

Name:	Studiengang: <input type="checkbox"/> B.A. <input type="checkbox"/> M.A.
Vorname:	Matrikelnummer:
Studienfächer:	Fachsemester:

Allgemeine Hinweise:

1. Überprüfen Sie bitte, ob Sie alle Seiten der Klausurangabe vollständig erhalten haben
(Gesamtzahl: **10**)
2. **Bearbeitungszeit: 90 Minuten**, maximal erreichbare **Punktzahl: 48**. Die jeweils erreichbare Punktzahl ist bei jeder Frage angegeben. Bitte teilen Sie Ihre Arbeitszeit entsprechend ein.
3. Denken Sie daran, die Daten oben einzutragen, **bevor** Sie mit der Bearbeitung beginnen.
4. Verwenden Sie für die Beantwortung aller Fragen diese Klausurangabe. Sie können jederzeit auch die Rückseiten beschreiben, falls der Platz auf der Vorderseite nicht ausreichen sollte. Bitte geben Sie in jedem Fall an, auf welche Frage sich die Lösung jeweils bezieht. Bei Multiple-Choice-Fragen treffen Sie bitte die Auswahl Ihrer Antworten ebenfalls auf der Klausurangabe.
5. Benutzen Sie keine Bleistifte, keine rot schreibenden Stifte und kein TippEx, o.ä.
6. Zugelassene Hilfsmittel: **keine**
7. Mobiltelefone sowie Computer am Arbeitsplatz - auch ausgeschaltet - sind **nicht zugelassen**.
8. Bitte legen Sie Lichtbildausweis und Studierendenausweis gut sichtbar vor sich, damit Ihre Identität möglichst störungsfrei überprüft werden kann.
9. Geben Sie keine mehrdeutigen (oder mehrere) Lösungen an. In solchen Fällen wird stets die Lösung mit der geringeren Punktzahl gewertet. Eine richtige und eine falsche Lösung ergeben also 0 Punkte.
10. Wenden Sie sich bei Unklarheiten in den Aufgabenstellungen immer an die Aufsichtsführenden. Hinweise und Hilfestellungen werden dann, falls erforderlich, offiziell für alle Teilnehmer durchgegeben.

Aufgabe 1 - Google Java Style Guide**(3 Punkte)**

Markieren Sie in folgender Tabelle, welche Schreibweise für welche Art von Name gemäß dem Style Guide erlaubt (✓) bzw. nicht erlaubt (✗) ist. Nach vollständiger Bearbeitung ist kein Feld mehr leer.

	Klasse	Methode	Konstante	Feld (nicht konstant)	Parameter	lokale Variable	Typ-Variable
UpperCamelCase							
lowerCamelCase							
UPPER_CASE							

Aufgabe 2 - Idiome**(3 Punkte)**

Nennen sie drei positive Eigenschaften einer Klasse, die das Immutable-Idiom erfüllt.

Aufgabe 3 - Code Smells / Refactoring**(2+2+2 Punkte)**

Führen Sie am folgenden Code drei von der Art her unterschiedliche Verbesserungen hinsichtlich Verständlichkeit bzw. Lesbarkeit durch. Die Änderungen dürfen nötigenfalls aufeinander aufbauen. Begründen Sie jeweils kurz, warum es sich um eine Verbesserung handelt.

```
1 public static boolean doStuff (int i, int j) {
2     int s1 = 0; int a = 0;
3     int a1 = Math.abs(i);
4     int a2 = Math.abs(j);
5     String tempI = Integer.toString(a1);
6     int[] arr1 = new int[tempI.length()];
7     for (int k = 0; k < tempI.length(); k++) {
8         arr1[k] = tempI.charAt(k) - '0';
9     }
10    String tempJ = Integer.toString(a2);
11    int[] array2 = new int[tempJ.length()];
12    for (int k = 0; k < tempJ.length(); k++) {
13        array2[k] = tempI.charAt(k) - '0';
14    }
15    for (int k = 0; k < arr1.length; k++) {
16        s1 = s1 + arr1[k];
17    }
18    for (int k = 0; k <= array2.length; k++) {
19
20        a = a + array2[k];
21    }
22    if (a == s1) {
23        return true;
24    } else {
25        return false;
26    }
27 }
```


Aufgabe 4 - Java Collections Framework**(1+1+1+1+1 Punkte)****Container**

- a) Was ist der Unterschied zwischen FIFO- und LIFO-Verhalten?

- b) Beschreiben sie den wichtigsten konzeptuellen Unterschied zwischen List und Collection.

- c) Beschreiben sie den wichtigsten konzeptuellen Unterschied zwischen Set und Collection.

equals, hashCode, compareTo

- d) Was muss gelten, damit die von einem Comparator c erzeugte Ordnung auf einer Menge M von Elementen als "konsistent zu equals" gilt?

- e) Was sollte jede Implementierung der Methode hashCode bezüglich der Methode equals gewährleisten?

Aufgabe 5 - Komplexitätsanalyse**(2+3+3+7 Punkte)**

- a) Sortieren Sie die folgenden neun Komplexitätsgrade nach aufsteigender Wachstumsgeschwindigkeit. Dabei ist c eine Konstante > 1 und n der wachsende Parameter.

$O(n^{1/c})$	$O(n^3)$	$O(n \cdot \log(n))$
$O(n)$	$O(1)$	$O(c^n)$
$O(\log(\log(n)))$	$O(\log(n))$	$O(n^2)$

Gegeben sei für die restlichen Teilaufgaben der Komplexitätsanalyse folgender Code:

```
1 void quicksort(int[] a) {
2   quicksort(a, 0, a.length - 1);
3 }
4
5 void quicksort(int[] a, int lo, int hi) {
6   if (lo < hi) {
7     int p = partition(a, lo, hi);
8     quicksort(a, lo, p - 1);
9     quicksort(a, p + 1, hi);
10  }
11 }
12
13 int partition(int[] a, int lo, int hi) {
14   int pivot = a[hi];
15   int i = lo - 1;
16   for (int j = lo; j < hi; j++) {
17     if (a[j] < pivot) {
18       i = i + 1;
19       swap(a, i, j);
20     }
21   }
22
23   swap(a, i + 1, hi);
24   return i + 1;
25 }
26
27 void swap(int[] a, int x, int y) {
28   int temp = a[x];
29   a[x] = a[y];
30   a[y] = temp;
31 }
```


Aufgabe 6 - Planning Poker**(2 Punkte)**

Planning Poker ist eine Heuristik, um den Aufwand von Aufgaben im Bereich weniger Personentage genauer abzuschätzen, kurz bevor deren Bearbeitung beginnt. Beschreiben sie kurz das Vorgehen, nennen sie dabei einzuhaltende Grundregeln und was mit deren Einhaltung erreicht werden soll.

Aufgabe 7 - Semantic Versioning**(3 Punkte)**

Welche Information kann der Benutzer einer Software-Bibliothek, die sich an "Semantic Versioning 2.0.0" hält, aus der Veränderung der Versionsnummer gewinnen?
Beantworten Sie die Frage, indem Sie die Interpretation der drei Haupt-Versionsnummern (Major, Minor, Patch) und wie diese bei einer Code-Änderung zu verändern sind, beschreiben.

