Studiengänge: Elektro- und Informationstechnik, Mechatronik, Informatik, Flug- und Fahrzeuginformatik, Wirtschaftsinformatik

# Prüfung Grundlagen der Programmierung 2 Objektorientierte Programmierung Softwareentwicklung 2

Prüfer: Prof. Glavina, Prof. Hahndel, Prof. Schmidt, Prof. Windisch

Zeit: 90 Minuten

Hilfsmittel: keine

		Matrikel-				
Studiengang	Dozent	nummer	Semester	Raum	Platz	

(Bitte sorgfältig schreiben!)

Aufgabe	1	2	3	4	5	6	Σ	Note
Punkte								

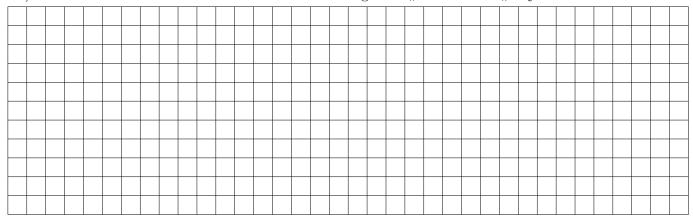
Schreiben Sie Ihre Antworten direkt in die dafür vorgesehenen Stellen der Seiten dieser Prüfungsangabe. Die Angabe besteht (inkl. Deckblatt) aus 14 (+2) Seiten. (**Wichtig:** Die beiden letzten Seiten dürfen Sie zu Ihrer Arbeitserleichterung abtrennen damit Sie bei den Aufgaben 4 und 6 nicht ständig umblättern müssen!)

Bitte geben Sie die ersten 14 Seiten wieder ab, auch wenn einzelne Seiten nicht beschrieben sein sollten. Tragen Sie bitte SOFORT Ihre persönlichen Angaben sowie den Dozenten ein (oben, vor diesem Absatz).

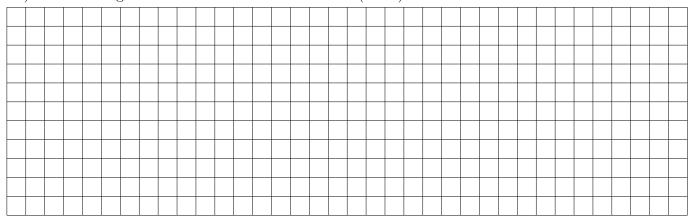
### Viel Erfolg!

# Aufgabe 1: (Allgemeine Fragen, ca. 20%)

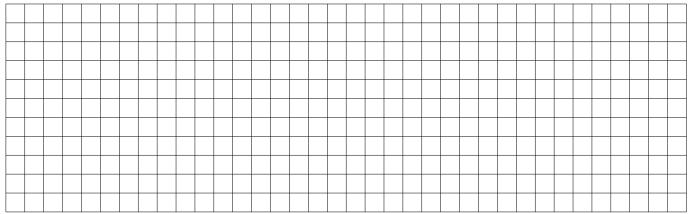
a) Erläutern Sie den Unterschied zwischen den Begriffen "Klasse" und "Objekt".



b) Welche Aufgabe hat die Java Virtual Machine (JVM)?

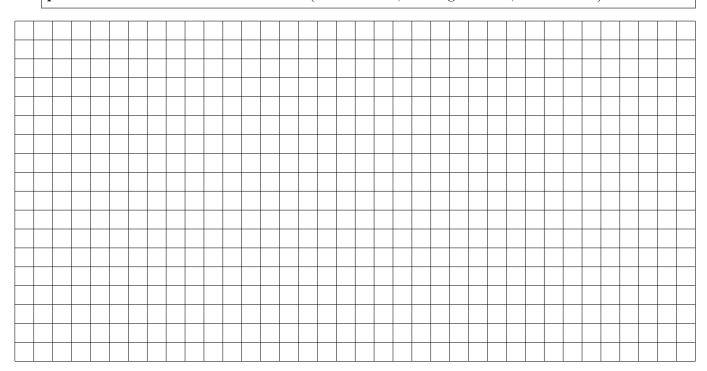


c) Wie erfolgt in Java die Freigabe des Speicherplatzes für ein Objekt ?

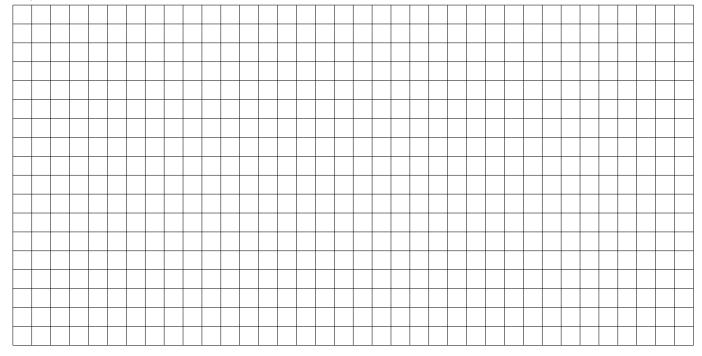


d) Erläutern Sie folgenden Methodenkopf (Inhalt und Funktion der Methode sind ohne Belang) und geben Sie auch zu jedem der Parameter an, ob die Übergabe per "call by value" oder "call by reference" erfolgt:

public static int zaehlePunkte(Student s, Integer mat, double n)

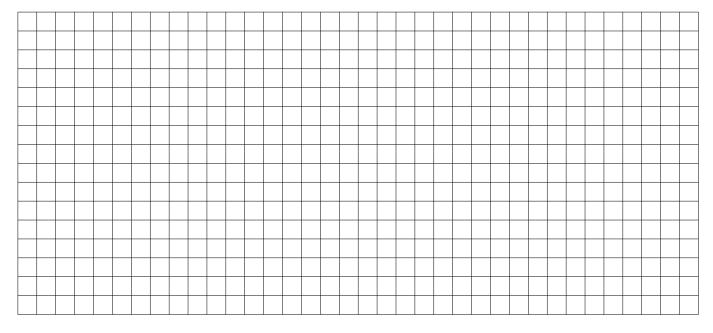


e) Beschreiben Sie kurz die Unterschiede zwischen Arrays in Java und C.



f) Welche Ausgabe erzeugt das folgende Programm:

```
class Abstand {
        private double abstand;
        Abstand (double a) {
                abstand=a;
                System.out.println("Abstand..." + a + "..erzeugt!");
        Abstand (double x1, double y1, double x2, double y2) {
                this (Math. sqrt((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1)));
                System.out.println("Abstand_berechnet!");
        }
        void setAbstand(double a) { abstand = a;
        public String () { return "" + abstand + "..."; }
class Variablen {
        public static void main(String args[]){
                int x, y = 42;
                String r = "Sommer", s = "Winterreifen", t;
                Abstand d1, d2, d3;
                x = y;
                y = 2*x;
                x++;
                System.out.println(x + ", " + y);
                t = s;
                s = r + "traum";
                System.out.println(t + ", \_" + s);
                d1 = new Abstand(1.0);
                d2 = new Abstand(0,0,2,0);
                d3 = d1;
                d1.setAbstand(2.0);
                System.out.println("" + d1 + d2 + d3);
                System.out.println((d1=d2) + ", " + (d1=d3));
        }
}
```



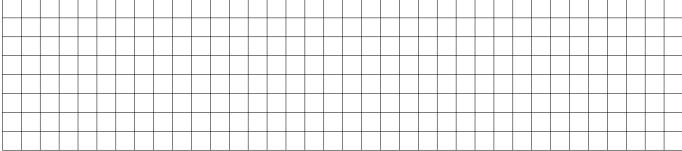
## Aufgabe 2: (Klassen, ca. 15%)

a) Gegeben seien folgende Klassendeklarationen und Objektinstantiierungen:

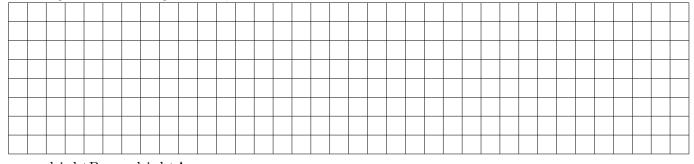
```
class KlasseA { int a; }
class KlasseB extends KlasseA { int b; }
class KlasseC extends KlasseB { int c; }
KlasseA objektA = new KlasseA();
KlasseB objektB = new KlasseB();
KlasseC objektC = new KlasseC();
```

Bewerten Sie die folgenden Codezeilen hinsichtlich ihrer Zulässigkeit und geben Sie eine kurze Erläuterung

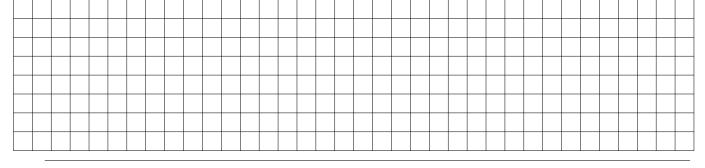
objektA = objektC;



objektB.a = objektA.a;



objektB = objektA;

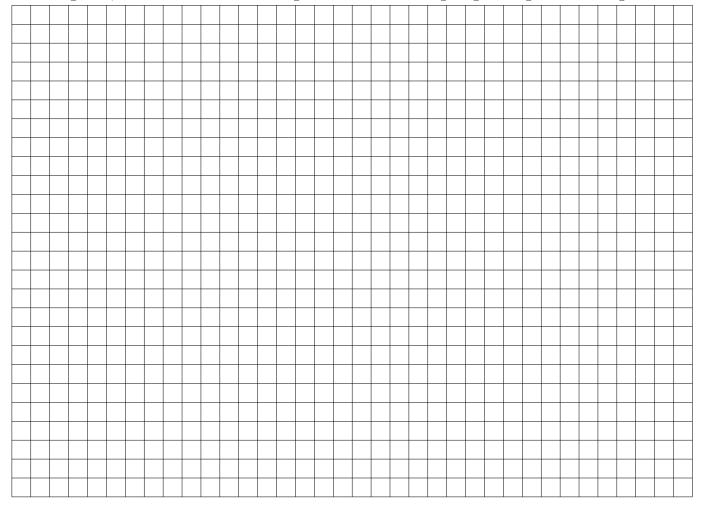


b) Gegeben sei die folgende Klasse UseVektor2D:

Das Programm produziert die folgende Ausgabe:

Ausgabe von v1: (5.0,7.0) Ausgabe von v2: (7.0,2.0) 5 + v1 ergibt: (10.0,12.0) v1 + v2 ergibt: (12.0,9.0)

Schreiben Sie die Klasse Vektor2D mit allen notwendigen Variablen und Methoden, die dazu nötig sind, um UseVektor2D lauffähig zu machen und die gezeigte Ausgabe zu erzeugen.

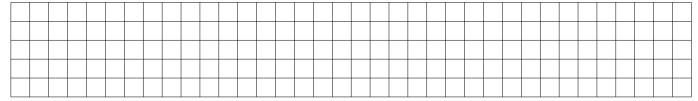


## Aufgabe 3: (Interfaces, ca. 10%)

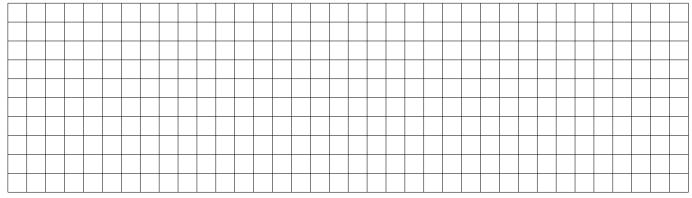
Gegeben sind die beiden folgenden Interface-Definitionen:

```
interface Kaeuflich {
    double getPreis(); // in EUR
}
interface Wiegbar {
    double getGewicht(); // in kg
}
```

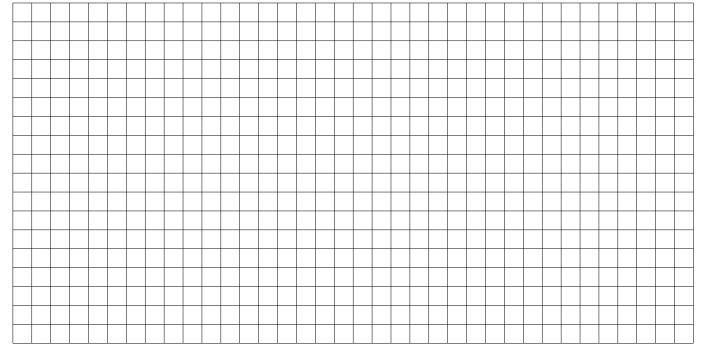
a) Schreiben Sie ein Interface Ware, das die gleichen Methoden wie die Interfaces Kaeuflich und Wiegbar und zusätzlich die String-wertige Methode getWarenName() fordert.



b) Schreiben Sie eine Methode static void printPreisListe(Ware[] w), die eine Liste der Waren im Array w mit Namen und Preis pro Kilogramm ausgibt.



c) Schreiben Sie eine Klasse Gemüse, deren Objekte man als Elemente des obigen Arrays vom Typ Ware verwenden könnte.



### Aufgabe 4: (Collections Framework, ca 20%)

Gegeben seien die Klasse Spieler

```
class Spieler {
    private String name;
    private int tore = 0;

    Spieler(String n) { name = n; }
    void addTor() { tore++; }
    int getTore() { return tore; }
}
```

sowie folgender Ausschnitt der Klasse FussballStat:

```
import java.util.*;
public class FussballStat {
   public static void main(String[] args) {
        private List<Mannschaft> mannschaften =
                                         new LinkedList<Mannschaft>();
        Spieler [] spieler1 = { new Spieler ("Podolski") , // ...
                                new Spieler("Schweinsteiger") };
        LinkedList<Spieler > spieler 2 = new LinkedList<Spieler >();
        {\tt spieler2.add(new~Spieler("Lucio"));~//~\dots}
        ArrayList<Spieler> spieler3 = new ArrayList<Spieler>();
        spieler3.add(new Spieler("Robben")); // ...
        Spieler [] spieler4;
        mannschaften.add(new Mannschaft(spieler1));
        mannschaften.add(new Mannschaft(spieler2));
        mannschaften.add(new Mannschaft(spieler3));
        spieler4 = mannschaften.get(0).getSpieler();
        // Weitere Methoden ...
```

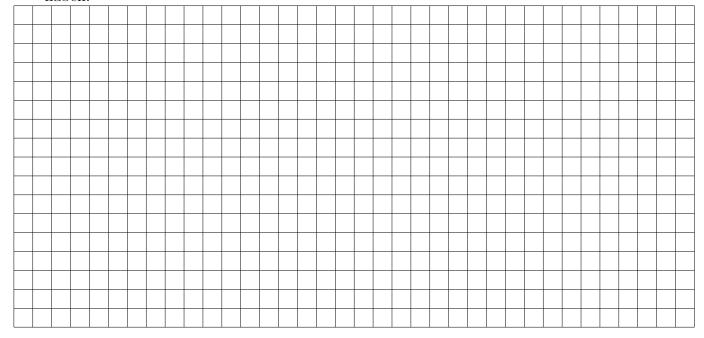
}

a) Ergänzen Sie das folgende Klassenfragment Mannschaft um sämtliche notwendigen Konstruktoren und Getter, damit die oben angegebene main-Methode in Klasse FussballStat lauffähig ist.

b) Ergänzen Sie die Klasse FussballStat um eine Methode mit dem folgenden Methodenkopf

```
Spieler[] getSpielerMitNToren(int n)
```

Diese Methode liefert ein Array mit allen Spielern aus mannschaften, die n Tore geschossen haben.



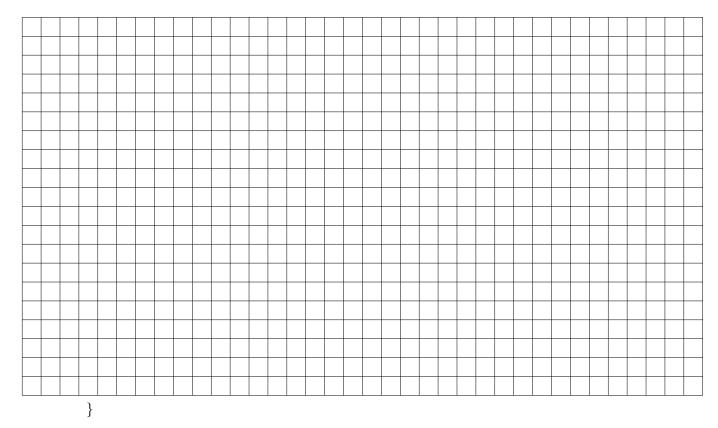
# Aufgabe 5: (Exceptions , ca. 15%)

Gegeben sei folgendes Programm, wobei die Klasse Reihung ein Array von Strings kapselt.

```
class Reihung {
    private String[] reihung = new String[10];

    void insert(int index, String element) throws IndexNegativeException {
```

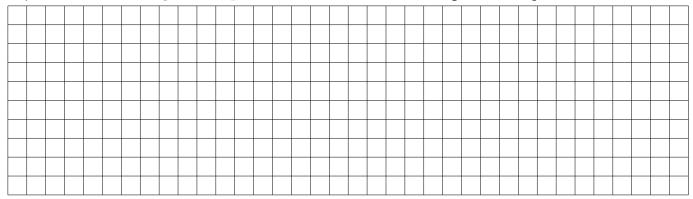
reihung[index] = element; //evt. ArrayIndexOutOfBoundsException



```
String get(int index){
    if (index<0 || index > reihung.length-1)
        return null;
    else
        return reihung[index];
}

public static void main(String[] args) throws IndexNegativeException {
    Reihung r = new Reihung();
    r.insert(0, "eins");
    r.insert(6, "sieben");
    r.insert(20, "einundzwanzig");
    System.out.println(r.get(20));
}
```

- a) Ergänzen Sie das gegebene Programmfragment um eine geeignete Ereignisbehandlung mit folgendem Verhalten: Tritt eine ArrayIndexOutOfBoundsException auf, wird geprüft ob ein negativer Index die Folge war. In diesem Fall wirft die Anwendung eine IndexNegativeException. Falls der Index positiv war, wird reihung ein neues, größeres Array zugewiesen, welches den Inhalt des bisherigen Arrays sowie das neue Element enthält.
- b) Geben Sie nachfolgend eine passende Definition für IndexNegativeException an.



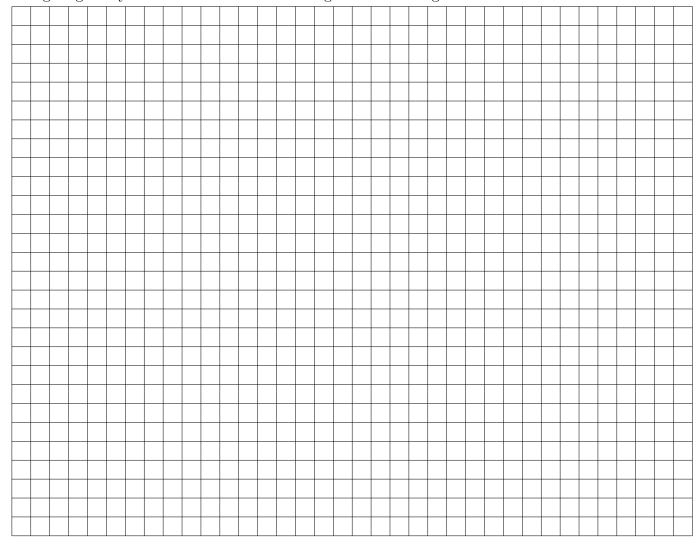
### Aufgabe 6: (GUI-Programmierung, ca. 20%)



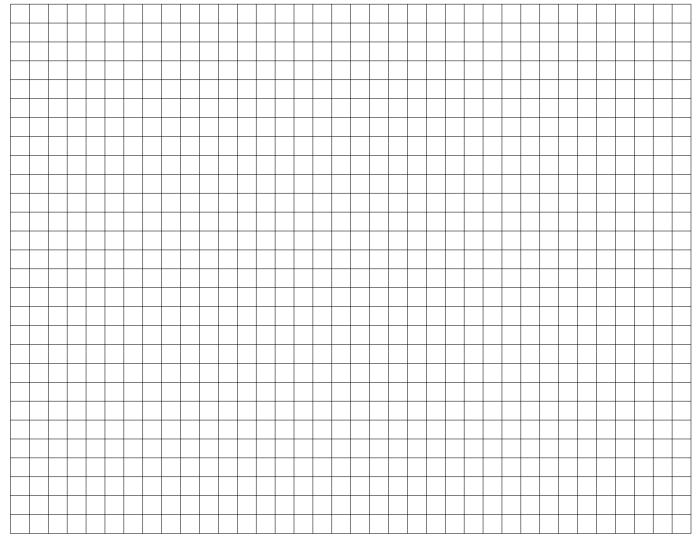
Gegeben seien die oben gezeigte Abbildung einer Anwendung zur Verkehrszählung, sowie das dazugehörige Programmfragment.

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
class Verkehrszaehlung extends JFrame implements ActionListener {
       private int pkw = 0, lkw = 0;
       private JLabel anzahlLabel = new JLabel("Anzahl");
       private JPanel centerPanel = new JPanel();
       private JButton lkwButton = new JButton("LKW");
       private JButton pkwButton = new JButton("PkW");
       private JTextField lkwTextfield = new JTextField("0");
       private JTextField pkwTextfield = new JTextField("0");
       private JButton resetButton = new JButton("Reset");
       Verkehrszaehlung(){
              super("VZ");
              pack();
              setVisible(true);
       }
       public static void main(String[] args) {
               Verkehrszaehlung anwendung = new Verkehrszaehlung();
       }
       public void actionPerformed (ActionEvent e){
       }
}
```

a) Ergänzen Sie das Programm-Fragment an der Stelle (a) so, dass Sie das in der Abbildung gezeigte Layout erhalten sowie die benötigten Listener registrieren.



- b) Ergänzen Sie das Programm-Fragment an der Stelle (b) so, dass die Anwendung wie folgt reagiert:
  - Wird der PKW-Button oder der LKW-Button gedrückt wird die korrespondierende Anzeige um eins hochgezählt.
  - Wird der Reset-Button gedrückt, werden beide Anzeigen auf 0 zurückgesetzt.



### Aufgabe 4: (Collections Framework, ca 20%)

Gegeben seien die Klasse Spieler

```
class Spieler {
    private String name;
    private int tore = 0;

    Spieler(String n) { name = n; }
    void addTor() { tore++; }
    int getTore() { return tore; }
}
```

sowie folgender Ausschnitt der Klasse FussballStat:

```
import java.util.*;
public class FussballStat {
   public static void main(String[] args) {
        private List<Mannschaft> mannschaften =
                                         new LinkedList<Mannschaft>();
        Spieler [] spieler1 = { new Spieler ("Podolski") , // ...
                                new Spieler("Schweinsteiger") };
        LinkedList<Spieler > spieler 2 = new LinkedList<Spieler >();
        {\tt spieler2.add(new~Spieler("Lucio"));~//~\dots}
        ArrayList<Spieler> spieler3 = new ArrayList<Spieler>();
        spieler3.add(new Spieler("Robben")); // ...
        Spieler [] spieler4;
        mannschaften.add(new Mannschaft(spieler1));
        mannschaften.add(new Mannschaft(spieler2));
        mannschaften.add(new Mannschaft(spieler3));
        spieler4 = mannschaften.get(0).getSpieler();
        // Weitere Methoden ...
```

### Aufgabe 6: (GUI-Programmierung, ca. 20%)



Gegeben seien die oben gezeigte Abbildung einer Anwendung zur Verkehrszählung, sowie das dazugehörige Programmfragment.

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
class Verkehrszaehlung extends JFrame implements ActionListener {
       private int pkw = 0, lkw = 0;
       private JLabel anzahlLabel = new JLabel("Anzahl");
       private JPanel centerPanel = new JPanel();
       private JButton lkwButton = new JButton("LKW");
       private JButton pkwButton = new JButton("PkW");
       private JTextField lkwTextfield = new JTextField("0");
       private JTextField pkwTextfield = new JTextField("0");
       private JButton resetButton = new JButton("Reset");
       Verkehrszaehlung(){
              super("VZ");
              pack();
              setVisible(true);
       }
       public static void main(String[] args) {
               Verkehrszaehlung anwendung = new Verkehrszaehlung();
       }
       public void actionPerformed (ActionEvent e){
       }
}
```