Ludwig-Maximilians-Universität München
Institut für Informatik
Prof. Dr. Peer Kröger
Ludwig Zellner, Florian Richter,
Andrea Maldonado, Nadine Schüler

WS 2019/20
15.12.2020

Wiederholungsprüfung
Einführung in die Programmierung

Allgemeine Hinweise:

- **Laden Sie sich zusätzlich zu dieser Angabe die Textdatei mit dem Titel "loesungsbogen.txt" als Vorlage für Ihre Abgabe herunter, in die Sie Ihre Lösungen zu den Aufgaben der Prüfung eintragen.**
- **Nutzen Sie bitte ausschließlich die .txt-Datei ("loesungsbogen.txt"), um ihre Lösung abzugeben.**
- **Die Art und Weise, eine Aufgabe zu lösen, wird ihnen detailliert auf der Angabe beschrieben. Lesen Sie diese daher bitte genau.**
- **Sie bekommen ab Beginn der Prüfung 180 Minuten Zeit, diese herunterzuladen, zu bearbeiten und Ihre Lösung wieder hochzuladen (alles via Uni2work). Somit haben Sie 90 Minuten extra Zeit, um auf eventuelle technische Probleme reagieren zu können.**
- **Speichern Sie vor Bearbeitung der Klausur die Lösungsdatei mit dem Titel "loesungsbogen_[Matrikelnummer].txt" an einem Ort ab, an dem Sie sie jederzeit wiederfinden. Ersetzen Sie dabei [Matrikelnummer] mit ihrer Matrikelnummer.**
- **Vergessen Sie nicht, die Lösungsdatei möglichst häufig Zwischenzuspeichern. So reduzieren Sie Stress durch technische Probleme am Ende der Abgabezeit.**
- **Notfall-Hotline: 089 / 2180 9313**

- Ich habe die Lösung eigenständig und ohne die Hilfe Dritter angefertigt.
- Ich bin der rechtmäßige Nutzer dieses Uni2Work-Accounts und gebe die Lösung nicht für jemand Drittes ab.
- Ich bin zum Zeitpunkt der Prüfung immatrikuliert und kann dies auch durch ein entsprechendes Dokument jederzeit während des Prüfungsprozesses bestätigen.
- Ich veröffentliche keinerlei Prüfungsdokumente wie Aufgabenstellung, Korrektur, etc. im öffentlichen Raum (Online, Aushang, Bekanntenkreis,...).
- Ich sorge dafür, dass ich regelmäßige Updates meiner Lösungen in Uni2Work mache, sodass es zu verminderten technischen Problemen am Ende kommt. Die letzte innerhalb der Abgabefrist hochgeladene Version wird korrigiert. Achtung: Uni2Work logged Sie nach einer gewissen Zeit der Inaktivität automatisch wieder aus!

Die Prüfung besteht aus 5 Aufgaben. Die Punktezahl ist bei jeder Aufgabe angegeben.

Aufgabe	mögliche Punkte	erreichte Punkte
1.Mathematische Grundlagen	30	
2.Daten und Algorithmen	18	
3.Grundlagen der Programmierung	36	
4.Datenstrukturen	30	
5.Polymorphismus	10	
Summe:	124	
Note:		

Bepunktung Multiple Choice: Geben Sie an, welche Antworten zu folgenden Fragen zutreffen, indem Sie die Ziffern der richtigen Antworten notieren. Alle nicht notierten Ziffern sind automatisch als falsch klassifiziert (Sie können sich also nicht enthalten). Beispiel: Angenommen, die Antworten 1,2,3 sind korrekt, 4,5,6 sind falsch, so notieren Sie in der Antwort-Datei 1,2,3 (Antworten 4,5,6 werden *NICHT* notiert und sind damit automatisch als falsch gekennzeichnet).

Pro richtiger Antwort bekommen Sie einen Punkt.

Aufgabe 1 Mathematische Grundlagen (30 Punkte)

(a) Gegeben sind die Mengen:
$A = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$
$B = \{n \in \mathbb{N} \mid n \text{ ist Primzahl}\}$
$C = \{n \in \mathbb{N} \mid n \text{ ist durch 2 teilbar}\}$
$D = \{2, 7, 4, 7, 13, 8, 27, 6\}$
Welche Aussagen sind korrekt?

1 $A \setminus B = C \cap D$

2 $\emptyset \subseteq A \cup C$

3 $A \cup (B \cap C) = A$

4 $B \cap D \subseteq A$

5 $A \cap B \cap C \cap D = \emptyset$

6 $|A \cap D| = 5$

(b) Gegeben sei die Relation $R = \{(m, n) \in \mathbb{N} \times \mathbb{N} \mid m \text{ ist Teiler von } n\}$. Welche Aussagen treffen zu?

1 $\forall n \in \mathbb{N} : (1, n) \in R$

2 $\forall (m, n) \in R : m \leq n$

3 $\forall n \in \mathbb{N} : (2, n) \in R \implies n$ ist nicht prim.

4 $\exists (m, n) \in R : m = n$

5 $\forall (m, n) \in R : (m/2, n/2) \in R$

6 $R \subseteq \{(m, n) \in \mathbb{N} \times \mathbb{N} \mid m \leq n\}\}$

(c) Gegeben sei wieder die Relation $R = \{(m, n) \in \mathbb{N} \times \mathbb{N} \mid m \text{ ist Teiler von } n\}$. Welche Aussagen treffen zu?

1 R ist reflexiv.

2 R ist symmetrisch.

3 R ist antisymmetrisch.

4 R ist transitiv.

5 R ist alternativ.

6 R ist eine Äquivalenzrelation.

(d) Gegeben sei die Funktion $ggt : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ wie in der Vorlesung. Welche Aussagen gelten?

1 ggt ist surjektiv.

2 ggt ist injektiv.

3 ggt ist total.

4 Das Urbild aller Primzahlen ist $(1, 1)$.

5 Das Bild von (9,7) ist 7.

6 Es gibt genau eine Zahl $n \in \mathbb{N}$ mit ggt(n,n) = n.

(e) Welche der folgenden Aussagen können mit vollständiger Induktion bewiesen werden?

1 $\forall x \in \mathbb{R}, x > 1 : \sqrt{x} < x$.

2 $\forall n \in \mathbb{N} : n^2 < 2^n$.

3 Die Potenzmenge einer n-elementigen Menge enthält 2^n viele Elemente.

4 $\forall n \in \mathbb{N} : 1 + 3 + \ldots + (2n - 1) = n^2$.

5 $\exists n \in \mathbb{N} : 9^n - 7$ ist teilbar durch 5.

6 $\forall \emptyset \neq M \subseteq \mathbb{N} : M$ hat ein kleinstes Element.

Aufgabe 2 Daten und Algorithmen (18 Punkte)

(a) Welche der folgenden Aussagen über Datenrepräsentationen sind korrekt?

1 Jede reelle Zahl kann in Gleitpunktdarstellung repräsentiert werden.

2 Die Menge aller positiven natürlichen Zahlen ist ein Alphabet.

3 Es gibt Ziffern (auch verschiedene), so dass gilt: $(6\_21)_8 = (1\_\_\_1011)_4$

4 Es gibt Ziffern (auch verschiedene), so dass gilt: $(\_A)_{16} = (\_\_\_\_10\_\_)_2$

5 Der Exponent der Gleitpunktdarstellung ist eine ganze Zahl.

6 Für jede rationale Zahl $q \neq 0$ ist die Gleitpunktdarstellung eindeutig.

(b) Gegeben sei folgende Sprache in BNF. Welche Sätze sind gültig?

```
<GETRAENK> ::= {<MODIFIKATOR>}* <AROMA> <TEE> <ZUSATZ>
<MODIFIKATOR> ::= heisser | leckerer | grosser | suesser
<AROMA> ::= <AROMA><AROMA> | Himbeer- | Kirsch- | Vanille-
<TEE> ::= Schwarztee | Gruentee
<ZUSATZ> ::= mit Milch | mit Rum | mit Zitrone | <Zusatz> und <Zusatz>
```

1 Himbeer-Schwarztee mit Rum

2 heisser Himbeer-Himbeer-Gruentee mit Zitrone

3 suesser grosser Vanille-Gruentee mit Rum und mit Milch

4 leckerer Kirsch-Schwarztee mit Milch und Rum und Zitrone

5 leckerer Gruentee mit Zitrone

6 grosser leckerer Kirsch-Schwarztee mit Zucker

(c) Welche Aussagen treffen auf den folgenden Algorithmus zu, der den Mittelwert eines Stacks aus ganzen Zahlen ausrechnen soll? Der Stack darf dabei zerstört werden.

```
def mittel(Stack s):
  if s == NULL OR s.peek() == NULL:
    return 0
  return help(s,1)

def help(Stack s, int n):
  if size(s) == 1:
    return s.pop()/n
  int sum = s.pop()+s.pop()
  s.push(sum)
  return help(s,n+1)
```

1 Der Algorithmus ist terminierend.

2 Der Algorithmus ist partiell korrekt.

3 Der Algorithmus ist total korrekt.

4 Der Algorthmus ist deterministisch.

5 Der Algorithmus ist determiniert.

6 Der Algorithmus ist rekursiv.

Aufgabe 3 Grundlagen der Programmierung (36 Punkte)

(a) Gegeben ist der folgende Ausdruck in Funktionsform:
$-(pow(/(+(pow(x,2),25),-(x,5)),1),5)$
Welche der Aussagen über diesen Ausdruck sind korrekt?

1 Der Ausdruck ergibt ausgerechnet x

2 Die Postfixform dieses Ausdrucks wäre: $x\ 2\ \ pow\ 25\ \ +\ x\ 5\ \ -\ \ /\ 1\ \ pow\ 5\ \ -$

3 Die Klammer $pow(x,2)$ wird als Erstes ausgewertet

4 Wenn wir den Ausdruck in Infixform bringen, sind generell keine Klammern nötig

5 Wir betrachten hier ausschließlich 2-stellige Operationen

6 Würden alle Klammern aus diesem Ausdruck gelöscht und alle Kommas durch Leerzeichen ersetzt werden, wird aus dieser Funktionsform die Präfixform

(b) Gegeben ist folgendes Code-Stück mit if-Verzweigungen:

```
1   int anzahl = 14;
2   double preis = 5.2;

3   if (anzahl > 10 || preis < 3.0){
4       if( preis < 1) {
5       System.out.println("Wow!");
6       } else {
7           System.out.println("Krass!");
8       }
9   } else if(anzahl < 10) {
10      System.out.println("Naja...");
11  }
```

Welche der Aussagen zur Ausführung dieses Code-Stücks sind korrekt?

1 Die Ausgabe ist "Wow!"

2 `anzahl` und `preis` werden im Laufe der Ausführung im Wert verändert

3 Die Ausgabe ist "Krass!"

4 Der Ausdruck `preis < 3.0` in Zeile 3 wird bei diesem Durchlauf nicht ausgewertet

5 In Zeile 4 findet bei der Auswertung ein Typecast statt

6 In Zeile 9 muss zwischen `else` und `if` eine geschweifte Klammer ({) eingefügt und nach dem folgenden if-statement am Ende von Zeile 10 wieder geschlossen (}) werden

(c) Gegeben ist folgendes Code-Stück mit switch-case:

```
1   String wort1 = "Hello";
2   String wort2 = "World";
3   String wort3 = new String("Hello");

4   if (wort1 == wort3) {
5       wort2 += " Peace";
6   }

7   switch(wort2){
8       case "Hello":
9           System.out.print("World"); break;
10      case "World":
11          System.out.print("Hello");
12      case "Hello World":
13          System.out.print(wort2 + "!"); break;
14      case "Hello World Peace":
15          System.out.print("World Peace"); break;
16      case "World Peace":
17          System.out.print("Peace!"); break;
18      default:
19          System.out.print("Candy for everyone!");
20  }
```

Welche der Aussagen zur Ausführung dieses Code-Stücks sind korrekt?

1 Die Ausgabe ist: "Peace!"

2 Da in Zeile 11 ein `break;` fehlt würde hier auch noch der folgende `case` ausgeführt werden

3 Ein äquivalenter Ausdruck in Zeile 4 wäre `if (wort1.equals(wort2))` und dieser würde auch die gleiche Ausgabe erzeugen

4 Jeder `switch-case` Block lässt sich auch durch `if-else` Ausdrücke darstellen

5 Zeile 5 führt eine String-Konkatenation durch

6 Der `default`-Fall wird immer dann ausgeführt, wenn keiner der zuvor genannten Fälle eintritt

(d) Gegeben ist das folgende Code-Stück mit einer do-while Schleife:

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

2       double ergebnis = rechnen(4, 2.5);
3       System.out.println("Das Ergebnis ist: " + ergebnis);
4   }

5   public static double rechnen(int a, double b) {
6       double ergebnis = 3D;
7       do{
8           ergebnis += b;
9           System.out.println(ergebnis);
10      } while((float) a < 15);
11      return ergebnis;
12  }
```

Welche der Aussagen zur Ausführung dieses Code-Stücks sind korrekt?

1 Die Ausgabe in Zeile 3 ist: "Das Ergebnis ist: ERROR"

2 Die Funktion `rechnen` hängt in einer Endlos-Schleife und wird nie ein Ergebnis zurückgeben

3 In Zeile 6 wird ein Fehler auftreten, weil Java nicht mit Zahlen-Buchstaben-Kombinationen umgehen kann.

4 In Zeile 10 findet eine explizite Typkonversion von `int` zu `float` statt

5 Nach Zeile 10 ist `a` jetzt vom Typ `float`

6 Bei einer `DO-WHILE` Scheife wird der Anweisungsblock immer mindestens einmal ausgeführt und erst danach wird die Bedingung geprüft

(e) Gegeben ist folgendes Code-Stück mit for-Schleifen:

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

2       int[] test = zahlenfolge(5);
3       for(int i=0; i<test.length; i++) {
4           System.out.println(test[i]);
5       }
6   }

7   public static int[] zahlenfolge(int a) {
8       int[] hilfsvar = new int[6];
9       for (int i = 0; i < a; i++) {
10          hilfsvar[i] = a;
11          i = i + 1;
12          hilfsvar[0] = hilfsvar[0]+hilfsvar[i];
13      }
14      return hilfsvar;
15  }
```

Welche der Aussagen zur Ausführung dieses Code-Stücks sind korrekt?

1 Es tritt hier **keine** ArrayIndexOutOfBoundsException auf

2 Die Schleife in Zeile 3 wird 5 Mal durchlaufen

3 Zeilen 11 und 12 lassen sich zu `hilfsvar[0] = hilfsvar[0] + hilfsvar[i++]` zusammenfassen und die Ausgabe bleibt gleich

4 Die Ausgabe ist "30 5 5 5 5 5"

5 Nachdem die `for`-Schleife beginnend in Zeile 9 das erste Mal durchlaufen wurde, hat `i` den Wert 2

6 Würde Zeile 14 fehlen, gibt die Funktion `zahlenfolge` immer 0 zurück

(f) Gegeben ist folgendes Code-Stück mit einer Ausnahmebehandlung:

```
1   Scanner sc = new Scanner(System.in);
2   String yourName;

3   do {
4       System.out.print("Please enter your name!\n");
5       yourName = sc.nextLine();
6       try {
7           if (yourName.isEmpty()) {
8               throw new NullPointerException("Error: Input was empty");
9           }
10      } catch (NullPointerException e) {
11          System.out.println(e.getMessage());
12          yourName = "some random person";
13      }
14  } while (yourName.isEmpty());

15  System.out.println("So, you are " + yourName);
```

Welche der Aussagen zur Ausführung dieses Code-Stücks sind korrekt?

1 Um in diesem Programm eine Eingabe zu machen ist eine Entwicklungsumgebung (wie zum Beispiel Eclipse, IntelliJ, etc.) zwingend erforderlich

2 Allgemein gilt: Für jeden `try`-Block darf es nur genau einen `catch`-Block geben, der die gegebenenfalls auftretende Ausnahme behandelt

3 Um dieses Code-Stück auszuführen muss über der Klassen-Definition der Scanner mit `import java.util.Scanner;` importiert werden

4 Da es sich hier um eine "Allgemeine Exception" handelt, muss diese zwingend behandelt werden

5 Sollte hier eine `NullPointerException` auftreten wird `yourName` mit dem String "some random person" gefüllt

6 Würde man nur ein Leerzeichen eingeben, läuft das Programmstück ohne Fehlermeldung durch

Aufgabe 4 Datenstrukturen (30 Punkte)

Gegeben eine Klasse `Queue` mit folgenden Methodensignaturen:

```
1  public class Queue {
2      public Queue() {...}
3      public boolean isEmpty() {...}
4      public Object get() {...}
5      public void put(Object obj) {...}
6      public Object rest() {...}
7  }
```

(a) Kreuzen Sie den Queueinhalt an, der nach folgenden Ausdrücken gespeichert ist.

```
Queue q = new Queue();
q.put(3); q.rest; q.put(7); q.put(9); q.put(2); q.rest(); q.put(3);
```

[1] 3, 7, 9, 2, 3 bzw. 3, 2, 9, 7, 3

[2] 3, 2, 9 bzw. 9, 2, 3

[3] 3, 7 bzw. 7, 3

(b) Welche Listeneigenschaften sind zur Implementierung der Klasse `Queue` aus Effizienzperspektive am besten geeignet?

[1] Einfach verankerte Liste

[2] Einfach verkettete Liste

[3] Einfach verankerte und einfach verkettete Liste

[4] Doppelt verankerte Liste

[5] Doppelt verkettete Liste

[6] Doppelt verankerte und Doppelt verkettete Liste

Gegeben eine Klasse `Stack` mit folgenden Methodensignaturen:

```
1   public class Stack {
2       public Stack() {...}
3       public boolean isEmpty() {...}
4       public Object top() {...}
5       public Object pop() {...}
6       public void push(Object obj) {...}
7   }
```

(c) Kreuzen Sie den Stackinhalt an, der nach folgenden Ausdrücken gespeichert ist.

```
Stack s = new Stack();
s.push(4); s.push(7); s.pop(); s.push(1);
s.push(2); s.pop(); s.push(6); s.push(6), s.pop();
```

☐ 1 4, 1, 6 bzw. 6, 1, 4

☐ 2 6, 2, 7 bzw. 7, 2, 6

☐ 3 6, 6, 2, 1, 7, 4 bzw. 4, 7, 1, 2, 6, 6

(d) Welche Listeneigenschaften sind zur Implementierung der Klasse `Stack` aus Effizienzperspektive am besten geeignet?

☐ 1 Einfach verankerte Liste

☐ 2 Einfach verkettete Liste

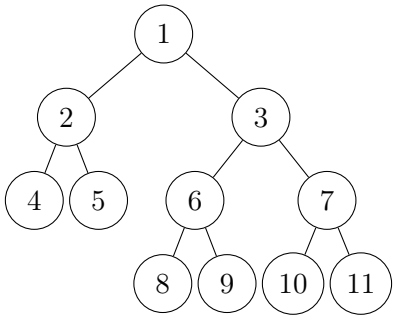
☐ 3 Einfach verankerte und einfach verkettete Liste

☐ 4 Doppelt verankerte Liste

☐ 5 Doppelt verkettete Liste

☐ 6 Doppelt verankerte und Doppelt verkettete Liste

(e) Gegeben sei der folgende Baum. Kreuzen Sie die korrekte Inorder- und Postorder-Reihenfolge des Baumes an.



☐ 1 4,2,5,1,8,6,9,3,10,7,11

☐ 2 1,2,3,4,5,6,7,8,9,10,11

☐ 3 11,10,9,8,7,6,5,4,3,2,1

☐ 4 2,3,1,4,5,8,9,6,10,11,7

☐ 5 4,5,2,8,9,6,10,11,7,3,1

☐ 6 2,1,3,4,5,6,3,7,8,9,10,11

(f) Kreuzen Sie die korrekte(n) Aussage(n) an.
Um die Gleichheit der Werte zweier Arrays `S` und `A` in Java zu überprüfen,

☐ 1 reicht es aus, die Arrays direkt miteinander zu vergleichen (`A == S`)

☐ 2 ist der Unterschied zwischen "Referenz"-Variable und "normaler" Variable zu beachten.

☐ 3 kann die Klasse `java.util.Arrays` verwendet werden.

☐ 4 reicht es nicht aus, die Arrays direkt miteinander zu vergleichen (`A == S`)

☐ 5 sollten alle Werte einzeln überprüft werden.

☐ 6 gibt es keine Möglichkeit. Zwei Arrays können nicht auf Gleichheit überprüft werden.

Aufgabe 5 Polymorphismus (10 Punkte)

Keine Panik, nur Plätzchen! Gegeben sind die unten stehenden Klassen "Gingerbread", "Biscuit" und "SeasonalBaking", in welcher Lebkuchen (Gingerbread) und (andere) Plätzchen gebacken werden. Weiterhin sind Code-Stücke und eine gewünschte Ausgabe gegeben.
Pro-Tipp: Aufgabenstellung erstmal komplett durchlesen.

```
public class Biscuit {
 public Biscuit() {
  }
  public int bakingInMin(){
    return 8;
  }
  public String taste(){
    return "yummy";
  }

  public String vs(Biscuit b){
    return taste() + ": "
                + b.bakingInMin();
  }
}
```

```
public class Gingerbread extends
Biscuit{
  public Gingerbread() {
  }
  public int bakingInMin(){
    return 20;
  }
  public String taste(){
    return "yum";
  }
  public String vs(Gingerbread z){
    return taste() + ": "
                + z.bakingInMin();
  }
}
```

```
public class SeasonalBaking {
  public static void main(String[] args) {
    Gingerbread piquant = new Gingerbread();
    Biscuit toothsome = new Biscuit();
    Biscuit delicious = new Gingerbread();

    // ??
  }
}
```

Code-Stücke zur Auswahl:

```
    0. System.out.println(delicious.taste());
    1. System.out.println(piquant.taste());
    2. System.out.println(toothsome.taste());

    3. System.out.println(delicious.bakingInMin());
    4. System.out.println(piquant.bakingInMin());
    5. System.out.println(toothsome.bakingInMin());

    6. System.out.println(delicious.vs(toothsome));
    7. System.out.println(piquant.vs(delicious));
    8. System.out.println(toothsome.vs(delicious));
    9. System.out.println(piquant.vs(toothsome));
```

Gewünschte Ausgabe:

```
yum
yum: 8
yum: 20
yum: 8
yummy
```

(a) Aus welche(r) Reihenfolge(n) der Code-Stücke an der Stelle "// ??" der SeasonalBaking Klasse resultiert die gewünschte Ausgabe?

- [0] 2,9,7,6,0
- [1] 1,6,7,8,0
- [2] 1,6,9,6,0
- [3] 1,1,9,1,4,1,4,0
- [4] 1,9,7,9,2
- [5] 1,6,7,9,2
- [6] 0,6,7,9,2
- [7] 1,6,7,6,2
- [8] 0,9,7,6,2
- [9] 2,6,9,8,0