Probeklausur: Grundlagen der Programmierung

Kurs: RV-WWIBE124

Dozent: Phillip Goellner

Datum: _____

Allgemeine Hinweise:

- Die Bearbeitungszeit beträgt 120 Minuten.
- In Anbetracht der begrenzten Bearbeitungszeit sollten die Antworten kurz und faktenorientiert formuliert werden.
  Logisch eindeutige und inhaltlich vollständige Stichpunkte sind erlaubt.
- Die Aufgaben dürfen direkt im dafür vorgesehenen Bereich auf dem Aufgabenblatt gelöst werden. Es ist aber auch erlaubt, die Aufgaben auf zusätzlichen Seiten zu beantworten.
- Es sind keinerlei Hilfsmittel (d.h. kein Skript, JavaDoc oder Taschenrechner) zugelassen.
- Die Benotung erfolgt entsprechend der Standard-Notenskala der DHBW Ravensburg.
- Seiten die keiner Matrikelnummer zugeordnet werden können, können nicht gewertet werden.
  Daher bitte die Matrikelnummer auf jeder Seite oben rechts eintragen.
- Komplett unleserliche Schrift kann ebenfalls nicht gewertet werden.
  Daher bitte leserlich schreiben.

Zusätzliche Hinweise zu den Programmier-Aufgaben:

- Generell ist davon auszugehen, dass in der entsprechenden Klasse keinerlei weitere Variablen, Konstanten, Konstruktoren oder Methoden definiert sind.
- Falls eine `run()` Methode vorgegeben ist, ist davon auszugehen, dass diese unmittelbar zum Start des Programms ausgeführt wird.
- Es ist nicht nötig, verwendete Klassen zu importieren.
- Sofern nicht explizit gefordert, ist ein kommentieren des Codes nicht nötig.
- Bitte auf das korrekte Setzen von Klammern `() [] {}`, Semikola `;`, Dezimalpunkten `.` und Anführungszeichen `"` achten
- Bitte auf die korrekte Verwendung von Vergleichsoperatoren `< > =` achten.
  Kombinierte Vergleichsoperatoren `≤ ≥` sind nicht erlaubt
- Bitte auf die korrekte Verwendung von arithmetischen Operatoren `+ - * / %` achten

Die Aufgaben beginnen auf der nächsten Seite.

Viel Erfolg!

Aufgabe 1: Theorie – 6 Punkte (kumuliert: 6 von 100)

Es folgt eine Tabelle mit Aussagen zu den Themen Exceptions und Vererbung. Kreuze jeweils an, ob die Aussage richtig oder falsch ist. [je 1 Punkt]

Aussage	Richtig	Falsch
Man behandelt Exceptions mit Try-Catch-Konstrukten.		
Unterklassen erben alle Attribute, Methoden und Konstruktoren von ihrer Oberklasse.		
Unterklassen von `RuntimeException` bezeichnet man als "Unchecked Exceptions"		
Dass eine Methode eine Exception werfen kann, wird mittels des Keywords `throws` in der Signatur deklariert.		
Klassen können von beliebig vielen Interfaces mittels des Keywords `extends` erben.		
Unterklassen können Methoden ihrer Oberklasse überschreiben. Dafür muss das Keyword `overrides` verwendet werden.		

Aufgabe 2: Angewandte Programmierung – 5 Punkte (kumuliert: 11 von 100)

Vervollständige die untenstehende Methode `run()`

1 Deklariere eine Variable und weise ihr den Ganzzahl-Wert 16 zu. [1 Punkt]

2 Danach prüfe ob die Variable einen größeren Wert als 11 hat.
Wenn dies der Fall ist, gebe einen beliebigen Text auf der Konsole aus. [3 Punkte]

3 Falls nicht, gebe den halben Variableninhalt auf der Konsole aus. [1 Punkt]

Code

```
void run() {



}
```

Aufgabe 3: Theorie – 4 Punkte (kumuliert: 15 von 100)

Erkläre die Begriffe *Method Overloading* und *Method Overriding*.
Dabei soll der Unterschied deutlich werden.

Antwort	
	Overloading:
	Overriding:

Aufgabe 4: Angewandte Programmierung – 4 Punkte (kumuliert: 19 von 100)

1 Vervollständige die untenstehende `run()` Methode, sodass unter Verwendung einer Schleife alle geraden Zahlen zwischen 10 und 18 (jeweils einschließlich) auf der Konsole ausgegeben werden. [3 Punkte]

2 Verwende zur Ausgabe die bestehende `printToConsole()` Methode. Diese muss nicht verändert oder nochmal abgeschrieben werden. [1 Punkt]

Code

```
void run() {









}

void printToConsole(int x) {
  System.out.println(x + " ist eine gerade Zahl");
}
```

Aufgabe 5: Angewandte Programmierung – 4 Punkte (kumuliert: 23 von 100)

Vervollständige die untenstehende `run()` Methode.

1 Deklariere und instanziiere ein Array, welches 69 `Karel`-Instanzen aufnehmen kann. [2 Punkte]

2 Speichere das bestehende `Karel`-Objekt an der letzten Index-Position des Arrays ab. [1 Punkt]

3 Worin besteht der Hauptnachteil eines Arrays? [1 Punkt]

Code

```
Karel karel = new Karel();

void run () {



}
```

Antwort

Aufgabe 6: Theorie – 6 Punkte (kumuliert: 29 von 100)

1 Was sind statische Testverfahren? [2 Punkte]

2 Was sind dynamische Testverfahren? [2 Punkte]

3 Nenne zwei Beispiele für dynamische Testverfahren. [2 Punkte]

Antwort	

Aufgabe 7: Programmier-Verständnis – 6 Punkte (kumuliert: 35 von 100)

1 In welcher Reihenfolge werden die Buchstaben auf der Konsole ausgegeben, wenn die untenstehende Methode `run()` aufgerufen wird? [4 Punkte]

2 Die Methode `igniteGas()` ist zwei mal implementiert. Erkläre kurz welche Voraussetzungen für das Implementieren mehrerer Methoden mit gleichem Namen gelten. [2 Punkte]

Hinweis: Der Code befindet sich auf der folgenden Seite.

Antwort	

Code

```
void run() {
  System.out.print("A");
  startIgnition(2);
  liftOff();
  System.out.print("K");
}

void startIgnition(int numberOfTanks) {
  checkFuelTanks(numberOfTanks);
  igniteGas();
  System.out.print("E");
}

void checkFuelTanks(int numberOfTanks) {
  if (numberOfTanks > 2) {
    System.out.print("H");
  } else {
    System.out.print("F");
  }
}

int getDefaultTemparature() {
  System.out.print("C");
  return 1337;
}

void igniteGas(int temperature) {
  System.out.print("B");
}

void igniteGas() {
  igniteGas(getDefaultTemparature());
  System.out.print("D");
}

void liftOff() {
  System.out.print("L");
}
```

Aufgabe 8: Angewandte Programmierung – 7 Punkte (kumuliert: 42 von 100)

Erstelle die Methode `smallest( )`, welche Ganzzahlen (`a`, `b`, `c`) als Übergabeparameter annimmt.

Es kann davon ausgegangen werden, dass immer drei unterschiedliche Werte an die Methode übergeben werden.

Die Methode soll immer den kleinsten Wert zurückgeben (als Rückgabewert).
Beispiele:

```
smallest(3, 2, 1)       →    1
smallest(-1, 7, 5)      →   -1
smallest(-7, -3, -1)    →   -7
```

Code

```
int smallest(int a, int b, int c) {



}
```

Aufgabe 9: Theorie – 8 Punkte (kumuliert: 50 von 100)

– 50%-Marke –

1 Nenne fünf Aspekte von Softwarequalität nach ISO 25010. [5 Punkte]

2 Erkläre drei davon. [3 Punkte]

Antwort	

Aufgabe 10: Programmier-Verständnis – 5 Punkte (kumuliert: 55 von 100)

Gib jeweils an welcher Wert in die Variable a gespeichert wird. Beachte dabei die arithmetischen Regeln in Java; die Ausdrücke sind so zu verstehen, als wären sie Teil eines Java-Programmes (alle Zeilen sind kompilierbar). [je 1 Punkt]

Jede Zeile ist unabhängig zu betrachten.

Berechnung	**Wert von a**
Beispiel: `int a = 1 + 2;`	3
`double a = 0.5 * 3;`	
`int a = 3 / 4;`	
`double a = 7.0 / 2;`	
`int a = (int) 1.5 * 3;`	
`double a = 3 * (int) (2.5 + 2);`	

Aufgabe 11: Theorie und Programmier-Verständnis – 8 Punkte (kumuliert: 63 von 100)

Beantworte die folgenden Fragen im Kontext des untenstehenden Klassendiagramms.

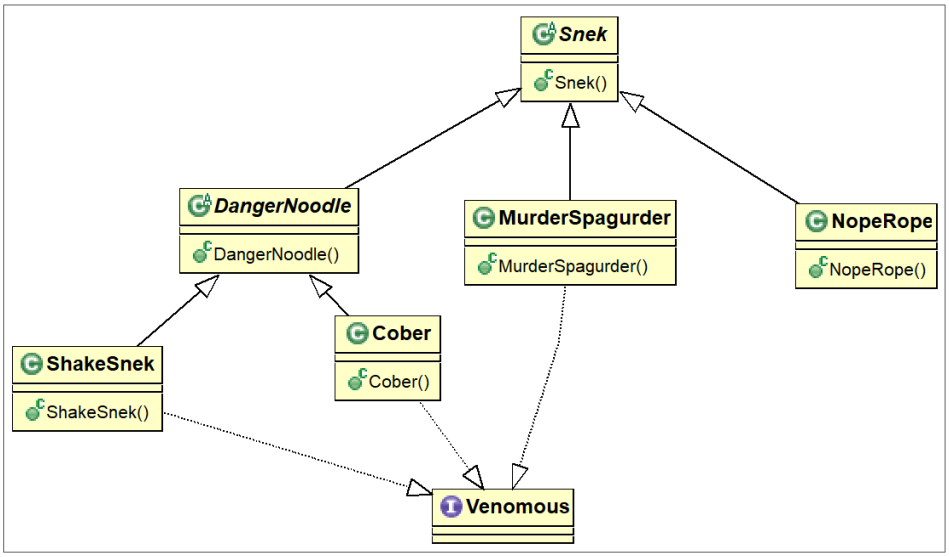
1 Wie viele verschiedene Typen hat ein Objekt der Klasse `ShakeSnek` (ausgenommen `java.lang.Object`)? [1 Punkt]

2 Wie nennt man den Mechanismus, der ermöglicht, dass Objekte mehrere Typen haben? [1 Punkt]

3 In welcher Beziehung stehen die Klassen `Snek` und `MurderSpagurder`? [1 Punkt]

4 Bewerte ob die untenstehenden 5 Code-Blöcke (folgende Seite!) jeweils korrekt sind oder nicht. [je 1 Punkt]
Bitte betrachte jeden Block unabhängig von den anderen.

Hinweise: `Snek` und `DangerNoodle` sind abstrakte Klassen. `Venomous` ist ein Interface.



Antwort	

Code-Block	Richtig	Falsch
`Snek snek = new NopeRope();`		
`Venomous venomousSnek = new DangerNoodle();`		
`Cober cober = new Cober();` `Venomous venomousCober = (DangerNoodle) cober;`		
`Snek[] allMySneks = new Snek[42];` `allMySneks[5] = new Venomous();`		
`Snek snek = new MurderSpagurder();` `Venomous venomousSnek = (MurderSpagurder) snek;`		

Aufgabe 12: Theorie & Angewandte Programmierung – 8 Punkte (kumuliert: 71 von 100)

1 Erkläre kurz was eine rekursive Methode ist. [2 Punkte]

2 Implementiere eine Methode `facotorial()`, die die Fakultät einer übergebenen Zahl größer Null berechnet und als natürliche Zahl zurückgibt. Ein Überprüfen auf ungültige Werte ist nicht notwendig.

Die Fakultät einer Zahl *n* ist gleich dem Produkt aller Zahlen von 1 bis *n*, z.B.:

```
factorial(1)   → 1
factorial(2)   → 2  (1 * 2)
factorial(3)   → 6  (1 * 2 * 3)
factorial(4)   → 24 (1 * 2 * 3 * 4)
```

Implementiere die Methode *rekursiv* auf der folgenden Seite. [6 Punkte]

Antwort	

Code	

Aufgabe 13: Theorie – 6 Punkte (kumuliert: 77 von 100)

1 Nenne zwei Gründe, aus denen man Packages in der Java-Entwicklung verwenden sollte. [2 Punkte]

2 Was sind die Bestandteile eines Fully Qualified Class Names? [2 Punkte]

3 Was wird in Java im Stack gespeichert, was im Heap? [2 Punkte]

Antwort	

Aufgabe 14: Angewandte Programmierung – 4 Punkte (kumuliert: 81 von 100)

Entwickle die Methode `floor()`, welche eine positive Dezimalzahl (`double`) entgegen nimmt und als Ergebnis die abgerundete natürliche Zahl (`int`) zurückgibt.

Beispiele:

```
floor(1.0)        → 1
floor(3.2)        → 3
floor(8.9)        → 8
```

Code	

Aufgabe 15: Theorie – 4 Punkte (kumuliert: 85 von 100)

1 Nenne zwei Unterschiede zwischen primitiven Datentypen und Objektdatentypen in Java [2 Punkte]

2 Welche beiden Konstrukte kennst du, mit denen man neue Datentypen in Java definiert? [2 Punkte]

Antwort	

Aufgabe 16: Angewandte Programmierung – 5 Punkte (kumuliert: 90 von 100)

Entwickle eine Methode `average( )`, die ein `double`-Array (mit Länge größer Null) entgegen nimmt und einen `double`-Wert zurückgibt, der dem arithmetischen Mittel aller Zahlen im übergebenen Array entspricht. [5 Punkte]
Dabei sollen Arrays beliebiger Länge übergeben werden können.

Beispiele:

```
average([2.0, 3.0, 4.0])              → 3
average([1.0, 2.0, 3.0, 4.0])         → 2.25
average([1.0, -5.0, 1.0])             → -1
average([0.0, 0.0, 1.0, -1.0])        → 0
```

Code

```
double average(double[] numbers) {



}
```

Aufgabe 17: Angewandte Programmierung – 10 Punkte (kumuliert: 100 von 100)

Vervollständige die untenstehende Klasse `Zoo` (folgende Seite!):

1 `Zoo` soll ein Attribut mit Namen `birds` haben, worauf nur innerhalb der Zoo-Klasse zugegriffen werden kann. `birds` soll den Datentypen Bird-Array haben und in der Lage sein 15 Bird-Objekte aufzunehmen (dafür darf direkt ein Wert zugewiesen werden). [4 Punkte]

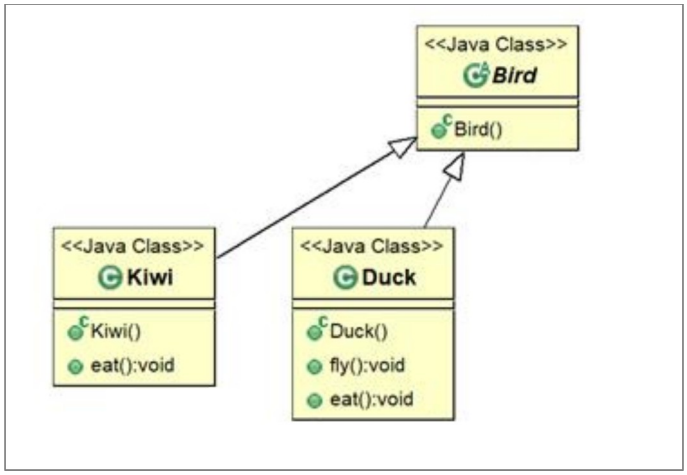
Vervollständige die Methode `letAllDucksFly()`:

2 Iteriere mit einer Schleife über alle Inhalte des Bird-Arrays `birds`. [2 Punkte]

3 Überprüfe in jeder Iteration, ob das gegenwärtige Objekt von Typ `Duck` ist. [2 Punkte]

4 Falls das gegenwärtige Objekt von Typ `Duck` ist, rufe die Methode `fly()` des Objektes auf [2 Punkte]

Beachte bei den Implementierungen das Klassendiagramm!



Code

```
public class Zoo {



    public void letAllDucksFly() {



    }

}
```

– Geschafft! –