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 2016)

Klausur am 10.09.2016

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	Summe
habe bearbeitet							
maximal	20	10	10	10	10	20	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 2016 / 2017 statt.

Hagen, den 21.09.2016

Im Auftrag

Hinweise zur Bearbeitung

1. Prüfen Sie die Vollständigkeit Ihrer Unterlagen. Die Klausur umfasst auf insgesamt 16 Seiten:
   - 1 Deckblatt
   - Diese Hinweise zur Bearbeitung
   - 6 Aufgaben auf Seite 3-14
   - Zwei zusätzliche Seiten 15 und 16 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 (20 Punkte)

Gegeben sei die folgende Klasse `Circle`:

```
public class Circle {
	protected int radius01;

	public Circle(int radius) {
		radius01 = radius;
	}

	public double getArea() {
		return Math.PI * radius01 * radius01;
	}
}
```

Schreiben Sie eine Klasse `Ellipse`, die die Klasse `Circle` erweitert und deren Instanzvariable `radius01` als eine seiner zwei Halbachsen nutzt. Ergänzen Sie einen Konstruktor, der die Halbachsen initialisiert und überschreiben Sie die Methode `getArea()` in geeigneter Weise. (5 Punkte)

Hinweis: Die Fläche einer Ellipse ist das Produkt der Längen ihrer Halbachsen und π.

(Fortsetzung der Aufgabe auf der folgenden Seite)

(Fortsetzung von Aufgabe 1)

Gegeben sei die folgende Testklasse:

```
public class TestCircleEllipse {
	public static void main(String[] args) {
		Ellipse e = new Ellipse(1, 2);
		System.out.println(e.getArea());

		Circle c = e;
		System.out.println(c.getArea());
	}
}
```

Geben Sie die Ausgabe des Programms an und begründen Sie Ihre Antwort. Berücksichtigen Sie dabei in Ihrer Antwort auch folgende Fragen:
- Welche Methoden-Implementierungen werden bei `e.getArea()` und bei `c.getArea()` aufgerufen?
- Für welches objektorientierte Prinzip sind die Methoden `getArea()` und deren Aufrufe beispielhaft?

Hinweis: Für die Angabe der Programmausgabe reichen zwei Nachkommastellen aus; rechnen Sie hierzu mit $\pi \approx 3{,}14$. (5 Punkte)

(Fortsetzung der Aufgabe auf der folgenden Seite)

(Fortsetzung von Aufgabe 1)

Implementieren Sie den Rumpf der Methode `equals(Object o)` für die Klasse `Circle`, mit der die Gleichheit von `Circle`-Instanzen geprüft werden soll (d.h. Gleichheit der Radien). Prüfen Sie zunächst mit `instanceof`, ob es sich bei dem aktuellen Parameter tatsächlich um eine Instanz der Klasse `Circle` handelt. (3 Punkte)

```
@Override
public boolean equals(Object o) {
```

```
}
```

Können Sie mit dieser Implementierung auch ein `Circle`-Objekt mit einem `Ellipse`-Object erfolgreich vergleichen? Wenn ja, warum? (2 Punkte)

Was halten Sie von diesem Klassenentwurf? Begründen Sie Ihre Antwort detailliert! (5 Punkte)

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

Gegeben sei das folgende Programmfragment:

```
public class Kommunikation {
      public static void main(String[] args) {
            Handy handy = new Handy();
            Tablet tablet = new Tablet();

            SMS sms = new SMS();
            EMail eMail = new EMail();

            Textnachricht<?> nachricht = sms;
            nachricht = eMail;

            sms.versendeMit(handy);
            eMail.versendeMit(tablet);

            /* [1] */

            Handy smsSender = sms.womitVersendet();      /* [2] */
            Tablet eMailSender = eMail.womitVersendet(); /* [3] */
      }
}

interface Kommunikationsgerät {}
class Handy implements Kommunikationsgerät {}
class Tablet implements Kommunikationsgerät {}
```

Geben Sie eine Implementierung für die Klassen `Textnachricht`, `SMS` und `EMail` an, sodass dieses Programmfragment kompiliert. Die Methoden `versendeMit()` und `womitVersendet()` sollen insgesamt nur ein einziges Mal implementiert werden. (5 Punkte)

(Fortsetzung der Aufgabe auf der folgenden Seite)

(Fortsetzung von Aufgabe 2)

Die Deklaration der Kommunikationsgeräte wird nun wie folgt verändert:

```
interface Kommunikationsgerät {}
interface MitInternet extends Kommunikationsgerät {}
interface MitGsm extends Kommunikationsgerät {}
class Handy implements MitGsm {}
class Tablet implements MitInternet {}
class Smartphone implements MitInternet, MitGsm {}
```

Weiterhin wird an der mit `/* [1] */` markierten Stelle der folgende Code ergänzt:

```
Smartphone smartphone = new Smartphone();
sms.versendeMit(smartphone);
eMail.versendeMit(smartphone);
```

Während diese zwei Aufrufe von `versendeMit(.)` durch den Compiler akzeptiert werden sollen, sollen die Aufrufe von

```
sms.versendeMit(tablet);
eMail.versendeMit(handy);
```

zurückgewiesen werden.

Müssen Sie hierzu Änderungen an Ihren Implementierungen der Klassen `Textnachricht`, `SMS` und `Email` vornehmen? Wenn ja, welche? Wenn nein, warum nicht? (1 Punkt)

Sind die beiden mit `/* [2] */` **und** `/* [3] */` markeirten Stellen nun noch korrekt? Begründen Sie Ihre Antwort und geben Sie gegebenenfalls eine Korrektur an! (4 Punkte)

Aufgabe 3: Fehlerbehandlung (10 Punkte)

In welche Gruppen können die Ausnahmesituationen („Exceptions"), die in einem Programm im Allgemeinen auftreten können, eingeordnet werden? (3 Punkte)

Nennen Sie 2 Ihnen bekannte Ausnahmetypen aus der Java-Standardbibliothek, die aus verschiedenen der oben genannten Gruppen stammen, und ordnen Sie sie zu. (1 Punkt)

(Fortsetzung der Aufgabe auf folgender Seite)

(Fortsetzung von Aufgabe 3)

Es soll eine Klasse in Java implementiert werden, die die Eingabe der PIN und den Versand einer SMS modelliert. Wenn versucht wird, eine SMS zu versenden und zuvor eine ungültige PIN eingegeben wurde, soll eine `NichtVerbundenException` ausgelöst werden. Ergänzen Sie das folgende Programm an den vorgesehenen Stellen um solch eine Ausnahmebehandlung. (6 Punkte)

Hinweis: Nicht an allen Stellen ist eine Ergänzung notwendig. Tragen Sie in Felder, die frei bleiben sollen, die Kennung `/* k. E. n. */` („keine Ergänzung notwendig") ein.

```
public class Kommunikation {
	public static void main(String[] args) {
		Smartphone smartphone = new Smartphone();
		smartphone.entsperre(1234);
```

```
		smartphone.versendeSms("Hallo Welt!");
```

```
	}
}
class Smartphone                                    {
	boolean pinGültig;
	void entsperre(Integer pinCode) {
		pinGültig = ...; /* Aufruf einer externen Bibliothek */
```

```
	}
	public void versendeSms(String nachricht)                    {
```

```
		System.out.println("Nachricht wird versendet.");
	}
}
class NichtVerbundenException                                    {}
```

Aufgabe 4: Polymorphie (10 Punkte)

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

Charakterisieren Sie jede Art kurz! (jeweils 2 Punkte)

Aufgabe 5: Verteilte Systeme (10 Punkte)

Welche Kommunikationsarten von Prozessen kennen Sie? Was zeichnet sie jeweils aus? (2 Punkte)

Nennen Sie vier Kommunikationsmittel und ordnen Sie sie den Kommunikationsarten von oben zu. (2 Punkte)

(Fortsetzung der Aufgabe auf folgender Seite)

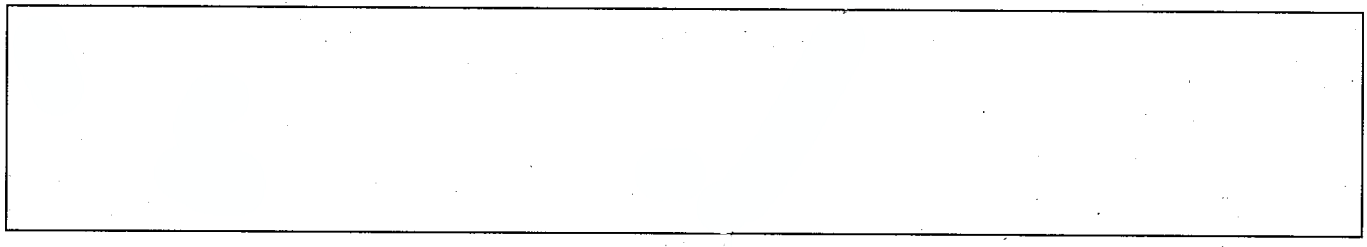
(Fortsetzung von Aufgabe 5)

Vervollständigen Sie die beiden folgenden Klassen, sodass

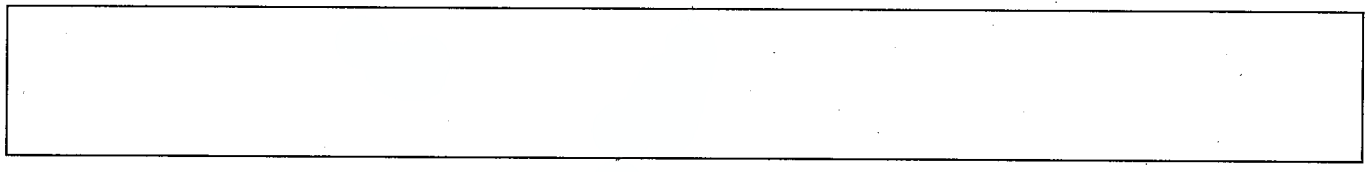
- die Klasse `SimpleServer` (nacheinander) beliebig viele Verbindungen auf Port 4242 annimmt, den Text `Hallo!` überträgt und die Verbindung beendet;
- die Klasse `SimpleClient` eine Verbindung zum Rechner mit dem Namen `localhost` auf Port 4242 aufbaut, eine Zeile ausliest und die Verbindung beendet. (6 Punkte)

```
public class SimpleServer {

    public static void main(String[] args) throws IOException {
```



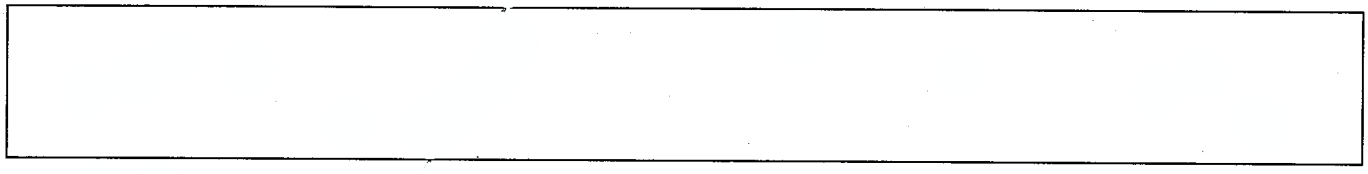
```
            socket.getOutputStream().write("Hallo!".getBytes());
```



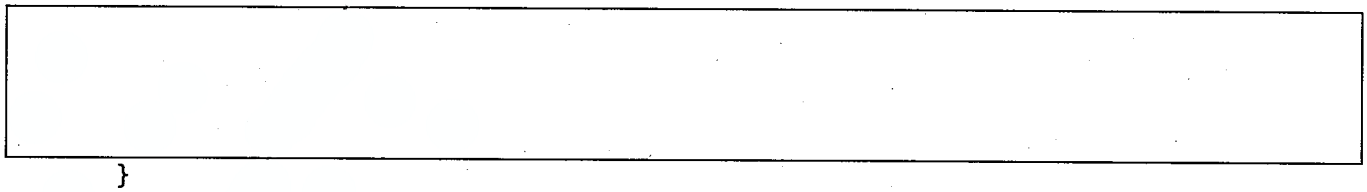
```
    }
}

public class SimpleClient {

    public static void main(String[] args) throws IOException {
```



```
        BufferedReader bufferedReader = new BufferedReader(
                new InputStreamReader(socket.getInputStream()));
        System.out.println(bufferedReader.readLine());
```



```
    }
}
```

Aufgabe 6: Fehlersuche (20 Punkte)

In das folgende Java-Programm haben sich einige Fehler eingeschlichen.

```
1  class SuperTest<T extends Serializable> {
2        T t;
3        final int argLength = -1.0d;
4        void m() {
5              m(null);
6        }
7        void m(String[] args) {
8              if (args != null)
9                    argLength = args.length;
10       }
11       private void n(){
12             Serializable s = this;
13       }
14 }
15
16 public class EinTest extends SuperTest<EinTest> {
17       EinTest() {
18             super(null);
19       }
20       public void m(Object[] args) {
21             super.m(args);
22             super.n();
23             1 + 2;
24       }
25 }
```

Identifizieren Sie acht Fehler anhand Ihrer Zeilennummern und erklären Sie sie.

Fehler (1) (2 Punkte)

Fehler (2) (2 Punkte)

Fehler (3) (2 Punkte)

Fehler (4) (2 Punkte)

(Fortsetzung der Aufgabe auf folgender Seite)

(Fortsetzung von Aufgabe 6)

Fehler (5) (2 Punkte)

Fehler (6) (2 Punkte)

Fehler (7) (2 Punkte)

Fehler (8) (2 Punkte)

Gegeben sei das folgende Programmfragment:

```
public class AndererTest {
      public static void main(String[] args) {
            System.out.println(args[0] + args[args.length]);
      }
}
```

Erklären Sie im Detail, was bei der Ausführung der `main(.)`-Methode geschieht, wenn das Programm mit dem Kommonado

```
java AndererTest Hello
```

gestartet wird. (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.