

Klausur
Einführung in die Programmierung
22.07.2019 (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 Klausurvorgabe 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)	12	
2 (Ein-/Ausgabe, Kontrollstrukturen)	10	
3 (Programmieraufgaben)	11	
4 (Array-Programmierung)	12	
5 (Klassen und Objekte)	17	
6 (Fehlersuche)	10	
7 (UML nach Java übertragen)	16	
8 (Speicherabbild)	12	
<i>Bonuspunkte</i>	<i>10</i>	
Gesamtpunktzahl	100+10=110	

Name, Vorname, Matrikelnummer

Studiengang

Aufgabe 1 (Multiple Choice, Wissen)**[12 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 „154 ist nicht 15+4“?	Richtig	Falsch
<code>System.out.print(15+4+" ist nicht "+15+4);</code>		
<code>System.out.print("'" + 15+4+" ist nicht 15+4");</code>		
<code>System.out.print(15+4+" ist nicht 15+4");</code>		
<code>System.out.print(15+(4+" ist nicht 15+4"));</code>		

Bewerten Sie folgende Aussagen. Ein Objektattribut 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.		

- b) Geben Sie die Werte der Variablen x und y nach Ausführung der folgenden Java-Anweisungen an. Für jeden korrekten Wert erhalten Sie 0.5 Punkte.

Anweisungen	x	y
<code>int x=10; int y=0; for (x=0; x<10; x+=2) y--;</code>		
<code>int x=10; int y=0; for (x=0; x<100; x+=2) y++;</code>		
<code>int x=40; int y=0; while(x>0) { x /= 2; y += 2; }</code>		
<code>int x = 30; int y=11; do { x--; } while (x-- > y++);</code>		

- c) Berechnen Sie die Summe der beiden Hexadezimalzahlen $1F_{16}$ und AA_{16} . Geben Sie das Ergebnis in Binär- und Hexadezimaldarstellung an.

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

Ergänzen Sie die Klasse `GeradeWorte` um eine `main`-Methode, die 50 Strings von der Konsole einliest und anschließend zunächst jeden String mit gerader Länge und danach alle Strings ungerader Länge auf der Konsole ausgibt. Jeder dieser Strings soll in einer eigenen Zeile stehen.

Geben Sie notwendige `import`-Anweisungen an.

Hinweis: Die Länge eines Strings `wort` können Sie mit einem Aufruf von `wort.length()` bestimmen. Beispiel: `"Hallo".length()` gibt 5 zurück.

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

    }

}
```

Aufgabe 3 (Programmieraufgaben)**[11 Punkte]**

- a) Schreiben Sie eine Methode, die für eine gegebene Zahl n das maximale t berechnet, so dass n durch 3^t teilbar ist. Wir bezeichnen t mit $\text{faktoren}_3(n)$.

Beispiel: Die Zahl 99 ist durch $9 = 3^2$ aber nicht durch $27 = 3^3$ teilbar. Zur Zahl $n = 99$ soll also 2 zurückgegeben werden.

```
public static int faktoren3(int n)
{
    assert(n!=0);
```

```
}
```

- b) Schreiben Sie nun eine Methode, die zu der übergebenen Zahl n das Maximum unter den geraden Zahlen in der Menge $\{ \text{faktoren}_3(1), \text{faktoren}_3(2), \dots, \text{faktoren}_3(n) \}$ berechnet.

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

```
}
```

Aufgabe 4 (Array-Programmierung)**[12 Punkte]**

- a) Programmieren Sie die Methode `teilsommen`, die für ein beliebiges eindimensionales Feld ganzer Zahlen ein Element an der Position `i` durch die Summe der Elemente bis einschließlich zur Position `i` ersetzt. Beispiel:

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

vorher

1	8	21	16	18	45	56	60	111
---	---	----	----	----	----	----	----	-----

nachher

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

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

```
}
```

- b) Programmieren Sie die Methode `gitter`, die für ein beliebiges $n > 0$ ein quadratisches, zweidimensionales Feld mit $2n+1$ Zeilen und $2n+1$ Spalten erzeugt und jede zweite Zeile und Spalte mit dem Zeichen `*` füllt (Gittermuster). Die übrigen Elemente sollen Leerzeichen enthalten.

Beispiele für $n = 1$ und 2 :

	*	
*	*	*
	*	

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

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

```
}
```

Aufgabe 5 (Klassen und Objekte)**[17 Punkte]**

Betrachten Sie die folgende Klasse Fahrzeug:

```
public abstract class Fahrzeug {
    private int anzPersonen;
    private String treibstoffArt;

    public Fahrzeug(int anzPersonen, String treibstoffArt)
    {
        this.anzPersonen = anzPersonen;
        this.treibstoffArt = treibstoffArt;
    }

    public abstract double berechneVerbrauchPro100Km();

    protected double kmProLiter() {
        return 100/berechneVerbrauchPro100Km();
    }

    public String toString() {
        return "tankt " + treibstoffArt + " für "
            + anzPersonen + " Personen, ";
    }

    public static void main(String[] args)
    {
        Lkw brummiA = new Lkw();
        Lkw brummiB = new Lkw();
        EBus oekoA = new EBus(41);
        EBus oekoB = new EBus(39);
        EBus oekoC = new EBus(41);
        Pkw taxi = new Pkw(5);

        System.out.println("" + brummiA + brummiA.kmProLiter());
        System.out.println("" + brummiB + brummiB.kmProLiter());
        System.out.println("" + oekoA + oekoA.kmProLiter());
        System.out.println("" + oekoB + oekoB.kmProLiter());
        System.out.println("" + oekoC + oekoC.kmProLiter());
        System.out.println("" + taxi + taxi.kmProLiter());
    }
}
```

Ergänzen Sie die folgenden Klassen `EBus`, `Pkw` und `Lkw` um geeignete Methoden und maximal zwei Attribute pro Klasse, so dass

- ein `EBus` keinen Treibstoff (`null`) tankt und einen Verbrauch von 0 Litern pro 100 km besitzt,
- ein `Pkw` stets „Benzin“ tankt und einen Verbrauch von 5 Litern pro 100 km besitzt und
- ein `Lkw` stets „Diesel“ tankt und einen Verbrauch von 25 Litern pro 100 km besitzt.

Die Anzahl der Personen in einem `Bus` und einem `Pkw` ist jeweils bei der Erstellung von Objekten dieser Klassen im Konstruktor anzugeben. Ein `Lkw` besitzt immer genau 2 Personen.

Ein Aufruf der `main`-Methode der Klasse `Fahrzeug` soll hierbei folgende Ausgabe erzeugen:

```
Lkw 1 tankt Diesel für 2 Personen, 4.0
Lkw 2 tankt Diesel für 2 Personen, 4.0
Ein EBus tankt null für 41 Personen, 0.0
Noch ein EBus tankt null für 39 Personen, 0.0
Noch ein EBus tankt null für 41 Personen, 0.0
Der Pkw tankt Benzin für 5 Personen, 20.0
```

```
public class EBus extends Fahrzeug
{
```

```
}
```

```
public class Pkw extends Fahrzeug  
{
```

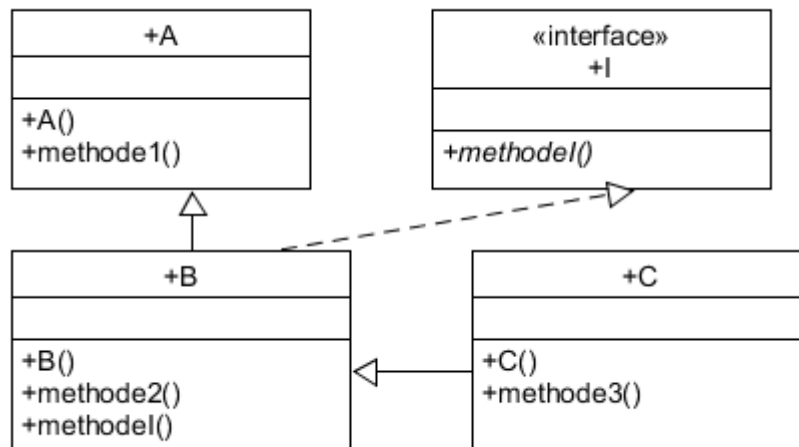
```
}
```

```
public class Lkw extends Fahrzeug  
{
```

```
}
```

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

Gegeben sei das folgende UML-Klassendiagramm



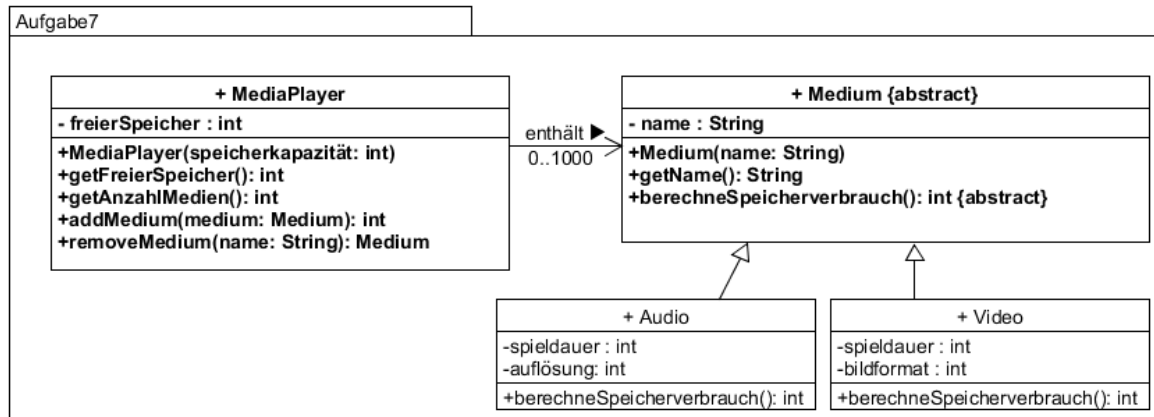
und die nachfolgende Test-Klasse. Markieren Sie fehlerhafte Programmzeilen in der Spalte Fehler mit einem X und begründen Sie für diese Zeilen, warum ein Fehler auftritt.

Hinweis: Falsche Markierungen führen zu einem Punktabzug.

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

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

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



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

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

Die Methoden der Klasse `MediaPlayer` sollen folgendes leisten:

- `getFreierSpeicher`: gibt jeweils den aktuellen freien Speicher zurück.
- `getAnzahlMedien`: gibt die Anzahl der gespeicherten Medien zurück
- `addMedium`: fügt ein `Medium` hinzu, falls eine Position frei ist und zusätzlich der freie Speicher noch ausreicht. Falls eingefügt werden konnte, wird der Index, an dem eingefügt wurde, zurückgegeben, sonst `-1`.
- `removeMedium`: entfernt das `Medium` mit dem kleinsten Index, dessen Name übereinstimmt, aus dem `MediaPlayer`. Falls ein passendes `Medium` gefunden wurde, wird das entfernte `Medium` als Ergebnis zurückgegeben, sonst `null`.

Name, Vorname, Matrikelnummer

Studiengang

Name, Vorname, Matrikelnummer

Studiengang

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;
        A a = new A(3,7);
        B b = new B(3,7);
        erg = a.gleich(b);
        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 boolean gleich(A a)
    {
        boolean erg =
            (m == a.getM()) &
            (n == a.getN());

        // Zeitpunkt 1
        return erg;
    }
}

public class B extends A {
    private int m;

    public B(int m, int n){
        super(n, m);
        this.m = m + n;
    }
}

```

	Bytecode	
args	...	main

Name, Vorname, Matrikelnummer

Studiengang