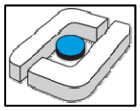
Thema:	Grundlagen Programmierung		
Dozent:	Prof. Dr. Rainer Roosmann	Seitenummer:	Seite 1 von 12
Studiengang:	Informatik - Medieninformatik	Bearbeitungszeit:	120 Minuten



Dies ist eine Probeklausur, die keine formalen Schlüsse auf die Form, die Struktur oder den Inhalt der endgültigen Klausur zulässt. Die Probeklausur wurde ursprünglich von Prof. R. Roosmann herausgegeben.

Es werden 50 Punkte zum Bestehen benötigt.

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**Summe**
15	**6**	**8**	**8**	**12**	**34**	**9**	**8**	**100**
								erreicht P.

Aufgabe 1 – MULTIPLE CHOICE (15 Punkte)

Geben Sie im Folgenden an, ob die Behauptung wahr oder falsch ist. Machen Sie nur ein Kreuz pro Frage. Bei richtiger Antwort gibt es einen Punkt, **bei falscher Antwort wird ein Punkt abgezogen**.

Machen Sie kein Kreuz, bleibt die Punktzahl unverändert. Die minimale Punktzahl für diese Aufgabe beträgt 0 Punkte.

	Wahr	**Falsch**
Alle benutzerdefinierten Klassen haben in Java genau eine Oberklasse.	□	□
In Java erben Klassen und elementare Datentypen von der Klasse `Object`.	□	□
Ein Objekt kennt die instanziierende Klasse und die Klasse kennt die Objekte die von ihr instanziiert wurden.	□	□
Eine Instanz der Klasse String wird als Literal bezeichnet, da diese eine Zeichensequenz (lat. Literal) verwaltet.	□	□
Jede Klasse in Java benötigt mindestens einen Konstruktor (durch den Compiler generiert oder durch Sie programmiert).	□	□
Wird für eine Objektvariable in Java die Sichtbarkeit nicht definiert, so ist diese per Default `public`.	□	□
In Java wird die Identität von Objekten über den == Operator und die Objektgleichheit über die equals-Methode geprüft.	□	□
Der Begriff „Überschreiben“ bezieht sich auf Methoden gleichen Namens aber unterschiedlicher Signatur innerhalb derselben Klasse.	□	□
Folgende Konstruktoren können in der Klasse Mitarbeiter nebeneinander existieren: • `public Mitarbeiter(String vorname, String nachname) {…}` • `public Mitarbeiter(String nachname, String geburtsOrt){…}`	□	□
Eine Klasse kann von beliebig vielen abstrakten Klassen erben (im Sinne einer Ableitung diese spezialisieren).	□	□
Eine Klasse kann beliebig viele Interfaces realisieren.	□	□
Der Konstruktor der Elternklasse wird nicht mitvererbt und muss im Konstruktor der Kindklasse aufgerufen werden (implizit durch den Compiler oder explizit durch Sie).	□	□
Wird das Schlüsselwort `abstract` auf Klassenebene verwendet (z. B. `public abstract class Punkt{…}`), so sind alle Methoden automatisch abstrakt und müssen von spezialisierenden Klassen zwingend überschrieben werden.	□	□
Überschriebene Methoden können unterschiedliche Sichtbarkeiten haben.	□	□
Die dynamische Polymorphie geht einher mit dem Überladen von Methoden und kann zur Compilezeit aufgelöst werden.	□	□
Die Anweisungsfolge `int i[] = new int[10]; i[10]=10;` wird vom Compiler akzeptiert und liefert eine Ausnahme zur Laufzeit.	□	□

Aufgabe 2) Sichtbarkeiten und Packages (6 Punkte)

Es sind vier Implementierungen der Methode `umfang()` einer Klasse `Kreis` gegeben. Welche dieser Implementierungen lassen sich fehlerfrei im Hauptprogramm ausführen, wenn sie in den Quellcode der Klasse Kreis eingefügt werden. Begründen Sie für jede problembehaftete Implementierung in maximal einem Satz – direkt neben der Methode –, warum diese nicht ausgeführt werden kann.

Klasse KreisTest.java:

```
package de.hsos.gp.auswertung
import de.hsos.gp.geometrie.Kreis;

public class TestKreis {
    public static void main(String[] args) {
        Kreis k = new Kreis(5.);
        System.out.println(k.umfang());
    }
}
```

Klasse Kreis.java:

```
package de.hsos.gp.geometrie

public class Kreis {
    protected double r; //Radius
    public Kreis(double radius) {
        this.r = radius;
    }
    /*hier Methoden umfang() 1. bis 4. einfuegen (s.u.)*/


}
```

```
1. public double umfang() {
     return 2*r*3.1416;
   }

2. protected double umfang() {
     return 2*r*3.1416;
   }

3. double umfang() {
     return 2*r*3.1416;
   }

4. public static double umfang() {
     return 2*r*3.1416;
   }
```

Begründung:
Zu 1.:
Zu 2.:
Zu 3.:
Zu 4.:

Erläuterung: Kreisumfang = 2 * Radius * Pi, mit Pi = 3.1416

Aufgabe 3) Ausdrücke auswerten (8 Punkte)

Gegeben seien folgende Variablen.

```
int         i = 164;
Integer    gi = i+36;
Integer   cgi = gi;
String     s1 = null;
String     s2 = "Don't Panic!";
char[]     c = {'M','a','c','h',' ','e','t','!'};
boolean check = true;
int[] ai      = new int[i];
```

Die folgenden Ausdrücke sind Teil einer `System.out.println(einzusetzender Ausdruck)` Anweisung. Schreiben Sie die erwarteten Ergebnisse des Ausdrucks auf.

einzusetzenderAusdruck	**Auswertung**

`123456 + i * (-1) == 42 ||`
`!check ||`
`gi.equals(i)`	

`check && gi == cgi ||` `123456 + i * (-1) == 42`	
`++i + gi++ + " Tage."`	
`-1.0 * --i == 164`	
`s2 + " Du bist ja keine " + s1`	
`s2 + c[c.length-1]`	
`ai[0]`	
`ai[i]`	

Aufgabe 4) Analyse von Verzweigungen (8 Punkte)

Gegeben sei die folgende Klasse mit den entsprechenden Methoden. Geben Sie die Werte der Objektvariablen this.e und this.f sowie ggf. den Rückgabewert in der Tabelle auf der Folgeseite an, nachdem ein Objekt der Klasse neu erzeugt und die genannte Methode mit den definierten Parametern ausgeführt wurde (für jeden Methodenaufruf gleicher Objektzustand).

```
public class Aufgabe_4 {
    private int e = 3;
    private int f = 2;
    public void m1(boolean a, boolean b) {
        if(a) {
            e = e * f;
        } else {
            f *= f;
        }
    }

    public void m2(boolean a, boolean b) {
        if(a || b) {
            if(a && b) {
                e = f * (int)Math.PI; //PI=3.1416...
            } else if(a) {
                e = f;
            } else {
                f = e * f + f % 2;
            }
        } else {
            m1(a, b);
        }
    }

    public boolean m3(boolean a, boolean b) {
        if(!a) {
            if(!b) {
                e *= ++f;
                f += e;
                return true;
            }
            f = 3;
        } else {
            if(!b) {
                f = e % 3;
            }
        }
        return false;
    }

    public int m4(boolean a, boolean b) {
        int erg = 0;
        if(!a && e % 2 != 0) {
            erg = e;
        }
        if(a && b) {
            tausche();
            erg = e * f;
        }
        if(!b && f % 2 == 0) {
            erg += f;
        }
        return erg;
    }
… //Fortfuehrung auf der Folgeseite
```

```
…
    private void tausche() {                    //
        int tmp = this.e;
        this.e = this.f;
        this.f = tmp;
    }
}
```

Lösung (ist hier einzutragen):

a	b	m1(a,b)			m2(a,b)			m3(a,b)			m4(a,b)		
		e	f	Rück-gabe	e	f	Rück-gabe	e	f	Rück-gabe	e	f	Rück-gabe
true	true												
true	false												
false	true												
false	false												

Eigene Notizen (werden nicht berücksichtigt):

Aufgabe 5) Einfache Klasse, Konstruktoren und Methodenaufrufe (12 Punkte)
Gegeben sei die folgende Klasse. Schreiben Sie auf, welche Ausgaben erzeugt werden, wenn das Programm gestartet wird. (Ergebnisse auf der Folgeseite notieren).

```
public class Angestellter {
    private static Integer zaehler = 0;
    private final static double maxProvision = 20.0; // Prozent
    private Integer id;
    private double gehalt = 1000.0; // Euro
    private double provision = 0.; // Prozent
    private boolean provisionAuszahlen = false;

    public Angestellter() {
        this.id = ++Angestellter.zaehler;
    }
    public Angestellter(double gehalt, boolean b) {
        this();
        this.gehalt = gehalt;
        this.provisionAuszahlen = b;
    }
    public double lohnAuszahlung() {
        if (this.provisionAuszahlen == false) {
            return this.gehalt;
        }
        return this.gehalt + this.gehalt * this.provision / 100;
    }
    public double lohnAuszahlung(double provision) {
        this.provisionAuszahlen = true;
        if (provision < maxProvision) {
            this.provision = provision;
        } else {
            this.provision = maxProvision;
        }
        return this.lohnAuszahlung();
    }
    public static Integer erhoeheZaehler() {
        return ++Angestellter.zaehler;
    }
    public String toString() {
        String erg = this.id + ": " + this.lohnAuszahlung();
        if (this.provisionAuszahlen == true) {
            erg += " inkl. " + this.provision + " % Provision.";
        }
        return erg;
    }

    public static void main(String[] args) {
        Angestellter m1 = new Angestellter();
        System.out.println(m1);
        Angestellter.erhoeheZaehler();
        Angestellter m2 = new Angestellter(2000.0, true);
        m2.lohnAuszahlung();
        System.out.println(m2);
        m2.lohnAuszahlung(30.0);
        System.out.println(m2);
        Angestellter m3 = new Angestellter(m2.lohnAuszahlung(300), false);
        System.out.println(m3);
        Angestellter m4 = m1;
        m4.lohnAuszahlung(10.0);
        System.out.println(m4);
        System.out.println(m1);
    }
}
```

Lösung Aufgabe 5:

Ausgabe 1:	
Ausgabe 2:	
Ausgabe 3:	
Ausgabe 4:	
Ausgabe 5:	
Ausgabe 6:	

Eigene Notizen (werden nicht berücksichtigt):

Aufgabe 6) Programmierung mit Sammlungen und Wiederholungen (6+2+4+6+4+4+4+4 = 34 Punkte)

Gegeben sei das folgende Grundgerüst der Klasse `MitarbeiterVerwaltung` zur Verwaltung von Objekten der Klasse `Mitarbeiter`.

```
public class MitarbeiterVerwaltung {
    private Mitarbeiter[] mitarbeiter;
    private int anzahl; //Anzahl verwalteter Mitarbeiter

    public MitarbeiterVerwaltung(int kapazitaet) {…}
    private boolean existiertMitarbeiter(Mitarbeiter m) {…}
    public boolean hinzufuegen(Mitarbeiter m) {…}
    public Mitarbeiter entfernen() {…}
    public int durchschnittsalter(int aktuellesJahr) {…}
    @Override
    public String toString() {…}
}
```

Aufgabe:

a) Definieren Sie die Klasse `Mitarbeiter` mit den Objektvariablen `String name` und `int jahr` (=> Geburtsjahr). Schreiben Sie einen überladenen Konstruktor (Übergabeparameter: `String name und int jahr`), sämtliche getter- und setter-Methoden und überladen die toString-Methode (Ausgabe: "Franz, 2011").

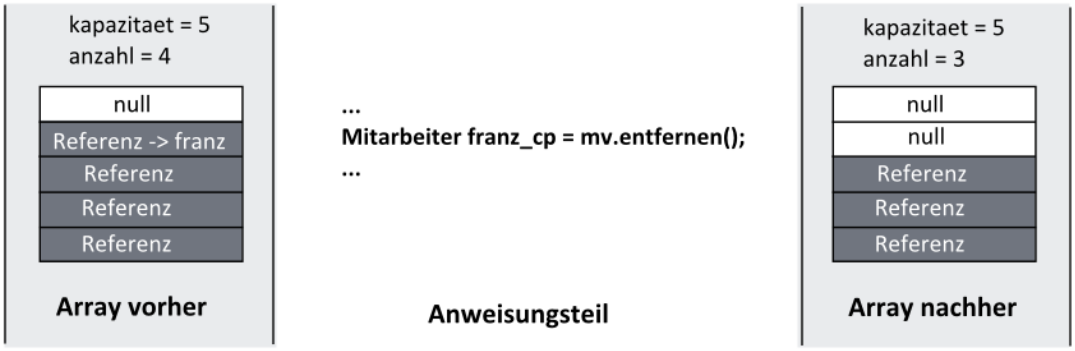
b) Schreiben Sie für die Klasse `MitarbeiterVerwaltung` den überladenen Konstruktor. Der Übergabeparameter `kapazitaet` definiert die Array-Größe. Die Objektvariable `anzahl` zeigt die Anzahl der tatsächlich verwalteten Elemente und ist initial 0.

c) Definieren Sie die private Hilfsmethode `boolean existiertMitarbeiter(…)`. Zur Prüfung auf Gleichheit zweier Mitarbeiter-Objekte reicht es, die Namen zu vergleichen. Vorgabe: Verwenden Sie eine while-Schleife!

d) Als Alternative zur Prüfung auf Gleichheit kann die Klasse `Mitarbeiter` um die equals-Methode erweitert werden. Was passiert, wenn die equals-Methode verwendet, aber nicht überschrieben wird? Erläutern Sie. Schreiben Sie zudem eine sinnvolle Realisierung der equals-Methode für die Klasse `Mitarbeiter` auf.

e) Definieren Sie die Methode `boolean hinzufuegen(…)` um ein Mitarbeiter-Objekt an der ersten freien Position des Arrays hinzuzufügen. Mitarbeiter sollen nicht doppelt vorkommen (Prüfung über: `existiertMitarbeiter(…)`). Es ist sicherzustellen, dass die Array-Grenze nicht überschritten wird. Wird ein Objekt erfolgreich dem Array hinzugefügt, wird `true` zurückgegeben, ansonsten `false`.

f) Schreiben Sie die Objektmethode `Mitarbeiter entfernen( )` über die das letzte Mitarbeiter-Objekt des Arrays zurückgegeben und aus dem Array entfernt wird. Das Entfernen eines Objektes beinhaltet u.a., dass das entsprechende Array-Element auf `null` gesetzt wird.
Achten Sie auf die Array-Grenzen. Ist das Array leer, soll `null` zurückgegeben werden.



g) Schreiben Sie die Methode `int durchschnittsalter(int aktuelles Jahr)`. Das Durchschnittsalter berechnet sich über das aktuelle Jahr abzüglich des arithmetischen Mittels der Geburtsjahre sämtlicher Nutzer und wird als int-Wert zurückgegeben, wobei das aktuelle Jahr als Parameter übergeben wird.
Vorgabe: Verwenden Sie eine for-Schleife!

h) Überschreiben Sie die toString-Methode. Die Mitarbeiter sollen über die Methode zeilenweise ausgegeben werden können.
Vorgabe: Nutzen Sie eine for-each Schleife!

Aufgabe 7) Polymorphie (9 Punkte)

Gegeben seien die folgenden vier Klassen.

```
public class K1 {
	public void m1(String s) {
		m3(s + " K1m1 " );
	}
	public void m2(String s) {
		System.out.println(s +" K1m2 ");
	}
	public void m3(String s) {
		m2(s +" K1m3 ");
	}
}

public class K2 extends K1 {
	@Override
	public void m1(String s) {
		m3(s +" K2m1 ");
	}
	@Override
	public void m2(String s) {
		System.out.println(s+" K2m2 ");
	}
	@Override
	public void m3(String s) {
		m2(s+" K2m3 ");
	}
}

public class K3 extends K2 {
	@Override
	public void m2(String s) {
		((K2) this).m1(s +" K3m2 ");
	}
	@Override
	public void m3(String s) {
		System.out.println(s +" K3m3 "  );
	}
}

public class K4 extends K3 {

	public void m4(String s) {
		K3 k = (K3) this;
		k.m2(s +" K4m4 ");
	}
}
```

In einem anderen Programm werden folgende Objekte erzeugt.

```
K1 k1 = new K1();
K1 k2 = new K2();
K1 k3 = new K3();
K4 k4 = new K4();
```

Welche Ausgabe liefern die folgenden einzelnen Zeilen, wenn sie jeweils nach den obigen Zeilen ausgeführt werden (falls sie keine Ausgabe angeben können, beschreiben Sie, was passiert)?

	Aufruf	Ergebnis
a)	`k1.m1("A");`	
b)	`k2.m1("D");`	
c)	`k2.m2("E");`	
d)	`k2.m3("F");`	
e)	`k3.m2("H");`	
f)	`k3.m3("I");`	
g)	`k4.m2("K");`	
h)	`k4.m3("L");`	
i)	`k4.m4("M");`	

Aufgabe 8) Zwei Fragen zur Programmierung (4 + 4 = 8 Punkte)

a) Erläutern Sie die Begriffe „statische Polymorphie“ und „dynamische Polymorphie“. Schreiben Sie zur Erläuterung Beispiel-Quellcode für das Überladen und Überschreiben von Methoden.

b) In Java lässt sich das Schlüsselwort `static` auf verschiedenen Ebenen einsetzen. Nennen und erläutern Sie diese. Verwenden Sie zur Erläuterung sinnvolle Beispiele (Quellcode).