

Klausur A
Einführung in die Programmierung
21.02.2022 (WiSe 2021/22)

Hinweise

1. Bearbeitungszeit: 90 Minuten
2. Zum Bestehen der Klausur sind 50 Punkte erforderlich.
3. Erlaubte Hilfsmittel: keine
4. Notieren Sie Ihre Lösungen auf eigenem Papier. Versehen Sie jedes Blatt mit Namen, Matrikelnummer, Studiengang und Seitenzahl. **Geben Sie bei jeder Lösung die Aufgabennummer und Teilaufgabe an.** Bitte geben Sie nicht die Lösungen von zwei verschiedenen Aufgaben auf einem Blatt an.
5. Bitte schreiben Sie deutlich mit einem schwarzen Stift.
6. Alle vorgegebenen und zu erstellenden Programmtexte beziehen sich auf die Programmiersprache Java.

Viel Erfolg!

Aufgabe	Max. Punktzahl	Erreichte Punktzahl
1 (Zahlendarstellung)	8	
2 (Programmieraufgabe)	30	
3 (Programmieraufgabe)	20	
4 (UML nach Java übertragen)	27	
5 (Speicherabbild)	15	
<i>Bonuspunkte</i>	<i>10</i>	
Gesamtpunktzahl	100+10=110	

Aufgabe 1 (Zahlendarstellung)**[8 Punkte]**

Beschreiben Sie am Beispiel der Zahl $n = -47,5$ den Aufbau von Fließkommazahlen. Warum gibt es Zahlen, die sich in Dezimalbruchdarstellung exakt darstellen lassen, als Fließkommazahl in Java aber nur als Näherung?

Aufgabe 2 (Programmieraufgabe)**[30 Punkte]**

Implementieren Sie eine Klasse **TeilwortLöschen** mit den folgenden Methoden:

i) **boolean istTeilwortAnIndex(String w, String v, int i)**

Es wird überprüft, ob v ein Teilwort von w ist, das am Index i beginnt.

Bspl.: Wird für w das Wort "abcdef", für v das Wort "bcde" und für i die Zahl 1 übergeben, so soll true zurückgegeben werden, da "bcde" ein Teilwort von w ist, das an Index 1 beginnt. Wird hier für v das Wort "cef" übergeben, so soll false zurückgegeben werden.

ii) **int findeTeilwort(String w, String v)**

Wenn v ein Teilwort von w ist, so soll der kleinste Index zurückgegeben werden, an dem das Teilwort v beginnt. Sonst soll -1 zurückgegeben werden.

Bspl.: Wird für w das Wort "abcdeabf" und für v das Wort "ab" übergeben, so soll 0 zurückgegeben werden, da das Teilwort "ab" in w an den Indizes 0 und 5 steht und 0 der kleinere Index ist.

iii) **String löscheTeilwort(String w, String v)**

Wenn das Wort v ein Teilwort von w ist, so wird **ein** Vorkommen von v in w gelöscht und das resultierende Wort zurückgegeben. Sonst wird null zurückgegeben.

Bspl.: Wird für w das Wort "abcdef" und für v das Wort "cdef" übergeben, so soll "ab" zurückgegeben werden.

iv) **String löscheAlleTeilworte(String w, String[] vArray)**

In dem Wort w sollen Teilworte aus dem Feld vArray gelöscht werden, bis keines dieser Worte mehr als Teilwort in w vorkommt. Die Reihenfolge des Löschens ist dabei egal. Das resultierende Wort soll zurückgegeben werden.

Bspl.: Wird für w das Wort "ababcf" und für vArray das Feld { "abc", "bf" } übergeben, so kann man aus w zunächst das Teilwort abc löschen und erhält somit das Wort "abf". Hierin kann dann das Teilwort "bf" gelöscht werden und man erhält somit das Wort "a". Hierin kann man nichts mehr löschen und somit wäre "a" eine korrekte Rückgabe.

Sie dürfen aus der Java-Bibliothek nur die Methoden *equals()*, *length()* und *charAt()* auf Strings benutzen. Für einen String w gibt die Methode *w.length()* die Länge von w und die Methode *w.charAt(i)* das i-te Zeichen von w zurück.

Aufgabe 3 (Programmieraufgabe)**[20 Punkte]**

- i) Schreiben Sie eine Methode `int[][] erzeugeFeldEinfach(int n, int m)`, die ein zweidimensionales Feld mit n Zeilen und m Spalten erzeugt, dieses Feld mit dem folgenden Muster füllt und zurückgibt:

0	0	1	1	0	0
0	0	1	1	0	0
1	1	0	0	1	1
1	1	0	0	1	1
0	0	1	1	0	0

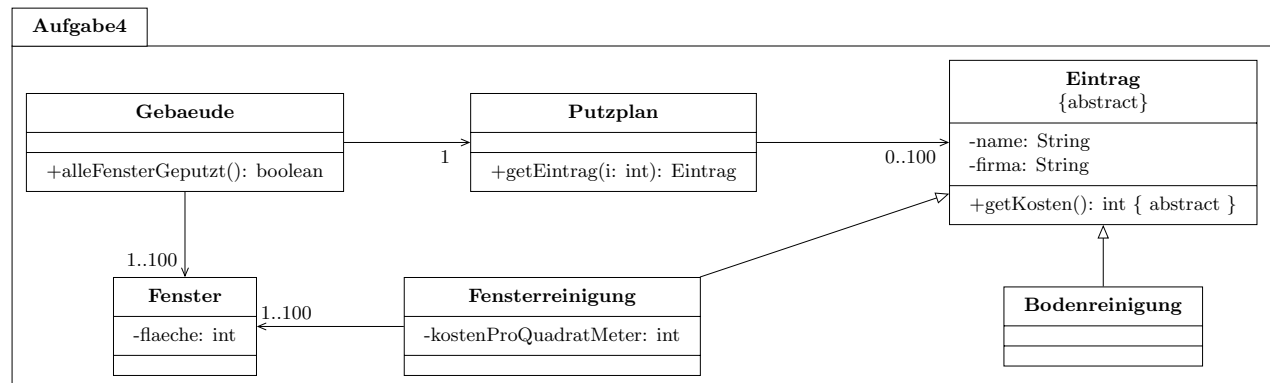
D.h. es sollen alternierend 2x2 Blöcke mit 0 und 1 gefüllt werden, wobei die 0. und 1. Zeile mit einem Block mit 0 beginnt, die 2. und 3. Zeile mit einem Block von 1 usw. Sie dürfen die Methode `erzeugeFeld` aus Teil ii) **nicht** verwenden!

- ii) Schreiben Sie nun eine Methode `int[][] erzeugeFeld(int n, int m, int s, int t)`, die wie oben ein Feld mit n Zeilen und m Spalten erzeugt, es mit alternierenden Blöcken mit 0 und 1 füllt (s.o.) und zurückgibt. Diesmal sollen die einzelnen Blöcke aber s Zeilen und t Spalten besitzen. Mit s=2 und t=2 erhält man also wieder das Muster aus Teil i).

Sie dürfen **keine** Klassen- oder Objektattribute oder Methoden aus der Java-Klassenbibliothek benutzen. Sie können aber weitere eigene (Hilfs-)Methoden hinzufügen.

Aufgabe 4 (UML nach Java übertragen)**[27 Punkte]**

Gegeben sei das folgende UML-Klassendiagramm.

Hinweis: Es dürfen **keine** Klassen der Java-Klassenbibliothek verwendet werden!

Programmieren Sie die angegebenen Klassen wie folgt:

1. Implementieren Sie in der Klasse **Gebaeude** zunächst **nur** die **Attribute**, die zur Verwaltung der Assoziationen zwischen den Klassen **Gebaeude** und **Fenster** sowie den Klassen **Gebaeude** und **Putzplan** notwendig sind.
2. Implementieren Sie in der Klasse **Putzplan** **nur** die **Attribute** und die im UML-Diagramm vorgegebene **get-Methode**, die zur Verwaltung der Assoziationen zwischen den Klassen **Putzplan** und **Eintrag** notwendig sind.
3. Implementieren Sie die Klassen **Eintrag**, **Fensterreinigung**, **Bodenreinigung** und **Fenster** **vollständig** in Java. Ergänzen Sie notwendige Attribute und Methoden, insbesondere sinnvolle Konstruktoren und alle Attribute und Methoden zur Verwaltung der Assoziation zwischen den Klassen **Fensterreinigung** und **Fenster**. Die Methode **getKosten** in der Klasse **Fensterreinigung** gibt die Summe der Kosten für alle in diesem Eintrag angegebenen Fenster zurück. Für ein Fenster berechnen sich die Kosten aus der Fläche des Fensters multipliziert mit den Kosten pro Fläche. Eine **Bodenreinigung** kostet immer 1570.
4. Implementieren Sie die Funktion **AlleFensterGeputzt** der Klasse **Gebaeude**, die überprüft, ob alle Fenster des Gebäudes geputzt wurden, d.h. in einem der Einträge des Putzplanes vorkommen.

Hinweis: Ein Fenster ist nur dann in einem Eintrag des Putzplans, wenn es sich um eine Fensterreinigung handelt.

Aufgabe 5 (Speicherabbild)**[15 Punkte]**

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

```
public class Test {  
    public static void main(String[] args) {  
        int erg;  
        A b1 = new B(1);  
        B b2 = new B(1);  
        erg = b1.tuWas(b1.m, b2.m);  
    }  
}
```

```
public class A {  
    protected static int a=2;  
    public int m;  
  
    public A(int m){  
        this.m = m;  
        a += 2;  
    }  
    public void tuWas(int a) {  
        m = a;  
        if (a>0) tuWas(a-1);  
    }  
}
```

```
public class B extends A {  
    public int m;  
    public int n;  
  
    public B(int m){  
        super(2*m);  
        this.m = m+2;  
        a++;  
    }  
    public void tuWas(int a) {  
        m = a + 2;  
    }  
    public int tuWas(int a, int b) {  
        int erg = b;  
        super.tuWas(a+1);  
  
        // Zeitpunkt 1  
        return erg;  
    }  
}
```

Name, Vorname, Matrikelnummer

Studiengang