Thema:	Programmierung 1			
Dozent:	Prof. Dr. Stephan Kleuker		Seitennummer:	Seite 1 von 12
Studiengang:	Informatik – Technische Informatik		Studiensemester:	1
Datum:	<eine Beispielklausur>		Bearbeitungszeit:	120 Minuten
Matrikelnummer:	1234567	Name:	ich	

Dies ist eine Probeklausur, die keine formalen Schlüsse auf die Form, die Struktur oder den Inhalt der endgültigen Klausur zulässt.

Diese Klausur ist Bestandteil der Bachelor-Prüfung im Sinn der Prüfungsordnung.

Zugelassene Hilfsmittel: keine.

Überprüfen Sie, dass Ihre Arbeit aus 12 Blättern besteht. Schreiben Sie oben auf die Klausur Ihren Namen und Ihre Matrikelnummer. Es ist nicht erlaubt, die Heftung der Klausur aufzutrennen. Schreiben Sie nur mit einem dokumentenechten Stift (kein Bleistift!) nicht in roter Farbe. Rückseiten können beschrieben werden.
Während der Klausur können keine Fragen gestellt werden, falls Sie meinen, dass eine Aufgabenstellung unklar ist, dokumentieren Sie Ihre Annahmen.

Es werden 50 Punkte zum Bestehen benötigt.

Aufgabe 1	6 **/6**	
Aufgabe 2	4 **/6**	
Aufgabe 3	9 **/10**	
Aufgabe 4	58 **/60**	
Aufgabe 5	6 **/6**	
Aufgabe 6	6 **/6**	
Aufgabe 7	6 **/6**	**Note:**
Gesamt	95 **/100**	1,0

Mit meiner Unterschrift bestätige ich, dass ich die Klausur alleine, nur unter Nutzung der zugelassenen Hilfsmittel, bearbeitet habe.

F. XYZ

Unterschrift

Aufgabe 1) Ausdrücke auswerten (6 Punkte)

Gegeben seien folgende Variablen.

```
int x = 42;
double d = 42.42;
String s = "Hai";
int z = 0;
```

Schreiben Sie auf, was die Auswertung der folgenden Ausdrücke ergibt.

Ausdruck	Auswertung
`x + 1`	43
`s + 1`	"Hai1"
`z + z + s`	"0Hai"
`x * d * d > d * x * x`	true
`s.equals(null)`	false
`(z + 1) / (1 + z)`	1

6

Aufgabe 2) Analyse von if (6 Punkte)

Gegeben sei die folgende Klasse mit den angegebenen Methoden. Geben Sie die Werte der Objektvariablen this.x und this.y an, nachdem jeweils ein Objekt der Klasse erzeugt und die genannte Methode jeweils ausgeführt wurde.

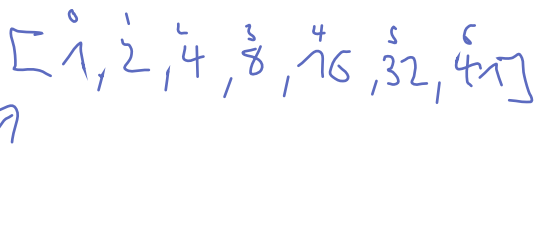
```
public class Aufgabe02{
  private int x = 6;
  private int y = 7;

  public void m1(boolean a, boolean b){
    if(a){
      this.x = this.y;
      this.y = this.x;
    }
    if(a){
      this.x = this.y;
      this.y = this.x;
    }
  }

  public void m2(boolean a, boolean b){
    if(a){
      if (a && b) {
        this.x= this.x * this.y;
      } else {
        if (a || b) {
          this.y = this.x * this.y;
        } else {
          this.y = 0;
        }
      }
  }
  public void m3(boolean a, boolean b){
    if(a){
        this.y = this.y + 1;
        a = !a;
    }
    if(b){
        this.x = this.x - 1;
        b = !b;
    }
    if( a || b){
        this.x = 42 * 42;
    }
  }

}
```

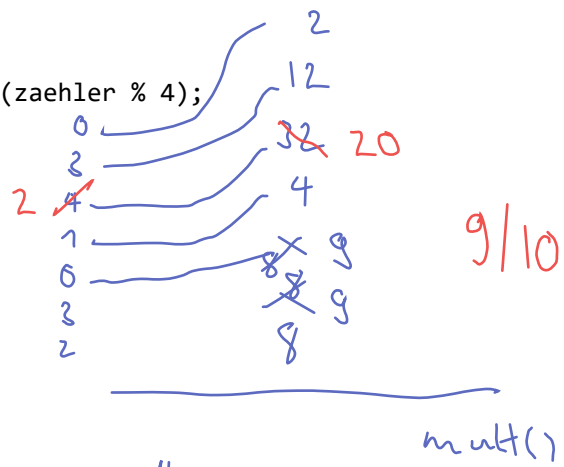
a	b	m1(a,b)		m2(a,b)		m3(a,b)	
		x	y	x	y	x	y
true	true	7	7	42	7	5	8
true	false	7	7	6	~~7~~ 42	6	8
false	true	6	7	6	~~42~~ 7	5	7
false	false	6	7	6	~~0~~ 7	6	7

4/6

Aufgabe 3) Analyse von Schleifen (2+2+3+3 = 10 Punkte)
Gegeben sei die folgende Klasse. Schreiben Sie auf, welche Ausgaben in welcher Reihenfolge ausgegeben werden, wenn ein Objekt der Klasse L (`L l = new L();`) erzeugt und die Methode doIt() aufgerufen wird (`l.doIt();`). (gerne neben das Programm, bei einer ArrayList werden alle Elemente innerhalb von eckigen Klammern ausgegeben).
Hinweis: Die erste ausgegebene Liste enthält 7 Werte.

```
import java.util.ArrayList;

public class L{
    private ArrayList<Integer> z = new ArrayList<Integer>();
    private EinUndAusgabe io = new EinUndAusgabe();

    public L(){
        int j = 2;
        for(int i = 1; j <= 7; i = i + i){
            this.z.add(i);
            j = j + 1;
        }
        this.z.add(41);
        this.io.ausgeben(this.z + "\n");
    }

    public Integer drei(){
        int ergebnis = 0;
        int zaehler = 0;
        while(zaehler < this.z.size() -1 ){
            if(z.get(zaehler + 1) - z.get(zaehler) > 3){
                io.ausgeben(z.get(zaehler)+"\n");
            }
            zaehler = zaehler + 1;
        }
        return ergebnis;
    }

    public void sum(){
      int zaehler = 0;
        while(zaehler < this.z.size() * 7){
            Integer tmp = z.get(zaehler % 5) + z.get(zaehler % 4);
            io.ausgeben(tmp+"\n");
            zaehler = zaehler + 7;
        }
    }

    public void mult(){
      for(int i: this.z) {
        for(int j = 1; j % 2 != 0; j = j + 1){
          int tmp = i * z.get(z.get(j) % 5);
          io.ausgeben(tmp+"\n");
        }
      }
    }

  public void doIt(){
    L x= new L();
    x.drei();
    x.sum();
    x.mult();
  }
}
```





Aufgabe 4) Programmierung mit Sammlungen (2+2+2+7+9+14+20+4 = 60 Punkte)

Gegeben sei die Klasse Hund mit den dort angegebenen üblichen ausimplementierten Methoden und die folgende Klasse Hundeliste. Ergänzen Sie die geforderten Objektmethoden für Hundeliste. Sie dürfen vereinfachend davon ausgehen, dass in der Sammlung hunde keine null-Werte stehen. Methodenparameter mit Klassen als Typen und hunde selbst können null sein. Für ArrayList<Hund>-Objekte dürfen nur die Methoden get(.), add(.) und size() genutzt werden. Sie dürfen eigene Hilfsmethoden schreiben.

```
public class Hund {
  private String name;
  private int groesse;
  public Hund(String n, int gr){ …
  public String getName() {…
  public int getGroesse() {…
  public void setName( String n) {…
  public void setGroesse(int gr) {…
  public boolean equals(Object o){…
}
```

```
public class Hundeliste {
  private ArrayList<Hund> hunde = new ArrayList<Hund>();
}
```

a) Schreiben Sie für alle Objektvariablen get und set-Methoden.
b) Schreiben Sie eine Methode, die einen Hund übergeben bekommt und diesen in die Sammlung einfügt.
c) Schreiben Sie eine Methode, die einen String- und einen int-Wert übergeben bekommt, daraus einen Hund konstruiert und diesen in die Sammlung einfügt.
d) Schreiben Sie eine Methode, die jeden Hund um 3 größer macht.
e) Schreiben Sie eine Methode, die überprüft, ob alle Hunde der Größe nach angeordnet sind, also die Liste mit dem kleinsten Hund beginnt und alle nachfolgenden Hunde immer nicht kleiner sind, also Größen der Form [3, 17, 17, 18, 34] wären ok, gibt es keine Liste ist das auch ok.
f) Schreiben Sie eine Methode, die überprüft ob es zwei Hunde in der Liste gibt, die den gleichen Namen, aber eine unterschiedliche Größe haben.
g) Schreiben Sie eine Methode die eine Liste von Hundelisten übergeben bekommt und genau dann true als Ergebnis liefert, wenn jeder Hund aus this.Hunde in mindestens einer der übergebenen Listen vorkommt (equals nutzen).
h) Schreiben Sie zwei JUnit-Tests mit denen Sie Ihre Methode aus f) einmal mit dem Ergebnis true und einmal mit dem Ergebnis false auf Korrektheit prüfen können (Hinweis: Die Aufgabe ist auch ohne eine vollständige Lösung zu f) bearbeitbar, imports müssen nicht angegeben werden).

a)
```
public ArrayList<Hund> getHunde() {
        return this.hunde;
}
public void set(ArrayList<Hund> h) {
        this.hunde = h;
}
```

b)
```
public void add(Hund h) {
        this.hunde.add(h);
}
```

c)
```
public void add(String n, int g){
        this.hunde.add(new Hund(n,g));
}
```

6

(Wiederholung)

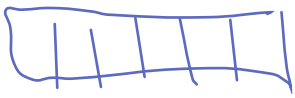
a) Schreiben Sie für alle Objektvariablen get und set-Methoden.
b) Schreiben Sie eine Methode, die einen Hund übergeben bekommt und diesen in die Sammlung einfügt.
c) Schreiben Sie eine Methode, die einen String- und einen int-Wert übergeben bekommt, daraus einen Hund konstruiert und diesen in die Sammlung einfügt.
d) Schreiben Sie eine Methode, die jeden Hund um 3 größer macht.
e) Schreiben Sie eine Methode, die überprüft, ob alle Hunde der Größe nach angeordnet sind, also die Liste mit dem kleinsten Hund beginnt und alle nachfolgenden Hunde immer nicht kleiner sind, also Größen der Form [3, 17, 17, 18, 34] wären ok, gibt es keine Liste ist das auch ok.
f) Schreiben Sie eine Methode, die überprüft ob es zwei Hunde in der Liste gibt, die den gleichen Namen, aber eine unterschiedliche Größe haben.
g) Schreiben Sie eine Methode die eine Liste von Hundelisten übergeben bekommt und genau dann true als Ergebnis liefert, wenn jeder Hund aus this.Hunde in mindestens einer der übergebenen Listen vorkommt (equals nutzen).
h) Schreiben Sie zwei JUnit-Tests mit denen Sie Ihre Methode aus f) einmal mit dem Ergebnis true und einmal mit dem Ergebnis false auf Korrektheit prüfen können (Hinweis: Die Aufgabe ist auch ohne eine vollständige Lösung zu f) bearbeitbar, imports müssen nicht angegeben werden).

d)
```
public void dreiGroesser() {
    for (Hund h : this.hunde) {
        h.setGroesse(h.getGroesse() + 3);
    }
}
```

if (this.hunde == null) { return; }

NullPointer möglich 5/7

e)
```
public boolean geordnet() {
    for(int i=0; i<this.hunde.size()-1; i++){
        if(this.hunde.get(i+1) >= this.hunde.get(i)) {
            return false;
        }
    }
    return true;
}
```



```
if(this.hunde == null){
    return true;
}
```

9

f)
```
public boolean gleicherNameUnterschiedlichGross() {
    if (this.hunde == null) {
        return false;
    }
    for (Hund h : this.hunde) {
        for (Hund h2 : this.hunde) {
            if (h.getName().equals(h2.getName()) && h.getGroesse() != h2.getGroesse()) {
                return true;
            }
        }
    }
    return false;
}
```

(viele Varianten denkbar)

14

(Wiederholung)

a) Schreiben Sie für alle Objektvariablen get und set-Methoden.
b) Schreiben Sie eine Methode, die einen Hund übergeben bekommt und diesen in die Sammlung einfügt.
c) Schreiben Sie eine Methode, die einen String- und einen int-Wert übergeben bekommt, daraus einen Hund konstruiert und diesen in die Sammlung einfügt.
d) Schreiben Sie eine Methode, die jeden Hund um 3 größer macht.
i) Schreiben Sie eine Methode, die überprüft, ob alle Hunde der Größe nach angeordnet sind, also die Liste mit dem kleinsten Hund beginnt und alle nachfolgenden Hunde immer nicht kleiner sind, also Größen der Form [3, 17, 17, 18, 34] wären ok, gibt es keine Liste ist das auch ok.
e) Schreiben Sie eine Methode, die überprüft ob es zwei Hunde in der Liste gibt, die den gleichen Namen, aber eine unterschiedliche Größe haben.
f) Schreiben Sie eine Methode die eine Liste von Hundelisten übergeben bekommt und genau dann true als Ergebnis liefert, wenn jeder Hund aus this.Hunde in mindestens einer der übergebenen Listen vorkommt (equals nutzen).
g) Schreiben Sie zwei JUnit-Tests mit denen Sie Ihre Methode aus f) einmal mit dem Ergebnis true und einmal mit dem Ergebnis false auf Korrektheit prüfen können (Hinweis: Die Aufgabe ist auch ohne eine vollständige Lösung zu f) bearbeitbar, imports müssen nicht angegeben werden).

g)
```
public boolean kommenAlleVor (ArrayList<Hundeliste> liste) {
    if (this.hunde == null){
        return true;
    }
    for (Hund h: this.hunde) {
        boolean gefunden = false;
        for (Hundeliste hl : liste ) {
            for (Hund h2 : hl.getHunde()) {
                if (h.equals(h2)) {
                    gefunden = true;
                }
            }
        }
        if (! gefunden) {
            return false;
        }
    }
    return true;
}
```

20

(Wiederholung)

a) Schreiben Sie für alle Objektvariablen get und set-Methoden.
b) Schreiben Sie eine Methode, die einen Hund übergeben bekommt und diesen in die Sammlung einfügt.
c) Schreiben Sie eine Methode, die einen String- und einen int-Wert übergeben bekommt, daraus einen Hund konstruiert und diesen in die Sammlung einfügt.
d) Schreiben Sie eine Methode, die jeden Hund um 3 größer macht.
j) Schreiben Sie eine Methode, die überprüft, ob alle Hunde der Größe nach angeordnet sind, also die Liste mit dem kleinsten Hund beginnt und alle nachfolgenden Hunde immer nicht kleiner sind, also Größen der Form [3, 17, 17, 18, 34] wären ok, gibt es keine Liste ist das auch ok.
e) Schreiben Sie eine Methode, die überprüft ob es zwei Hunde in der Liste gibt, die den gleichen Namen, aber eine unterschiedliche Größe haben.
f) Schreiben Sie eine Methode die eine Liste von Hundelisten übergeben bekommt und genau dann true als Ergebnis liefert, wenn jeder Hund aus this.Hunde in mindestens einer der übergebenen Listen vorkommt (equals nutzen).
g) Schreiben Sie zwei JUnit-Tests mit denen Sie Ihre Methode aus f) einmal mit dem Ergebnis true und einmal mit dem Ergebnis false auf Korrektheit prüfen können (Hinweis: Die Aufgabe ist auch ohne eine vollständige Lösung zu f) bearbeitbar, imports müssen nicht angegeben werden).

h)

```
@Test
public void testfalse() {
    Hundeliste hl = new Hundeliste();
    Assertions.assertfalse(hl.gleicheHundeUnterschiedlichGross());
}

@Test
public void testtrue() {
    Hundeliste hl = new Hundeliste();
    hl.add("X", 42);
    hl.add("X", 47);
    Assertions.assertTrue(hl.gleicheHundeUnterschiedlichGross());
}
```

4

Aufgabe 5) Polymorphie (6 Punkte)

Gegeben seien die folgenden drei Klassen.

```
public class K1{
  protected EinUndAusgabe io
      = new EinUndAusgabe();

  public void m1(String s){
    m2(s);
  }

  public void m2(String s){
    m3(s);
  }

  public void m3(String s){
    this.io.ausgeben(s+"1\n");
  }
}
```

```
public class K2 extends K1{

  @Override
  public void m1(String s){
    this.m1(s);
  }

  @Override
  public void m2(String s){
    super.m3(s);
  }

  @Override
  public void m3(String s){
    this.io.ausgeben (s + "2\n");
  }
}
```

```
public class K3 extends K2{

  @Override
  public void m1(String s){
    super.m3(s);
  }

  @Override
  public void m3(String s){
    super.m1(s);
  }
}
```

In einem anderen Programm werden folgende zwei Objekte erzeugt.

```
K1 k2 =new K2();
K1 k3 = new K3();
```

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
`k2.m1("D");`	Fehler: m1 ruft sich selbst immer wieder auf
`k2.m2("E");`	E1
`k2.m3("F");`	F2
`k3.m1("G");`	G2
`k3.m2("H");`	H1
`k3.m3("I");`	I2

6

Aufgabe 6) Frage zur Programmierung (6 Punkte)
Erklären Sie anhand eines kleinen Beispiels die Unterschiede und Zusammenhänge der Begriffe Objektvariable, Objektmethode, Klassenvariable, Klassenmethode.

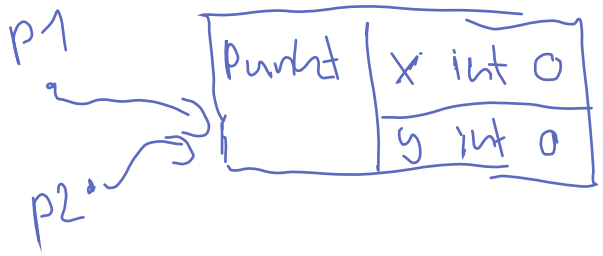
```
public class Blubb {

  private int objektvariable; // in jedem Objekt individuell vorhanden

  private static int klassenvariable; // nur eine Variable für alle Objekte; Klasseneigenschaft


  public void objektmethode() {
     // kann auf eigene Objektvariablen, Klassenvariablen
     // und Klassenmethoden zugreifen
  }

  public static void klassenmethode() {
     // kann nur auf Klassenvariablen und Klassenmethoden
     // zugreifen
  }
```

6

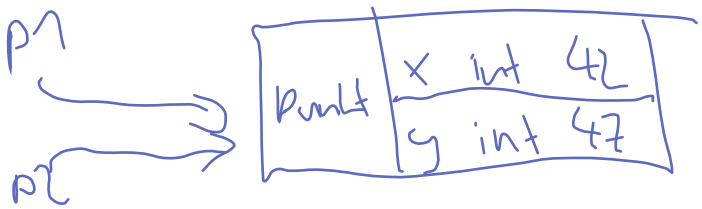
Aufgabe 7) Frage zur Programmierung (6 Punkte)
Java arbeitet mit Objektreferenzen, erklären Sie dies anschaulich mit einem Beispiel. Beschreiben Sie zum Beispiel, wie zwei Variablen das gleiche Objekt referenzieren können und welche Auswirkungen das haben kann.

gegeben eigene Klasse Punkt mit Methode setX(), setY(),

```
Punkt p1 = new Punkt();
// p1 ist Objektreferenz, darüber Objekt bearbeitbar
Punkt p2 = p1;
// p1 und p2 referenzieren gleiches Objekt
```




```
p1.setX(42);    // über beide Referenzen wird
p2.setY(47);    // dasselbe Objekt bearbeitet.
```




```
// p1 == p2 bleiben identisch
```

6

(kann abgetrennt werden, dann aber nicht für Lösungen verwendbar; muss immer abgegeben werden)