

Klausur
Einführung in die Programmierung
19.02.2019 (WS 2018/19)

Hinweise

1. Bearbeitungszeit: 120 Minuten
2. Zum Bestehen der Klausur sind 50 Punkte erforderlich.
3. Bitte auf das erste Blatt Name, Studiengang und Matrikelnummer eintragen.
4. Erlaubte Hilfsmittel: keine
5. Die Klammerung der Klausur nicht lösen!
6. Die Lösungen in die Klausurvorlage eintragen. Weiteres Papier können Sie bei der Klausuraufsicht anfordern.
7. Alle vorgegebenen und zu erstellenden Programmtexte beziehen sich auf die Programmiersprache Java.

Viel Erfolg!

Aufgabe	Max. Punktzahl	Erreichte Punktzahl
1 (Multiple Choice, Wissen)	10	
2 (Ein-/Ausgabe, Kontrollstrukturen)	10	
3 (Programmieraufgaben)	15	
4 (Array-Programmierung)	13	
5 (Klassen und Objekte)	14	
6 (Fehlersuche)	10	
7 (UML nach Java übertragen)	16	
8 (Speicherabbild)	12	
<i>Bonuspunkte</i>	<i>10</i>	
Gesamtpunktzahl	100+10=110	

Aufgabe 1 (Multiple Choice, Wissen)**[10 Punkte]**

- a) Kreuzen Sie in den folgenden Teilaufgaben an, welche Aussagen richtig und welche falsch sind. Für jede korrekt markierte Aussage erhalten Sie 0,5 Punkte, für jede falsch markierte werden 0,5 Punkte abgezogen. Pro Teilaufgabe erhalten Sie mindestens 0 Punkte.

Welche der folgenden Anweisungen ergibt die Ausgabe Hello154?	Richtig	Falsch
<code>System.out.print("Hello"+15+4);</code>		
<code>System.out.print("Hello"+150+4);</code>		
<code>System.out.print("Hello"+(15+4));</code>		
<code>System.out.print(("Hello"+15)+4);</code>		

Bewerten Sie folgende Aussagen. Ein Klassenattribut mit Paketsichtbarkeit	Richtig	Falsch
wird in UML unterstrichen.		
wird in UML durch ~ gekennzeichnet.		
wird in Java durch <code>static</code> gekennzeichnet.		
wird in Java durch <code>package</code> gekennzeichnet.		

Bewerten Sie die folgenden Aussagen. Sei A eine Klasse in Java. Dann gilt:	Richtig	Falsch
Wenn A eine abstrakte Methode enthält, so muss A abstrakt sein		
Wenn A abstrakt ist, so muss A eine abstrakte Methode enthalten		
Wenn A ein Interface implementiert, so muss A alle Methoden des Interfaces implementieren		
Wenn A nicht abstrakt ist und ein Interface implementiert, so muss A alle Methoden des Interfaces implementieren		

- b) Berechnen Sie die Binär- und Hexadezimaldarstellung der Zahl 1022

c) Berechnen Sie die Zweierkomplementdarstellung (8 Bit) der Zahl -17

Aufgabe 2 (Ein-/Ausgabe, Kontrollstrukturen)**[10 Punkte]**

Ergänzen Sie die Klasse `Mittelwert` um eine `main`-Methode, die solange Ganzzahlen (`int`) von der Konsole einliest, bis `-1` eingegeben wird. Anschließend sollen die Summe und der Mittelwert (arithmetisches Mittel) der eingegebenen Zahlen (ohne die abschließende Eingabe `-1`) auf der Konsole ausgegeben werden.

Geben Sie notwendige `import`-Anweisungen an.

```
public class Mittelwert
{
    public static void main(String[] args)
    {

    }

}
```

Aufgabe 3 (Programmieraufgaben)**[15 Punkte]**

- a) Schreiben Sie eine Methode, die in einem int-Feld mit Werten zwischen 0 und 9 nach dem Wert sucht, der am häufigsten im Feld vorkommt. Sollten Werte gleichhäufig vorkommen, so soll der kleinste von ihnen zurückgegeben werden.
Beispiel: Für das Feld { 1,2,3,3,5,9,7,7,1,3,7 } soll der Wert 3 zurückgegeben werden, da 3 und 7 jeweils dreimal vorkommen und 3 kleiner als 7 ist.

```
public static int berechneHaeufigstenWert(int[] feld)
{
    assert(feld != null);
```

```
}
```

- b) Schreiben Sie nun eine Methode, die zu der übergebenen Zahl n die Ziffer zurückgibt, die am häufigsten in der Dezimaldarstellung der Zahl n vorkommt. Sollten mehrere Ziffern gleichhäufig vorkommen, so soll die kleinste der Ziffern zurückgegeben werden.

Beispiel: Für die Zahl $n=2031771$ kommen die Ziffer 1 und 7 jeweils zweimal vor. Es muss also in diesem Fall die 1 zurückgegeben werden.

Sie dürfen die Methode `berechneHaeufigstenWert()` aus Aufgabenteil a) benutzen.

```
public static int berechneHaeufigsteZiffer(int n)
{
    assert (n>0) ;
```

```
}
```

Aufgabe 4 (Array-Programmierung)**[13 Punkte]**

- a) Programmieren Sie die Methode `umkehr`, die für ein beliebiges eindimensionales Feld ganzer Zahlen die Reihenfolge der Elemente umkehrt. Beispiel:

1	-7	13	-5	2	27	11	4	51
---	----	----	----	---	----	----	---	----

vorher

51	4	11	27	2	-5	13	-7	1
----	---	----	----	---	----	----	----	---

nachher

Hinweis: Innerhalb der Methode `umkehr` darf kein neues Feld erzeugt werden!

```
public static void umkehr(int[] feld)
{
    assert(feld != null);
```

```
}
```

- b) Programmieren Sie die Methode `treppe`, die für ein beliebiges $n > 0$ ein rechteckiges, zweidimensionales Feld mit n Zeilen und $2n-1$ Spalten erzeugt. Die erste Zeile enthält beginnend ab dem Zeilenanfang n mal das Zeichen `*` und $n-1$ Leerzeichen. In jeder Zeile verschieben sich die Sterne um eine Spalte nach rechts.

Beispiel für $n = 5$:

*	*	*	*	*				
	*	*	*	*	*			
		*	*	*	*	*		
			*	*	*	*	*	
				*	*	*	*	*

```
public static char[][] treppe(int n)
{
    assert(n>0);
```

```
}
```

Aufgabe 5 (Klassen und Objekte)**[14 Punkte]**

Betrachten Sie die folgende Klasse Person:

```
public abstract class Person {
    private String name;

    public Person(String name)
    {
        this.name = name;
    }
    public String toString()
    {
        return name + " ist " + getBeruf()
            + " und verdient " + getGehalt() + " Euro";
    }
    public abstract int getGehalt();
    public abstract String getBeruf();

    public static void main(String[] args)
    {
        Person[] person = { new Anwalt("Andreas"),
                             new Politiker("Helmut",10000,'m'),
                             new Politiker("Heidi",5000,'w'),
                             new Manager(),
                             new Manager() };

        for (int i=0; i<person.length; i++)
            System.out.println(person[i]);
    }
}
```

Ergänzen Sie die folgenden Klassen um geeignete Methoden und Attribute, so dass die obige main-Methode die Ausgabe

Andreas ist Anwalt und verdient 20000 Euro
Helmut ist Politiker und verdient 10000 Euro
Heidi ist Politikerin und verdient 5000 Euro
Franz der 1. ist Manager und verdient 10000 Euro
Franz der 2. ist Manager und verdient 20000 Euro

erzeugt.

Die Methode `toString()` darf hierbei in diesen Klassen nicht überschrieben werden.

```
public class Anwalt extends Person  
{
```

```
}
```

```
public class Politiker extends Person  
{
```

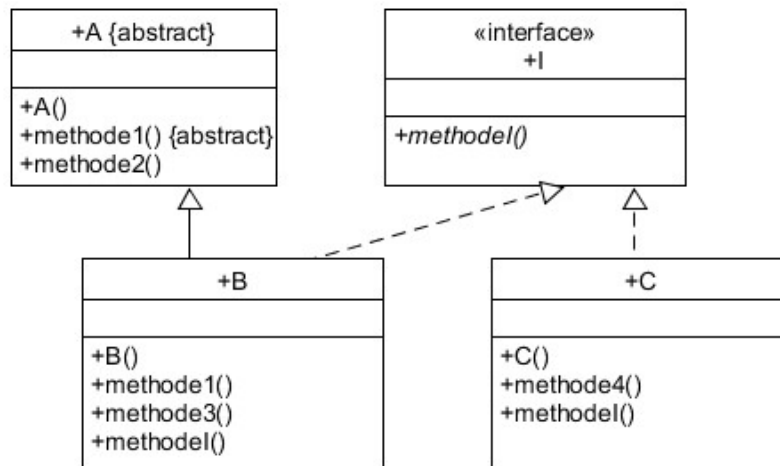
```
}
```

```
public class Manager extends Person  
{
```

```
}
```

Aufgabe 6 (Fehlersuche)**[10 Punkte]**

Gegeben sei das folgende UML-Klassendiagramm

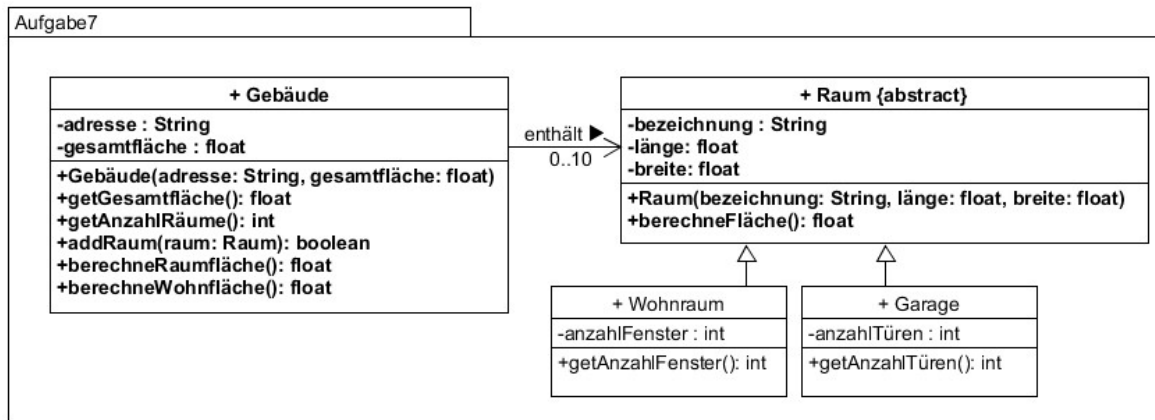


und die nachfolgende Test-Klasse. Markieren Sie fehlerhafte Programmzeilen in der Spalte Fehler mit einem X und begründen Sie für diese Zeilen, warum ein Fehler auftritt.

	Fehler	Begründung
public class Test {		
public static void test1(A einA) {		
einA.methode1();		
einA.methode2();		
einA.methodeI();		
}		
public static void test2(B einB) {		
einB.methode1();		
einB.methode2();		
einB.methodeI();		
}		
public static void test3(I einI) {		
einI.methode1();		
einI.methodeI();		
}		
public static void main(String[] args) {		
A einA;		
einA = new A();		
einA = new B();		
einA = new C();		
I einI;		
einI = new B();		
einI = new C();		
einI = new I();		
}		
}		

Aufgabe 7 (UML nach Java übertragen)**[16 Punkte]**

Gegeben sei das folgende UML-Klassendiagramm. Programmieren Sie die Klassen `Gebäude` und `Raum` vollständig in Java.



Hinweis: Außer `String` dürfen keine Klassen der Java-Klassenbibliothek verwendet werden!

Ergänzen Sie die Klasse `Gebäude` um Attribute zur Verwaltung der Assoziation. Ansonsten dürfen keine weiteren Attribute und Methoden hinzugefügt werden!

Die Methoden der Klasse `Gebäude` sollen folgendes leisten:

- `getGesamtfläche`: gibt die Gesamtfläche des Gebäudes zurück
- `getAnzahlRäume`: gibt die Anzahl der Räume zurück
- `addRaum`: fügt einen Raum ein, falls eine Position frei ist und zusätzlich die Gesamtfläche des Gebäudes noch ausreicht. Falls eingefügt werden konnte, ist der Rückgabewert `true`, sonst `false`.
- `berechneRaumfläche`: berechnet die Gesamtfläche aller zugeordneten Räume.
- `berechneWohnfläche`: berechnet die Gesamtfläche aller zugeordneten Wohnräume.

Die Methode `berechneFläche` der Klasse `Raum` soll die Fläche des Raums berechnen.

Aufgabe 8 (Speicherabbild)

[12 Punkte]

Zeichnen Sie das Speicherabbild zu folgendem Programmcode zum Zeitpunkt 1, d.h. vor der `return`-Anweisung der Methode `gleich`.

```

public class Test {
    public static void main
        (String[] args)
    {
        boolean erg;
        B b1 = new B(3,7);
        B b2 = new B(1,4);
        erg = b1.gleich(b2);
        System.out.print(erg);
    }
}

public class A {
    protected static int c=1;
    private int m;
    private int n;

    public A(int m, int n){
        this.m = m;
        this.n = n;
        c++;
    }
    public int getM(){
        return m;
    }
    public int getN(){
        return n;
    }
}

public class B extends A {
    private int m;

    public B(int m, int n){
        super(m, n);
        this.m = m + n;
        c++;
    }
    public boolean gleich(B b)
    {
        boolean erg =
            getM()+getN()+m ==
            b.getM()+b.getN()+b.m;

        // Zeitpunkt 1
        return erg;
    }
}

```

	Bytecode	
args	...	main

