklariert ist?

Aufgabe 3: Zuweisungen (10 Punkte)

a) Sind die markierten Zuweisungen von Werten an die Attribute und Variablen xwert, zaehler2 und ywert korrekt? Begründen Sie Ihre Antwort.

```
class C {
  final int xwert, ywert = 9;
  C(){
    xwert = 7;  // korrekt? Ja oder Nein?
    /* Begründung */



    mult();
  }
  void mult(){
    final int zaehler1 = 10, zaehler2 = -10;
    zaehler2 = 0;  // korrekt? Ja  oder Nein?
    /* Begründung */




    ywert = 27; // korrekt? Ja oder Nein?
    /* Begründung */




    ...
  }
}
```

b) Sind die folgenden Zuweisungen zulässig? Geben Sie eine kurze Begründung Ihrer Antwort an.
(`Tier` ist in dem Beispiel ein Klassentyp, `Eisbaer` ist Subtyp von `Tier`.)

```
Object   o     = new String[6]; /* Zulässig Ja oder Nein? */
/* Begrüngung der Antwort: */




Tier[]   tier  = new Eisbaer[6];/* Zulässig Ja oder Nein? */
/* Begründung der Antwort */




Eisbaer[]baer  = new Tier[6];   /* Zulässig Ja oder Nein */
/* Begründung der Antwort */
```

Aufgabe 4: objektorientierte Denkweise (10 Punkte)

Die Programmierin Daniela K. möchte ein objektorientiertes Programm für ein Motorrad schreiben. Sie beginnt mit folgendem Programmfragment:

```
public class Motorrad {

    private boolean motorLaeuft;

    public void start() {
        motorLaeuft = true;



    }

    public static void main(String[] args) {



        start();



    }
}
```

Der Comiler meldet ihr:

```
Compiler: "Cannot make a static reference to the non-static method
start() from the type Motorrad".
```

So kommt sie auf die Idee, die Methode `start` `static` zu machen.

a) Warum ist das falsch?

b) Wie könnte man den Fehler im Programm beheben?
Tragen Sie die Änderung in den obigen Programmtext ein.

Aufgabe 5: Interfaces, Streams, generische Typen (8 Punkte)

Gegeben seien die folgenden beiden Interfaces `ReadStream` und `WriteStream` und die Klasse `IOStream`.

```
interface ReadStream<T> {
   T read();
}

interface WriteStream<T> {
   void write(T object);
}

class IOStream<T> implements ReadStream<T>, WriteStream<T> {...}
```

Ihre Aufgabe ist, unter Verwendung des jeweils geeigneten der drei oben angegebenen Typen, eine Klasse `WillNurLesen` und eine Klasse `WillNurSchreiben` zu erstellen. `WillNurLesen` stellt nur Leseoperationen für String-Objekte bereit, `WillNurSchreiben` nur Schreiboperationen für String-Objekte.
Ergänzen Sie dazu das unten angegebene Programmgerüst.

```
class WillNurLesen {
   /* Definieren Sie eine Variable eingabe zum Lesen von Strings */
   /* bitte hier Text ergänzen */



   String lese(){
      /* Befehl zum Lesen */
      /* bitte hier Text ergänzen */



   }
}

 class WillNurSchreiben {
    /* Definieren Sie eine Variable ausgabe zum Schreiben von Strings */
     /* bitte hier Text ergänzen */



    void schreibe(String st){
       /* Befehl zum Schreiben */
        /* bitte hier Text ergänzen */



    }
 }
```

Aufgabe 6: Objektströme (10 Punkte)

a) Objektströme müssen bei der Ausgabe mehrfach referenzierte Objekte erkennen und Zirkularitäten auflösen können. Wie machen sie das?

`Antwort: ( in 3 bis 7 Sätzen oder stichpunktartig)`

b) Damit ein Objektstrom das von einem Objekt erreichbare Geflecht vollständig ausgeben kann, muss er auch Zugriff auf die privaten Attribute des Geflechts haben. Würde man Objektströmen diesen Zugriff immer automatisch gewähren, könnten Objektströme missbraucht werden, um geschützte Implementierungsteile auszuforschen. Wie löst Java dieses Problem?

`Antwort: (in 3 bis 7 Sätzen oder stichpunktartig)`

Aufgabe 7: Überschreiben und Überladen (12 Punkte)

Gegeben ist das folgende Programm. Welche Ausgabe liefert es und warum?

```
public class TierTest {
   public static void main(String[] args) {
      Tier t1   = new Tier();
      Tier t2   = new Vogel();
      Fisch f   = new Karpfen();
      Vogel v1  = new Vogel();
      Vogel v2  = new Spatz();
      Huhn h    = new Huhn();
      Karpfen k = new Karpfen();

      Super sup1 = new Super();
      Super sup2 = new Sub();

      sup1.m(h, v2);
      sup2.m(v1, k);
      sup1.m(t1, t2);
      sup1.m(v1, k);
      sup1.m(v2, f);
      sup2.m(v1, f);
   }
}

class Super {
   public void m(Tier t1, Tier t2) {
      System.out.println("1");
   }

   public void m(Tier t, Fisch f) {
      System.out.println("2");
   }

   public void m(Fisch f, Tier t) {
       System.out.println("5");
   }
}

class Sub extends Super {
   public void m(Tier t1, Fisch t2) {
      System.out.println("3");
   }

   public void m(Vogel v, Fisch f) {
      System.out.println("4");
   }
}

class Tier {
}
```

```
class Fisch extends Tier {
}

class Vogel extends Tier {
}

class Huhn extends Vogel {
}

class Spatz extends Vogel {
}

class Karpfen extends Fisch {
}
```

`Antwort:`
`sup1.m(h, v2);` ausgegeben wird:
denn:

`sup2.m(v1, k);` ausgegeben wird:
denn:

`sup1.m(t1, t2);` ausgegeben wird:
denn:

`sup1.m(v1, k);` ausgegeben wird:
denn:

`sup1.m(v2, f);` ausgegeben wird:
denn:

`sup2.m(v1, f);` ausgegeben wird:
denn:

Aufgabe 8: abstrakte Klassen (10 Punkte)

- Was gilt für die Implementierung abstrakter Klassen?
  `Antwort:`

- Was können abstrakte Klassen an Subklassen vererben?
  `Antwort:`

- Lassen sich abstrakte Klassen instanziieren? ?
  `Antwort:`

- Wozu können abstrakte Klassen verwendet werden?
  `Antwort:`

- Java verzichtet auf Mehrfachvererbung. Was macht man in Java, wenn eine Klasse mehrere Supertypen besitzen soll?
  `Antwort:`

Aufgabe 9: Parallelität (12 Punkte)

Am Lehrgebiet Programmiersysteme benutzen Daniela und Ursula denselben Drucker. Jede druckt eine Datei fünfmal auf dem Drucker aus. Die Namen der zu druckenden Dateien werden der `main`-Methode der Klasse `Test` als Parameter übergeben. Das einmalige Drucken einer Datei sei ein *Druckjob*. Jede Benutzerin initiiert also fünf Druckjobs.

```
import java.io.*;

class Drucker {
  void druckeDatei(String dateiname) {
    try {
      BufferedReader in = new BufferedReader(new FileReader(dateiname));
      String line = in.readLine();
      while (line != null) {
        // Zeile line auf dem Drucker ausgeben
        ...
        line = in.readLine();
      }
    }
    catch (Exception e) {
      System.out.println("Eine Ausnahme ist aufgetreten.");
    }
  }
}

class Benutzer extends Thread {
  Drucker drucker;
  String dateiname;
  int anzahl;

  Benutzer(Drucker drucker, String dateiname, int anzahl) {
    this.drucker = drucker;
    this.dateiname = dateiname;
    this.anzahl = anzahl;
  }

  public void run() {
    for (int i=0; i<anzahl; i++) {
      drucker.druckeDatei(dateiname);
    }
  }
}

class Test {
  public static void main (String[] argv) {
    if (argv.length >= 2) {
      Drucker d = new Drucker();
      Benutzer daniela  = new Benutzer(d, argv[0], 5);
      Benutzer ursula = new Benutzer(d, argv[1], 5);
      daniela.start();
      ursula.start();
    }
    else
      System.out.println("Bitte zwei Dateinamen als Argumente uebergeben!");
  }
}
```

Bei dem angegebenen Programm kann es passieren, dass ein Druckjob von Ursula mit einem Druckjob von Daniela vermischt auf dem Drucker ausgegeben wird. Synchronisieren Sie die beiden Threads für Daniela und Ursula nun so, dass zunächst alle Druckjobs der einen Benutzerin gedruckt werden und erst im Anschluss daran die Druckjobs der anderen Benutzerin. Begründen Sie, warum Ihr Vorschlag das Problem löst.

Tragen Sie hier die erste Programmänderung ein:

Tragen Sie hier die zweite Programmänderung ein:

Tragen Sie hier Ihre genaue Begründung ein:

Aufgabe 10: Beobachter (12 Punkte)

Gegeben sind die folgenden Klassen `Ampel`, `SchaltzustandEvent`, `SchaltzustandListener`, `Ampelzustand`, `Ampelsteuerung`.

a.) Ergänzen Sie das Interface SchaltzustandListener so, dass es hier sinnvoll eingesetzt werden kann.

b.) Beschreiben Sie in ein bis zwei Sätzen, was die Klasse Ampelsteuerung tut. Ergänzen Sie dann die Kommentare in der Klasse Ampelsteuerung.

```
import java.awt.*;

public class Ampel extends Canvas implements SchaltzustandListener {

/* Diese Klasse stellt eine Ampel in einem Canvas-Objekt
   durch drei untereinanderliegende Kreise dar.
   Sie reagiert auf die Ereignisse rot, gelb, gruen und
   rot\_gelb, indem sie die entsprechenden, zum Zustand
   rot, gelb etc. gehoerenden Kreise der Ampel in der(den)
   entsprechenden Farbe(n) anzeigt.
   Die anderen Kreise werden grau dargestellt.
   Nach Erzeugung eines Ampelobjekts sind zunaechst alle
   Kreise grau, d.h. die Ampel ist aus. */
}
```

Die Klasse `Ampel` verwendet die Klassen `SchaltzustandEvent` und `SchaltzustandListener`:

```
import java.util.*;

public class SchaltzustandEvent extends EventObject {
  public SchaltzustandEvent(Object source) {
    super(source);
  }
}


// Diese Klasse  passend ergänzen:
public interface SchaltzustandListener
 {









}


import java.awt.*;
import java.awt.event.*;
import java.util.*;
```

```
public class Ampelzustand extends Button implements ActionListener {

/* Die Klasse Ampelzustand ist von java.awt.Button abgeleitet und
   realisiert die Weiterschaltung des Ampelzustands.
   Der Zustand wird bei jedem Mausklick weitergeschaltet.
   Die Zustandsuebergaenge sind:
     aus
     rot
     rot\_gelb
     gruen
     gelb
     rot
     ...
     rot
     ... .
   Immer, wenn sich der Zustand aendert, wird ein entsprechendes Ereignis, z.B.
   rot\_gelb gefeuert. */

}

public class Ampelsteuerung {

   // Bitte ergaenzen: Was macht die Klasse?
   //
   //
   //
   //
   //
   //
   //
   //
   //
   //
   //
   //

  public static void main(String[] args) {
    Ampelzustand ampelzustand = new Ampelzustand();
    Ampel ampel = new Ampel();

    //Bitte Kommentar ergaenzen
    //
    //
    //
    ampelzustand.addSchaltzustandListener(ampel);

    //Bitte Kommentar ergaenzen
    //
    //
    //
    Frame f = new Frame();
    BorderLayout borderLayout1 = new BorderLayout();
    f.setSize (310,450);
```

```
    f.setLocation (100,100);
    f.setLayout(borderLayout1);

    //Bitte Kommentar ergaenzen
    //
    //
    //
    f.add(ampel, BorderLayout.CENTER);
    f.add(ampelzustand, BorderLayout.SOUTH);

    //Bitte Kommentar ergaenzen
    //
    //
    //
    f.addWindowListener(new WindowAdapter() {
      public void windowClosing (WindowEvent e) {
        // Programm beenden
        System.exit (0);
      }
    });

    //Bitte Kommentar ergaenzen
    //
    //
    //
    f.setVisible (true);
  }
}
```