Test-Klausur zur Veranstaltung

Objektorientierte Programmierung

im Sommersemester 2024

universität ulm

22. Mai 2024, 12:30 Uhr

Bearbeitungszeit: 30 min

Florian Sihler, Raphael Straub,
Prof. Matthias Tichy

Institut für Softwaretechnik und Programmiersprachen

Nachname:	Vorname:	Matrikelnummer:
Studiengang und angestrebter Abschluss (B. Sc./M. Sc.):		Fachsemester:

Hiermit erkläre ich, dass ich Probeklausur-prüfungsfähig bin.
Sollte ich nicht auf der Liste der ~~angemeldeten~~ Studierenden aufgeführt sein, die die Probeklausur wahrnehmen wollen, dann nehme ich hiermit zur Kenntnis, dass diese Prüfung ~~nicht gewertet werden~~ sinnvoll sein wird.

_____

Datum, Unterschrift

Name Ihres Tutors:

Lassen Sie dieses Feld leer, wenn Sie keinen Tutor haben.

Zur allgemeinen Beachtung:

- Füllen Sie das Deckblatt vollständig und korrekt aus.
- Lesen Sie sich zunächst die Klausur sorgfältig durch (sie besteht aus **6** Seiten).
- Bearbeiten Sie die Aufgaben direkt auf den Aufgabenblättern.
- Es sind keine Hilfsmittel erlaubt.
- Aufgaben, welche nicht mit einem dokumentenechten Stift in den Farben blau oder schwarz bearbeitet worden sind, werden nicht bewertet.

Zusätzliches benötigtes Paper wird Ihnen von der Aufsicht zur Verfügung gestellt.

Punkteverteilung					
1 von 6	2 von 10	3 von 5	4 von 5	$\sum$ von 26	**Note**
					Korrektur

Aufgabe 1 — Wissensfragen

2 + 2 + 2 = 6 Punkte

Kreuzen Sie zu jeder Frage die korrekte(n) Antwortmöglichkeit(en) an. Gehen Sie bei gegebenen Code-Schnipsel davon aus, dass diese jeweils korrekt in umliegenden Code eingebettet sind, also beispielsweise in der `main`-Methode einer Klasse stehen.

a) Gegeben sei der folgende Code. Es gibt nur *eine* korrekte Antwort: /2

```
1 var a = 35;
```

- ◯ Der Typ der Variable a ist `char`.
- ◯ Der Typ der Variable a ist `long`.
- ◯ Der Typ der Variable a ist `byte`.
- ◯ Der Typ der Variable a ist `String`.
- ◯ Der Typ der Variable a ist `int`.
- ◯ In Java gibt es *keine* Typinferenz.

b) Betrachten Sie erneut folgenden Code, es gibt nur *eine* korrekte Antwort: /2

```
1 byte[] i = {1, 2, 3};
2 do {
3    System.out.print(i[i.length - 1]);
4    i[i.length - 1] = 0;
5 } while (i.length < 3);
```

- ◯ Der Code erzeugt die Ausgabe „123“.
- ◯ Der Code erzeugt die Ausgabe „3“.
- ◯ Der Code erzeugt die Ausgabe „321“.
- ◯ Der Code erzeugt die Ausgabe „1“.
- ◯ Der Code erzeugt die Ausgabe „300“.
- ◯ Der Code erzeugt einen Kompilierfehler.

c) Kreuzen Sie alle Aussagen an, die für Java korrekt sind (mehrere Antworten möglich): /2

- ☐ Variablen sind in Java standardmäßig unveränderlich.
- ☐ Arrays werden in Java auf dem Heap abgelegt.
- ☐ Java hat einen Garbage Collector.
- ☐ Das Liskovsche Substitutionsprinzip findet in Java *keine* Anwendung.
- ☐ Das Typsystem von Java macht Tests überflüssig.
- ☐ Alle Probleme die wir mit Schleifen lösen können, können wir auch mit Rekursion lösen.
- ☐ Abstrakte Klassen können in Java mit dem Schlüsselwort **new** instanziiert werden.

/6

Aufgabe 2 — Imperative Programmierung **5 + 5 = 10 Punkte**

Sie dürfen für die folgenden Aufgaben Hilfsfunktionen implementieren!

/5

a) Implementieren Sie eine Funktion `int getMaximum(int[])`, die das Maximum eines Arrays von ganzen Zahlen zurückgibt. Sie dürfen davon ausgehen, dass das Array mindestens ein Element enthält.

Beispiele:

- `getMaximum(new int[] {1, 2, 3, 4, 5})` gibt 5 zurück.
- `getMaximum(new int[] {3, 2, 42, 1, -3})` gibt 42 zurück.

b) Schreiben Sie eine Funktion `printDiamond(int n)`, die ein Diamantmuster der Größe `n` auf der Konsole ausgibt. Das Muster soll aus Sternen („*“) bestehen. Sie dürfen davon ausgehen, dass `n` immer ≥ 1 ist.

/5

Beispiel:

- `printDiamond(3)` soll folgendes Muster ausgeben (Leerfelder werden hier der Übersichtlichkeit wegen mit einem Punkt „`.`“ dargestellt):

```
...*
..***
.*****
..***
...*
```

/10

Aufgabe 3 — Fehler finden **5 Punkte**

Markieren Sie in folgendem Code alle Kompilier- und Laufzeitfehler und verwenden Sie die Zeilen unten um kurz zu beschreiben, was der Fehler ist **und** wie man diesen korrigieren kann.

In der Datei `Errors.java`:

```
1  class Errors {
2     public String foo() { return "foo"; }
3
4     public int double(int x) { return (2*)x; }
5
6     public static void main() {
7        Errors e = new MegaErrors();
8        System.in.print(e.foo());
9     }
10 }
```

In der Datei `MegaErrors.java`:

```
1  class MegaErrors implements Errors {
2     public String foo(int i) {
3        return "foo-" + i;
4     }
5  }
```

Beachten Sie, dass die Anzahl der Zeilen nicht mit der Anzahl der Fehler übereinstimmen muss und Ihnen mehr Platz zur Verfügung stehen kann.

- _____
- _____
- _____
- _____
- _____
- _____

/5

Aufgabe 4 — Generische Freuden

2 + 1 + 2 = 5 Punkte

Machen Sie sich zunächst mit dem folgenden Code vertraut:

```
1 public class Pair<F, S> {
2    public final F fst;
3    public final S snd;
4
5    public Pair(F fst, S snd) {
6       this.fst = fst;
7       this.snd = snd;
8    }
9 }
```

Geben Sie im Folgenden jeweils nur die Ausdrücke an, die notwendig sind um die jeweilige Aufgabe zu lösen. Folgefehler werden Ihnen dabei nicht zur Last gelegt. Die Verwendung von sogenannten „Raw Types“ ist in dieser Aufgabe allerdings nicht erlaubt, geben Sie die *passendsten* Typen vollständig an.

a) Erstellen Sie eine Instanz von `Pair` mit *`"Hello"`* als erstem und `42` als zweitem Element und speichern Sie diese in einer Variable `p`. /2

b) Geben Sie den Wert des ersten Elements von `p` aus. /1

c) Erzeugen Sie eine Kopie von `p` und speichern Sie diese in einer neuen Variable `p2`. /2

/5