Prof. Christian Bischof, Ph.D.
Jonathan Heinen, Michael Lülfesmann, Carsten Otto

Vorname: _____

Nachname: _____

Matrikelnummer: _____

Studiengang (bitte ankreuzen):

- ○ Informatik Bachelor
- ○ Mathematik Bachelor
- ○ Informatik Lehramt
- ○ CES Bachelor
- ○ Sonstige: _____

	Anzahl Punkte	Erreichte Punkte
Aufgabe 1	**10**	
Aufgabe 2	**13**	
Aufgabe 3	**22**	
Aufgabe 4	**21**	
Summe	**66**	

Hinweise:

- Geben Sie Ihre Antworten in lesbarer und verständlicher Form an.
- Schreiben Sie mit dokumentenechten Stiften, nicht mit roten Stiften oder mit Bleistiften.
- Bitte beantworten Sie die Aufgaben auf den Aufgabenblättern (benutzen Sie auch die Rückseiten).
- Zusätzlich bereitgestellte Blätter werden **sofort** an die Aufgabenblätter **geheftet**.
- **Auf alle Blätter** (inklusive zusätzliche Blätter) müssen Sie **Ihren Namen und Ihre Matrikelnummer** schreiben.
- Was nicht bewertet werden soll, streichen Sie bitte durch.
- Werden **Täuschungsversuche** beobachtet, so wird die Übung mit **0 Punkten** bewertet.
- Geben Sie am Ende der Übung **alle Blätter zusammen mit den Aufgabenblättern ab**.

Matrikelnummer: **Name:**

Aufgabe 1 (Programmanalyse): (10 Punkte)

Geben Sie die Ausgabe des Programms für den Aufruf `java M` an. Schreiben Sie hierzu jeweils die ausgegebenen Zeichen hinter den Kommentar "`OUT:`".

```
public class A {
    public int x = 1;
    public A() {
        this(5);
    }
    public A(double d) {
        this((int) d / 3);
    }
    public A(int i) {
        x += i;
    }
    public void f(double z) {
        this.x += x + 1;
    }
}

public class B extends A {
    public int y = 5;
    public void f(double x) {
        this.x += x;
        y++;
    }
    public void f(int x) {
        y = 0;
    }
}

public class M {
    public static void main(String[] args) {
        A a = new A(7.0);
        System.out.println(a.x); //              OUT: [   ]

        a.f(10);
        System.out.println(a.x); //              OUT: [   ]

        B b = new B();
        System.out.println(a.x + " " + b.y); // OUT: [   ]   [   ]

        b.f(10.0);
        System.out.println(b.x + " " + b.y); // OUT: [   ]   [   ]

        a = b;
        a.f(1.0);
        System.out.println(a.x + " " + b.x); // OUT: [   ]   [   ]

        a.f(2);
        System.out.println(a.x + " " + b.y); // OUT: [   ]   [   ]
    }
}
```

Lösung:

Matrikelnummer: **Name:**

```
public class M {
    public static void main(String[] args) {
        A a = new A(7.0);
        System.out.println(a.x); //              OUT: [ 3 ]

        a.f(10);
        System.out.println(a.x); //              OUT: [ 7 ]

        B b = new B();
        System.out.println(a.x + " " + b.y); // OUT: [ 7 ]   [ 5 ]

        b.f(10.0);
        System.out.println(b.x + " " + b.y); // OUT: [ 16]   [ 6 ]

        a = b;
        a.f(1.0);
        System.out.println(a.x + " " + b.x); // OUT: [ 17]   [ 17]

        a.f(2);
        System.out.println(a.x + " " + b.y); // OUT: [ 19]   [ 8 ]
    }
}
```

Matrikelnummer: **Name:**

Aufgabe 2 (Verifikation): **(11 + 2 = 13 Punkte)**

Gegeben sei folgender *Java*-Algorithmus P zur Berechnung des Binomialkoeffizienten

$$\binom{n}{k} = \prod_{i=1}^{k} \frac{n+1-i}{i} \quad \text{für } k \geq 0$$

$\langle \varphi \rangle$ (Vorbedingung)

```
res = 1;
x = 0;
while (x < k) {
    res = res * (n − x) / (x + 1);
    x = x + 1;
}
```

$\langle \psi \rangle$ (Nachbedingung)

a) Als Vorbedingung für den oben aufgeführten Algorithmus P gelte $k \geq 0$ und als Nachbedingung:

$$res = \prod_{i=1}^{k} \frac{n+1-i}{i}$$

Vervollständigen Sie die Verifikation des folgenden Algorithmus im Hoare-Kalkül, indem Sie die unterstrichenen Teile ergänzen. Hierbei dürfen zwei Zusicherungen nur dann direkt untereinander stehen, wenn die untere aus der oberen folgt. Hinter einer Programmanweisung darf nur eine Zusicherung stehen, wenn dies aus einer Regel des Hoare-Kalküls folgt.

Gehen Sie in Ihrer Lösung davon aus, dass die Berechnung der Variablen *res* mit reellen Zahlen geschieht. Insbesondere gibt es in dieser Aufgabe keine Rundungseffekte wie bei der *int*-Division in Java.

Hinweise:

- Ein leeres Produkt hat den Wert 1, beispielsweise gilt $\prod_{i=x}^{y} i := 1$ für $x > y$.
- Auf der nächsten Seite finden Sie eine Vorlage, die Sie direkt ausfüllen dürfen.

Matrikelnummer: **Name:**

$\langle k \geq 0 \rangle$

$\langle$ _____ $\rangle$

```
res = 1;
```

$\langle$ _____ $\rangle$

$\langle$ _____ $\rangle$

```
x = 0;
```

$\langle$ _____ $\rangle$

$\langle$ _____ $\rangle$

```
while (x < k) {
```

$\langle$ _____ $\rangle$

$\langle$ _____ $\rangle$

```
    res = res * (n - x) / (x + 1);
```

$\langle$ _____ $\rangle$

$\langle$ _____ $\rangle$

```
    x = x + 1;
```

$\langle$ _____ $\rangle$

```
}
```

$\langle$ _____ $\rangle$

$\langle res = \prod_{i=1}^{k} \frac{n+1-i}{i} \rangle$

b) Beweisen Sie die Terminierung des Algorithmus P. Geben Sie hierzu eine Variante für die `while`-Schleife an. Zeigen Sie, dass es sich tatsächlich um eine Variante handelt, und beweisen Sie damit unter Verwendung des Hoare-Kalküls mit der Voraussetzung $k \geq 0$ die Terminierung.

Matrikelnummer: **Name:**

Lösung:

a)

$\langle k \geq 0 \rangle$

$\langle k \geq 0 \wedge 1 = 1 \rangle$

```
res = 1;
```

$\langle k \geq 0 \wedge res = 1 \rangle$

$\langle k \geq 0 \wedge res = 1 \wedge 0 = 0 \rangle$

```
x = 0;
```

$\langle k \geq 0 \wedge res = 1 \wedge x = 0 \rangle$

$\langle res = \prod_{i=0}^{x-1} \frac{n-i}{i+1} \wedge x \leq k \rangle$

```
while (x < k) {
```

$\langle res = \prod_{i=0}^{x-1} \frac{n-i}{i+1} \wedge x \leq k \wedge x < k \rangle$

$\langle res * (n-x)/(x+1) = \prod_{i=0}^{x} \frac{n-i}{i+1} \wedge x < k \rangle$

```
    res = res * (n - x) / (x + 1);
```

$\langle res = \prod_{i=0}^{x} \frac{n-i}{i+1} \wedge x < k \rangle$

$\langle res = \prod_{i=0}^{x+1-1} \frac{n-i}{i+1} \wedge x + 1 \leq k \rangle$

```
    x = x + 1;
```

$\langle res = \prod_{i=0}^{x-1} \frac{n-i}{i+1} \wedge x \leq k \rangle$

```
}
```

$\langle res = \prod_{i=0}^{x-1} \frac{n-i}{i+1} \wedge x \leq k \wedge x \not< k \rangle$

$\langle res = \prod_{i=1}^{k} \frac{n+1-i}{i} \rangle$

Ende unter Benutzung alternativer Invariante:

$\langle res = \prod_{i=1}^{x} \frac{n+1-i}{i} \wedge x \leq k \rangle$

```
while (x < k) {
```

$\langle res = \prod_{i=1}^{x} \frac{n+1-i}{i} \wedge x \leq k \wedge x < k \rangle$

$\langle res \cdot \frac{n-x}{x+1} = \frac{n-x}{x+1} \cdot \prod_{i=1}^{x} \frac{n+1-i}{i} \wedge x < k \rangle$

```
    res = res * (n - x) / (x + 1);
```

$\langle res = \frac{n-x}{x+1} \cdot \prod_{i=1}^{x} \frac{n+1-i}{i} \wedge x < k \rangle$

$\langle res = \prod_{i=1}^{x+1} \frac{n+1-i}{i} \wedge x + 1 \leq k \rangle$

```
    x = x + 1;
```

$\langle res = \prod_{i=1}^{x} \frac{n+1-i}{i} \wedge x \leq k \rangle$

```
}
```

$\langle res = \prod_{i=1}^{x} \frac{n+1-i}{i} \wedge x \leq k \wedge x \not< k \rangle$

$\langle res = \prod_{i=1}^{k} \frac{n+1-i}{i} \rangle$

b) Eine Variante ist $V = k - x$, denn aus der Schleifenbedingung $x < k$ folgt $V = k - x \geq 0$. Somit:

$\langle k - x = m \wedge x < k \rangle$

$\langle k - x = m \rangle$

```
res = res * (n - x) / (x + 1);
```

$\langle k - x = m \rangle$

$\langle k - (x + 1) < m \rangle$

```
x = x + 1;
```

$\langle k - x < m \rangle$

Damit ist die Terminierung der einzigen Schleife in P gezeigt.

_____.

Matrikelnummer: **Name:**

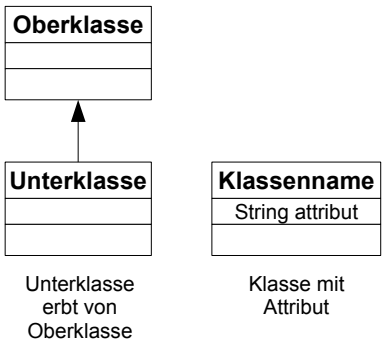
Aufgabe 3 (Fuhrpark): **(5 + 8 + 9 = 22 Punkte)**

Wir betrachten eine große Firma, die einen eigenen Fuhrpark mit verschiedenen Fahrzeugen verwaltet. In diesem Fuhrpark befinden sich neben konventionellen Fahrzeugen auch moderne Hybrid- und Solarautos. Zusätzlich gibt es im Fuhrpark auch einige Tanklaster.

a) In dieser Teilaufgabe geht es um den Entwurf einer entsprechenden Klassenhierarchie für die Fahrzeuge des Fuhrparks.

- Für jedes Fahrzeug ist der aktuelle Füllstand des Benzintanks (in ganzen Litern) bekannt.
- Ein Hybridauto funktioniert zusätzlich mit gespeichertem Strom, der auch zum Antrieb genutzt werden kann. Der Ladestand der Batterie wird ganzzahlig in Prozent gemessen und ist für jedes Hybridauto bekannt.
- Da es sich um eine moderne Firma handelt, gibt es im Fuhrpark sogar Solarautos. Ein Solarauto ist ein Hybridauto, das zusätzliche Energie aus Solarzellen bezieht. Zu jedem Solarauto ist bekannt, welcher Hersteller die Solarzellen produziert hat.
- Bei einem Tankwagen ist wichtig, wie viel Benzin – zusätzlich zum normalen Tank, gemessen in ganzen Litern – im Tankaufbau vorhanden ist.

Entwerfen Sie unter Berücksichtigung der Prinzipien der Datenkapselung eine geeignete Klassenhierarchie für die Fahrzeuge im Fuhrpark. Achten Sie darauf, dass gemeinsame Merkmale in Oberklassen zusammengefasst werden. Notieren Sie Ihren Entwurf graphisch und verwenden Sie dazu die folgende Notation:



Geben Sie für jede Klasse ausschließlich den jeweiligen Namen und die Namen und Datentypen ihrer Attribute an. Methoden von Klassen müssen nicht angegeben werden – auch nicht von den Methoden, die in den nächsten Teilaufgaben eingeführt werden.

b) Alle Fahrzeuge des Fuhrparks werden täglich vollgetankt. Hierbei werden auch die Tankaufbauten der Tankwagen gefüllt.

Jedes Fahrzeug hat eine Methode `int volltanken()`, die den Tank des jeweiligen Fahrzeugs füllt und zurückgibt, wie viel Benzin für diese Füllung benötigt wurde. Der Füllstand des Tanks wird dabei entsprechend angepasst. Weiterhin stellt jeder Tankwagen eine Methode `int aufbauVolltanken()` zur Verfügung, die nur den Tankaufbau füllt und die dafür benötigte Benzinmenge zurückgibt. Auch hier wird der Füllstand des Tankaufbaus angepasst. Diese beiden Methoden müssen Sie **nicht** implementieren, dürfen diese aber natürlich benutzen!

Schreiben Sie in dieser Teilaufgabe eine Methode `int betankeFahrzeuge(...)`. Diese Methode soll jedes in einem entsprechenden Array übergebene Fahrzeug volltanken und anschließend zurückgeben, wie viele Liter Benzin insgesamt benötigt wurden. Gehen Sie davon aus, dass das übergebene Array nicht der `null`-Wert ist. Kennzeichnen Sie die Methode mit dem Schlüsselwort `static`, falls angebracht.

c) Für den Chef des Fuhrparks ist interessant, welche Reichweite ein bestimmtes Fahrzeug im Moment hat. Hierbei ist natürlich relevant, wie voll der Tank ist. Bei Hybrid- und Solarautos kann sich die Reichweite aufgrund der speziellen Eigenschaften dieser Fahrzeuge vergrößern.

Die Reichweite der einzelnen Fahrzeuge lässt sich wie folgt berechnen:

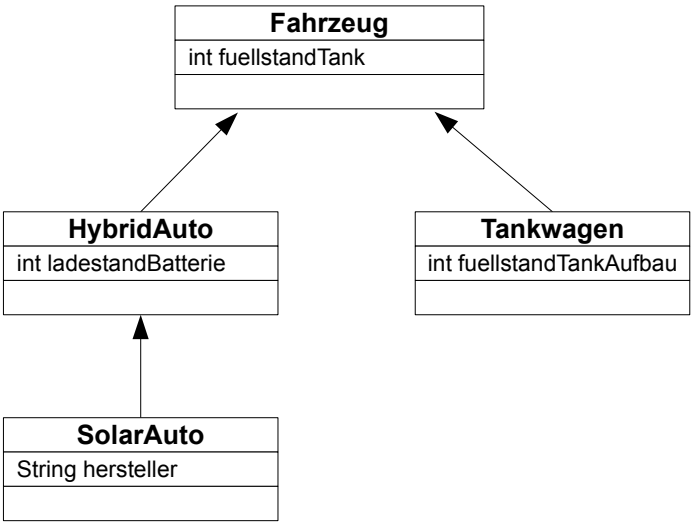
- Jedes Fahrzeug mit einem Benzintank kann pro Liter Benzin 15 Kilometer fahren.
- Jedes Hybridauto kann zusätzlich pro Prozent Batterieladung 2 Kilometer fahren.
- Durch den Einsatz der Solarzellen verlängert sich die durch Benzin und Strom erzeugte Reichweite um 5%. Hierbei sind eventuelle Wetterschwankungen unerheblich.
- Der Aufbautank eines Tankwagens ist für die Reichweitenberechnung **nicht** relevant.

Schreiben Sie die benötigten Methoden `int getReichweite()`, die diese Reichweite gerundet auf einen ganzzahligen Wert zurückgibt. Geben Sie dabei jeweils an, für welche Klasse Sie die Methode implementieren. Verwenden Sie für Zwischenergebnisse, wenn angebracht, entsprechende Methoden der Oberklasse(n), um Code-Duplikation zu vermeiden. Kennzeichnen Sie die Methoden mit dem Schlüsselwort `static`, falls angebracht.

Matrikelnummer: **Name:**

Lösung:

a)



b)

```
public static int betankeFahrzeuge(Fahrzeug[] fahrzeuge){
    int summe = 0;

    for(int i = 0; i < fahrzeuge.length; i++){

        if(fahrzeuge[i] != null){

            summe += fahrzeuge[i].vollTanken();

            if(fahrzeuge[i] instanceof Tankwagen){
                summe += ((Tankwagen) fahrzeuge[i]).aufbauVolltanken();
            }
        }
    }
    return summe;
}
```

c)

- Klasse Fahrzeug:

```
public int getReichweite(){
    return fuellstandTank * 15;
}
```

- Klasse HybridAuto:

```
public int getReichweite(){
    return super.getReichweite() + ladestandBatterie * 2;
}
```

- Klasse SolarAuto:

```
public int getReichweite(){
    return (int)(super.getReichweite() * 1.05);
}
```

Matrikelnummer: **Name:**

Aufgabe 4 (Studentenverzeichnis): **(8 + 2 + 11 = 21 Punkte)**

Heutzutage sind die meisten Studenten Mitglied bei einem Studentenportal im Internet. Hier haben sie die Möglichkeit, ein eigenes Profil mit ihren Kontaktdaten zu pflegen, ihre Bilder einzustellen und vor allem mit anderen Mitgliedern zu kommunizieren. Eine wichtige Funktion ist die Freundesliste, in der die Mitglieder verwaltet werden, mit denen man befreundet ist.

In dieser Aufgabe soll eine vereinfachte Darstellung eines Studentenportals um Funktionen erweitert werden.
Die folgenden Klassen `Student` und `Foto` werden von dem Betreiber bereitgestellt:

Student:
Ein *Student* hat eine Sammlung von eigenen Fotos, auf denen beliebige Studenten abgebildet sein können. Weiterhin hat jeder Student eine möglicherweise leere Liste von Freunden (anderen Studenten). Die folgenden Methoden stellt die Klasse Student zur Verfügung:

- `void sendeNachricht(String text)`
  Schickt dem Studenten eine Nachricht mit dem übergebenen Text.
- `int anzahlFreunde()`
  Gibt die Anzahl der Freunde des Studenten zurück (≥ 0).
- `Student getFreund(int index)`
  Gibt einen bestimmten Freund (Instanz der Klasse Student) zurück, wobei die erlaubten Indizes von `0` bis `anzahlFreunde() - 1` laufen.
- `int anzahlFotos()`
  Gibt die Anzahl der Fotos zurück, die der Student hochgeladen hat (≥ 0).
- `Foto getFoto(int index)`
  Gibt ein bestimmtes Foto zurück, wobei die erlaubten Indizes von `0` bis `anzahlFotos() - 1` laufen.

Foto:
Die Klasse Foto repräsentiert Fotos, die von Studenten hochgeladen wurden. Jedes Foto hat eine Liste von Studenten, die auf diesem Foto abgebildet sind.

- `int anzahlStudenten()`
  Gibt die Anzahl der Studenten zurück, die auf dem Foto abgebildet sind (≥ 0).
- `Student getStudent(int index)`
  Gibt einen bestimmten auf dem Foto abgebildeten Studenten zurück, wobei die erlaubten Indizes von `0` bis `anzahlStudenten() - 1` laufen.
- `void drucke()`
  Druckt das Foto aus.

Die Methoden `Student getFreund(int index)`, `Foto getFoto(int index)` und `Student getStudent(int index)` geben für die jeweils gültigen Indizes nie `null` zurück.

a) In dieser Teilaufgabe soll das System um eine Möglichkeit erweitert werden, Partyeinladungen zu verschicken. Die Freundesstruktur soll so ausgenutzt werden, dass Freunde, Freundesfreunde, usw. eingeladen werden. Hierbei unterscheiden wir verschiedene Stufen von Freunden eines bestimmten Studenten.

- Auf *Stufe 1* befinden sich alle direkten Freunde.
- Auf *Stufe 2* befinden sich alle direkten Freunde der Freunde auf *Stufe 1*.
- ...
- Auf *Stufe* $n+1$ befinden sich alle direkten Freunde der Freunde auf *Stufe* n.

Da hier jede Freundschaft auf Gegenseitigkeit beruht, ist also jeder Student beispielsweise auch auf der Stufe 1 seiner direkten Freunde.

Implementieren Sie nun eine rekursive Methode `void einladungenVerschicken(int stufenLimit)` in der Klasse `Student`, die eine Nachricht an alle Freunde der Stufen 1 bis `stufenLimit` verschickt. Gehen Sie davon aus, dass das diese Methode vom Benutzer nur mit `stufenLimit >= 1` aufgerufen wird. Die Benutzung von Schleifen ist zulässig.

Verschicken Sie jeweils die Nachricht `"Einladung!"`. Es ist in Ordnung, wenn Studenten die Einladung mehrfach bekommen. Kennzeichnen Sie die Methode mit dem Schlüsselwort `static`, falls angebracht.

b) In Aufgabenteil a) erhält ein Student sehr wahrscheinlich mehrere Einladungen beim Aufrufen der Methode `void einladungenVerschicken(int stufenLimit)`. Beschreiben Sie **in höchstens zwei Sätzen** eine Idee zur Anpassung dieser Methode, so dass jeder Student maximal eine Einladung bekommt. Schreiben Sie keinen Quellcode!

c) Manchmal, vor allem nach einer Party, ist es interessant zu wissen, auf welchen Fotos man zu sehen ist. Hierfür macht es besonders viel Sinn, die Fotos seiner Freunde zu durchsuchen.

Erweitern Sie die Klasse `Student` um die Methode `void druckeFreundesfotosMitMir()`. Diese Methode durchsucht die Fotos der direkten Freunde des Studenten und druckt die Fotos aus, auf denen er abgebildet ist. Kennzeichnen Sie die Methode mit dem Schlüsselwort `static`, falls angebracht.

Lösung:

a)
```
public void einladungVerschicken(int stufenLimit){

    if(stufenLimit == 0)
        return;

    for(int i = 0; i < anzahlFreunde(); i++){
        getFreund(i).sendeNachricht("Einladung!");
        getFreund(i).einladungVerschicken(stufenLimit -1);
    }

}
```

b) Die Studenten, denen bereits eine Einladung geschickt wurde, müssen in einer geeigneten Datenstruktur (z.B. eine Liste) mitgeführt werden. Eine Einladung wird nur dann verschickt, wenn der Student sich noch nicht in der mitgeführten Datenstruktur befindet.

c)
```
public void druckeFreundesFotosMitMir(){

    // durchlaufe alle Freunde
    for(int i = 0; i < anzahlFreunde(); i++){

        Student freund = getFreund(i);
```

```
            // durchlaufe alle Fotos des Freundes i
            for(int j = 0; j < freund.anzahlFotos(); j++){

                Foto foto = freund.getFoto(j);
                // durchlaufe alle abgebildete Freunde
                for(int k = 0; k < foto.anzahlStudenten(); k++){

                    Student student = foto.getStudent(k);
                    if(student == this){

                        foto.drucke();
                    }
                }
            }
        }
    }
```