Philipps-Universität Marburg
Fachbereich Mathematik und Informatik
AG Programmiersprachen und Werkzeuge

Wintersemester 2021/2022
13.1.2022

Prof. Dr. Christoph Bockisch
MSc. Steffen Dick

Klausur zur Vorlesung

Objektorientierte Programmierung

Wichtige Hinweise:

- Schalten Sie, falls noch nicht geschehen, umgehend ihr Mobiltelefon aus!
- Schalten Sie außerdem alle nicht medizinisch notwendigen Lärmquellen aus.
- Entfernen Sie jetzt alle unerlaubten Gegenstände vom Tisch. Erlaubt sind nur ein Stift (kein Rot-, Grün- oder Bleistift), Getränke und ein DinA4-Blatt mit Notizen. Halten Sie außerdem Ihren Studentenausweis, so wie Ihren Personalausweis oder Reisepass bereit.
- Schreiben Sie auf jedes Blatt Ihren Namen und Matrikelnummer. Blätter ohne Namen werden nicht korrigiert und ergeben 0 Punkte! Füllen Sie insbesondere die folgende Tabelle in Druckbuchstaben aus:

Vorname	
Nachname	
Matrikelnummer	
Studienfach	
Angestrebter Abschluss	

- Die Bearbeitungszeit beträgt **1** Zeitstunde.
- Verwenden Sie kein eigenes Papier für Notizen. Am Ende der Klausur befinden sich 3 Extrablätter. Sie können auf Anfrage weitere Blätter erhalten. Machen Sie gut kenntlich, wenn Sie Zusatzblätter für Lösungen verwenden und tragen Sie dort ebenfalls Namen und Matrikelnummer ein.
- Außer Ihrem handbeschriebenen DinA4-Blatt sind keine weiteren Hilfsmittel erlaubt. Zuwiderhandlung zieht einen Ausschluss von der Klausur nach sich.
- Mehrere, widersprüchliche Lösungen zu einer Aufgabe werden mit 0 Punkten bewertet.
- Sollten Sie eine Frage haben, wenden Sie sich bitte leise an die Tutor*innen.

Wird bei der Korrektur ausgefüllt:

Aufgabe	1	2	3	4	Gesamt
Punkte	10	15	12	13	50
Erreichte Punkte					

Vorname:	Nachname:	Matrikelnummer:

Aufgabe 1: Wissensfragen — 10 Punkte

Im Folgenden stehen einige Aussagen. Ist die Aussage korrekt, machen Sie ein Kreuz in der Spalte *Wahr*. Sollte die Aussage falsch sein, machen Sie ein Kreuz in der Spalte *Falsch*.

Erklären Sie korrekte Aussagen und berichtigen Sie falsche Aussagen mit einem kurzen Satz.

Aussage	Wahr	Falsch
In Java darf eine lokale Variablen nicht denselben Namen haben wie ein Feld in der gleichen Klasse.		
Java-Code wird nicht direkt in nativen Maschinen-Code, sondern in Java-Byte-Code übersetzt, der dann in einer virtuellen Maschine ausgeführt wird.		
Innerhalb einer Methode mit dem Rückgabetypen `void` kann das Schlüsselwort `return` verwendet werden, um eine Methode vorzeitig abzubrechen.		
Die Determiniertheit eines Algorithmus bezieht sich auf den Prozess eines Algorithmus. Ein Algorithmus ist also genau dann determiniert, wenn es zu jeder möglichen Eingabe genau einen Prozess gibt.		
Der Begriff *Terminierung* aus der Informatik kommt vom Insekt *Termite*, da diese in frühen Computern dafür gesorgt haben, dass Programme vorzeitig beendet wurden.		

Vorname:	Nachname:	Matrikelnummer:

Aufgabe 2: Fehlersuche — 15 Punkte

In jeder der folgenden Teilaufgaben verstecken sich jeweils **2 Fehler**. Markieren Sie jeden der Fehler, beschreiben Sie Ihn kurz und geben Sie einen Lösungsvorschlag an. Geben Sie jeweils an: die Zeilennummer des Fehlers, die Art des Fehlers (Compilerfehler[1], semantischer Fehler, oder ob eine Exception geworfen wird) und eine kurze Erklärung inklusive korrigiertem Code wie der Fehler zu beheben ist.

Sie können davon ausgehen, dass nur gültige Argumente an die Methoden übergeben werden, die nicht weiter geprüft werden müssen. Außerdem existieren alle aufgerufenen Methoden.

a) Eine Methode, die alle Vorkommen eines Wertes in einem Array ersetzt und zurückgibt, ob eine Ersetzung vorgenommen wurde: — 5

```
1  boolean replace(Object[] arr, Object oldV, Object newV) {
2      boolean replaced;
3      for (int i = 0; i <= arr.length; i++) {
4          if (arr[i] == oldV) {
5              arr[i] = newV;
6              replaced = true;
7          }
8      }
9      return replaced;
10 }
```

b) Eine Methode, die zählt, wie oft der angegebene Wert in einem Array vorkommt: — 5

```
1  int count(Integer[] arr, int v) {
2      long count = 0;
3      Integer searchValue = new Integer(v);
4      for (Integer entry : arr) {
5          if (entry == searchValue) {
6              ++count;
7          }
8      }
9      return count;
10 }
```

[1] Der Compiler erkennt Fehler in der Syntax und falsche Verwendung von Typen.

Vorname:	Nachname:	Matrikelnummer:

c) Eine Methode, die jedes Zeichen in einem String um n in der Ascii-Tabelle verschiebt: 5

```
1  String shiftString(String word, int n){
2    String returnString = "";
3    for(int i = 0; i < word.length; i++){
4      returnString = ((char) (word.charAt(i) + n)) + returnString;
5    }
6    return returnString;
7  }
```

Vorname:	Nachname:	Matrikelnummer:

Aufgabe 3: Rekursion

12 Punkte

6

a) Gegeben Sei die folgende Methode `mc91taux`. Schreiben Sie `mc91taux` in eine rekursive Methode `mc91tauxRec` um, die sich genau so verhält, wie die Untenstehende. Sie dürfen davon ausgehen, dass lediglich natürliche Zahlen an die Methode übergeben werden und dass die Methode immer terminiert.

```
1  static int mc91taux(int n, int c) {
2    while (c != 0) {
3      if (n > 100) {
4        n -= 10;
5        c -= 1;
6      } else {
7        n += 11;
8        c += 1;
9      }
10   }
11   return n;
12 }
```

Vorname:	Nachname:	Matrikelnummer:

Betrachten Sie die folgenden Funktionen und geben Sie jeweils an, um welche Art der Rekursion (baumartig/linear) es sich handelt. Begründen Sie Ihre Antwort kurz und geben Sie alle ausschlaggebenden rekursiven Aufrufe (Zeilennummer) mit an.

b) 2

```
1  double fak(int n){
2    if(n < 1){
3      return 1;
4    }
5    return fak(n - 1) * n;
6  }
```

c) 2

```
1  boolean isPalindrom(String word){
2    if(word.length < 2){
3      return true;
4    }
5    if(word.charAt(0) == word.charAt(word.length - 1)){
6      return isPalindrom(word.substring(1, word.length() - 1));
7    }
8    return false;
9  }
```

d) 2

```
1  boolean hasValue(Part part, int value){
2    if(part == null){
3      return false;
4    }
5    if(part.value == value){
6      return true;
7    }
8    return hasValue(part.left, int value)
9          || hasValue(part.right, int value);
10 }
```

Vorname:	Nachname:	Matrikelnummer:

Aufgabe 4: Programmierung — 13 Punkte

In dieser Aufgabe sollen Sie eine einfach verkettete Liste implementieren, die ausschließlich `int` Werte enthält. Verwenden Sie dabei weder Collections, falls Ihnen diese bereits bekannt sein sollten, noch Arrays.

Implementieren Sie zunächst eine Klasse `IntegerListElement`.

a) Geben Sie die vollständige Definition der Klasse `IntegerListElement` an. Diese Klasse soll zwei Felder besitzen: Eines, welches den zu speichernden Wert eines Elements enthält (`value`) und eines, was das nächste Element der Liste enthält (`next`). Implementieren Sie die zwei Methdoen `getNext()` und `getValue()`, welche die Werte des jeweiligen Feldes zurückliefern. Implementieren Sie außerdem zwei sinnvolle Konstruktoren. Wählen sie überall dort wo es möglich ist, sinnvolle Sichtbarkeitsmodifikatoren. — 3

Sie dürfen im Folgenden auf `IntegerListElement` mittels `ILE` verweisen.

Implementieren Sie die Klasse `IntegerList`.

b) Geben Sie die Definition der Klasse `IntegerList` an. Eine Liste hat ein Feld `head`, das das erste Element einer Liste repräsentiert. Bei einer leeren Liste soll `head` auf `null` gesetzt sein. Implementieren Sie einen sinnvollen Konstruktor. — 1

c) `boolean isEmpty()`, die zurückgibt, ob eine Liste leer ist. — 1

d) `void prepend(int value)`, die `value` an den Anfang der Liste anhängt. — 2

e) `int indexOf(int value)`, die die erste Position eines Wertes in der Liste zurückliefert. Wenn der Wert nicht enthalten ist, soll `-1` zurückgeliefert werden. — 3

f) Schreiben jeweils einen JUnit Test, der überprüft, dass: (1) eine neu erzeugte Liste leer ist, (2) eine Liste, der ein Wert vorne angehängt wurde, nicht leer ist und (3) der Index eines zuvor vorne angehängten Wertes `0` ist. — 3

Hinweis: Sie können davon ausgehen, dass die dafür nötigen Importe bereits getätigt wurden.

Implementierung von `IntegerListElement`:

Vorname:	Nachname:	Matrikelnummer:

Implementierung von `IntegerList`:

Vorname:	Nachname:	Matrikelnummer:

Implementierung der Tests:

Vorname:	Nachname:	Matrikelnummer:

Extra Seite

Vorname:	Nachname:	Matrikelnummer:

Extra Seite

Vorname:	Nachname:	Matrikelnummer:

Extra Seite