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 01618

Einführung in die objektorientierte Programmierung

(Kursdurchführung des Sommersemester 2017)

Klausur am 16.09.2017

Zweistündige Klausur
(10.00 – 12.00 Uhr)

Lesen Sie zuerst die Hinweise auf der folgenden Seite!

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

Klausurort: _____

Aufgabe	1	2	3	4	5	6	7	8	Summe
habe bearbeitet									
maximal	10	10	10	10	10	10	10	10	80
erreicht									
Korrektur									

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

○ Sie haben die Klausur leider nicht bestanden. Für den nächsten Versuch wünschen wir Ihnen viel Erfolg. Die nächste Klausur findet im Wintersemester 2017 / 2018 statt.

Hagen, den 27.09.2017

Im Auftrag

Hinweise zur Bearbeitung

1. Prüfen Sie die Vollständigkeit Ihrer Unterlagen. Die Klausur umfasst auf insgesamt 18 Seiten:
   - 1 Deckblatt
   - Diese Hinweise zur Bearbeitung
   - Ein Formular zur Bescheinigung der Teilnahme an dieser Klausur gegenüber dem Finanzamt
   - 8 Aufgaben auf Seite 4-16
   - Zwei zusätzliche Seiten 17 und 18 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 keinen 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 dieser Klausur angefügten 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 80 Punkte erreichbar. Wenn Sie mindestens 40 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: Klassenimplementierung (10 Punkte)

Gegeben sei das folgende Programmfragment:

```
public class HeimkinoDemo {

	public static void main(String[] args) {
		TonSystem m = new TonSystem(2);
		BluRayPlayer t = new BluRayPlayer();
		TvGerät c = new TvGerät(t, m);

		c.bluRayPlayer.einlegen(new BluRayMedium("Star Wars"));

		try {
			c.tonSystem.erwarteKanäle(6);
			System.out.println("Surround-Sound-Tonsystem.");
		} catch (Exception e) {
			System.out.println("Kein Surround-Sound Tonsystem.");
		}

		try {
			c.tonSystem.erwarteKanäle(2);
			System.out.println("Stereo-Tonsystem.");
		} catch (Exception e) {
			System.out.println("Kein Stereo-Tonsystem.");
		}
	}

}
```

Geben Sie Implementierungen für die Klassen `TonSystem`, `BluRayPlayer`, `BluRayMedium` und `TvGerät` an, sodass dieses Programmfragment kompiliert werden kann und bei Terminierung der `main`-Methode die folgenden Zeilen auf der Konsole ausgegeben werden.

```
Tonsystem mit 2 Kanälen erzeugt.
Film 'Star Wars' eingelegt.
Kein Surround-Sound Tonsystem.
Stereo-Tonsystem.
```

Diese Zeichenketten sollen selbstverständlich durch die jeweiligen Anweisungen zusammengesetzt und ausgegeben werden.

(Fortsetzung der Aufgabe auf der folgenden Seite)

(Fortsetzung von Aufgabe 1)

Aufgabe 2: (Beschränkt) param. Polymorphie (10 Punkte)

Gegeben sei das folgende Programmfragment:

```
public class Wohnzimmer {
      public static void main(String[] args) {
            Kiste kiste = new Kiste();
            Regal regal = new Regal();
            Brettspiel schach = new Brettspiel(kiste);
            Buch herrDerRinge = new Buch(regal);
            Gegenstand<?> einGegenstand = spiel;

            Möbelstück möbel = einGegenstand.lagerort();
            Kiste spieleKiste = spiel.lagerort();
            Regal bücherRegal = buch.lagerort();
            /* [1] */
      }
}

class Möbelstück { }
class Kiste extends Möbelstück { }
class Regal extends Möbelstück { }
```

Geben Sie eine Implementierung für die Klassen `Gegenstand`, `Brettspiel` und `Buch` an, sodass dieses Programmfragment kompiliert. Die Methode `lagerort()` soll insgesamt nur ein einziges Mal implementiert werden. (5 Punkte)

(Fortsetzung der Aufgabe auf der folgenden Seite)

(Fortsetzung von Aufgabe 2)

Die `main`-Methode wird nun an der mit `/* [1] */` markierten Stelle um die folgenden Zeilen ergänzt:

```
Kiste andereKiste = new Kiste();
Brettspiel dame = new Brettspiel(andereKiste);
schach.tauschePlatz(dame);

Regal anderesRegal = new Regal();
Buch harryPotter = new Buch(anderesRegal);
herrDerRinge.tauschePlatz(harryPotter);
```

Während diese zwei Aufrufe von `tauschePlatz(.)` durch den Compiler akzeptiert werden sollen, soll der Aufruf von

```
herrDerRinge.tauschePlatz(schach);
```

zurückgewiesen werden.

Geben Sie eine (einzige) Implementierung der Methode `tauschePlatz(.)` an, die diese Kriterien erfüllt. In welcher Klasse steht diese? (2,5 Punkte)

Erklären Sie, warum der Compiler den letztgenannten Aufruf zurückweist. (2,5 Punkte)

Aufgabe 3: Subtyping (10 Punkte)

Was besagt die Grundregel des Subtyping? (2 Punkte)

Welche syntaktischen Voraussetzungen müssen erfüllt sein, wenn in einer Subklasse eine geerbte Methode überschrieben werden soll? (3 Punkte)

Ist es ausreichend, die syntaktischen Voraussetzungen zu überprüfen, um sicherzustellen, dass ein Typ ein Subtyp ist? Welche besondere Herausforderung liegt hierin für den Programmierer / die Programmiererin? (5 Punkte)

Aufgabe 4: Polymorphie (10 Punkte)

Welche vier Arten von Polymorphie kennen Sie? (2 Punkte)

Charakterisieren Sie jede Art kurz! (jeweils 2 Punkte)

Aufgabe 5: Arrays (10 Punkte)

Gegeben sei das folgende Programmfragment, welches die Zahlen n mit $1 \leq n \leq max$ in Hexadezimalnotation konvertieren, in ein Array speichern und ausgeben soll:

```
public class ArrayDemo {
	public static void main(String[] args) {
		int max = Integer.parseInt(args[0]);
		/* [1] */
		/* [2] */
		/* [3] */
	}
}
```

Teilaufgabe a: Erste Implementierung **(4 Punkte)**

Hinweis:
Die statische Bibliotheksfunktion `Integer.toHexString(int)` konvertiert die per Parameter angegebene Zahl in Hexadezimalnotation. So liefert `Integer.toHexString(11)` etwa die Zeichenkette "`b`".

Wodurch muss `/* [1] */` ersetzt werden, um das (eindimensionale) Array mit dem Namen `zahlen` zu initialisieren? (1 Punkt)

Wodurch muss `/* [2] */` ersetzt werden, um die Hexadezimalnotationen aller n mit $1 \leq n \leq max$ in dieses Array zu schreiben? Bitte verwenden Sie eine Schleife! (2 Punkte)

Wodurch muss `/* [3] */` ersetzt werden, um den Inhalt des Arrays auf die Konsole zu schreiben? Bitte verwenden Sie eine Schleife – und zwar eine Schleife einer anderen Art als die, die Sie in Ihrer Implementierung zur Ersetzung von `/* [2] */` verwendet haben! (1 Punkt)

(Fortsetzung der Aufgabe auf der folgenden Seite)

(Fortsetzung von Aufgabe 5)

Teilaufgabe b: Zweite Implementierung (6 Punkte)

In der zweiten Implementierung soll nun nicht nur die Hexadezimalnotation einer Zahl gespeichert werden, sondern auch ihre Binärnotation. Dazu soll ein zweidimensionales Array (also ein Array von Arrays) verwendet werden.

Hinweis:
Analog zu `Integer.toHexString(int)` existiert auch eine Methode `Integer.toBinaryString(int)`, die der Konvertierung einer Zahl in Binärnotation dient; so liefert `Integer.toBinaryString(73)` etwa die Binärnotation von `73`, also die Zeichenkette „`1001001`".

Wodurch muss `/* [1] */` ersetzt werden, um das Array mit dem Namen `zahlen` zu initialisieren, welches aus Arrays der Dimension 2 besteht? (2 Punkt)

Wodurch muss `/* [2] */` ersetzt werden, um die Paare Binär- und Hexadezimalnotation in dieses Array zu schreiben? Bitte verwenden Sie eine Schleife! (2 Punkte)

Wodurch muss `/* [3] */` ersetzt werden, um die Zahlen in Binär- und Hexadezimalnotation (genauer: in dem Format „`x : y`", wobei `x` die Binär- und `y` die Hexadezimanotation ist) auf die Konsole zu schreiben? Bitte verwenden Sie eine Schleife – und zwar eine Schleife einer anderen Art als die, die Sie in Ihrer Implementierung zur Ersetzung von `/* [2] */` verwendet haben. (2 Punkt)

Aufgabe 6: Kontrollfluss und der this-Parameter (10 Punkte)

Gegeben sei folgendes Java-Programm:

```
public class EineNeueKlasse {
	String tier = "Pinguin";
	public static void main(String[] args) {
		new EineNeueKlasse("Biene").m(42L);
	}
	EineNeueKlasse(String tier) {
		System.out.println(this.tier);
	}
	void m(Number n){
		System.out.println("Esel");
		new WeitereInnereKlasse().methode(new SubInnereKlasse("Giraffe"));
	}
	void m(Long l) {
		System.out.println("Löwe");
		new WeitereInnereKlasse().methode(new SubInnereKlasse("Giraffe"));
	}

	class InnereKlasse {
		String tier = "Elefant";
		InnereKlasse(String tier) {
			System.out.println(this.tier);
		}
		void n(WeitereInnereKlasse weitereInnere) {
			System.out.println("Hund");
		}
		void n(EineNeueKlasse eineNeue) {
			System.out.println("Hamster");
		}
	}

	class SubInnereKlasse extends InnereKlasse{
		String tier = "Adler";
		public SubInnereKlasse(String tier) {
			super("Hase");
			System.out.println(tier);
		}
	}

	static class WeitereInnereKlasse {
		{
			System.out.println("Bär");
			/* [2] */
		}
		void methode(InnereKlasse innereKlasse) {
			innereKlasse.n(this);
		}

	}

	static void andereMethode() {
		/* [1] */
	}

}
```

(Fortsetzung der Aufgabe auf der folgenden Seite)

(Fortsetzung von Aufgabe 6)

Welche Zeilen wurden nach Terminierung der `main(.)`-Methode auf der Konsole ausgegeben? (6 Punkte)

An der mit `/* [1] */` markierten Stelle wird die folgende Anweisung eingefügt. Warum weist der Compiler sie zurück?

```
System.out.println(this.tier);
```

(2 Punkte)

An der mit `/* [2] */` markierten Stelle wird die folgende Anweisung eingefügt. Warum weist der Compiler sie zurück?

```
System.out.println(EineNeueKlasse.this.tier);
```

(2 Punkte)

Aufgabe 7: Aufzählungstypen (10 Punkte)

Die Monate der Freibadsaison (Mai bis September) sollen als Aufzählungstyp `SaisonMonat` implementiert werden. Die Werte des Aufzählungstypen sollen über die folgenden vier Methoden verfügen:

- `boolean istNach(SaisonMonat monat)`: Liefert `true`, wenn der durch das Empfängerobjekt bezeichnete Monat *vor* dem Monat liegt, der durch das Objekt bezeichnet wird, welches durch den Parameter `monat` referenziert wird; ansonsten `false`. (Wobei angenommen wird, dass beide Objekte für Monate des gleichen Jahres stehen.)
- `boolean istVor(SaisonMonat monat)`: Liefert `true`, wenn der durch das Empfängerobjekt bezeichnete Monat *nach* dem Monat liegt, der durch das Objekt bezeichnet wird, welches durch den Parameter `monat` referenziert wird; ansonsten `false`. (Wobei angenommen wird, dass beide Objekte für Monate des gleichen Jahres stehen.)
- `SainsonMonat naechsterSaisonMonat()`: Liefert das Element der Aufzählung, welches den Monat beschreibt, der unmittelbar *nach* dem durch das Empfängerobjekt bezeichneten Monat liegt; wobei dies auch der erste Monat des folgenden Jahres sein kann.
- `SainsonMonat vorherigerSaisonMonat()`: Liefert das Element der Aufzählung, welches den Monat beschreibt, der unmittelbar *vor* dem durch das Empfängerobjekt bezeichneten Monat liegt; wobei dies auch der letzte Monat der vorherigen Saison sein kann.

Die `main`-Methode der Klasse `FreibadSaisonDemo` verdeutlicht die Bedeutung der Methoden anhand einiger Beispiele. Der in jeder Zeile hinten angefügte Kommentar gibt dabei jeweils den Wert an, zu dem der Ausdruck dieser Zeile auswerten soll.

```
public class FreibadSaisonDemo {
	public static void main(String[] args) {
	    SaisonMonat.JULI.istVor(SaisonMonat.MAI);         // false
	    SaisonMonat.JULI.istVor(SaisonMonat.JUNI);        // false
	    SaisonMonat.JULI.istVor(SaisonMonat.JULI);        // false
	    SaisonMonat.JULI.istVor(SaisonMonat.AUGUST);      // true
	    SaisonMonat.JULI.istVor(SaisonMonat.SEPTEMBER);   // true

	    SaisonMonat.JULI.istNach(SaisonMonat.JUNI);       // true
	    SaisonMonat.JULI.istNach(SaisonMonat.JULI);       // false
	    SaisonMonat.JULI.istNach(SaisonMonat.AUGUST);     // false

	    SaisonMonat.JULI.naechsterSaisonmonat();          // FreibadSaison.AUGUST
	    SaisonMonat.JULI.vorherigerSaisonmonat();         // FreibadSaison.JUNI

	    SaisonMonat.SEPTEMBER.naechsterSaisonmonat();     // FreibadSaison.MAI
	    SaisonMonat.MAI.vorherigerSaisonmonat();          // FreibadSaison.SEPTEMBER
	}
}
```

Da das Freibad in Kürze in der Lage ist, das Wasser der Becken zu beheizen, zieht der Betreiber in Erwägung, die Saison auf die angrenzenden Monate April und Oktober auszudehnen. Vervollständigen Sie folgende Implementierung der Klasse `FreibadSaison` derart, dass die Methodenimplementierungen bei einer Erweiterung des Aufzählungstyps um die Elemente `APRIL` und `OKTOBER` nicht verändert werden müssen.

(Fortsetzung der Aufgabe auf folgender Seite)

(Fortsetzung von Aufgabe 7)

```
enum SaisonMonat {
      MAI, JUNI, JULI, AUGUST, SEPTEMBER;
```

(1 Punkt)

```
      boolean istNach(SaisonMonat monat) {
```

```
      }
```

(1 Punkt)

```
      boolean istVor(SaisonMonat monat) {
```

```
      }
```

(4 Punkte)

```
      SaisonMonat naechsterSaisonMonat() {
```

```
      }
```

(4 Punkte)

```
      SaisonMonat vorherigerSaisonMonat() {
```

```
      }
}
```

Aufgabe 8: Parallelität (10 Punkte)

Welche Varianten lokaler Parallelität lassen sich unterscheiden? Erklären Sie sie! (3 Punkte)

Warum ist es auch dann von Vorteil, ein Softwaresystem so zu implementieren, dass es parallel ausgeführt werden kann, wenn es nur auf einem Einprozessorsystem laufen soll? (3 Punkte)

Welche Aufgabe hat der Scheduler? Welche Arten des Schedulings können unterschieden werden? Wann ist das Scheduling fair? (4 Punkte)

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.