Hochschule
Kaiserslautern
University of
Applied Sciences

Fachbereich Betriebswirtschaft
Bachelor Information Management PO 2013

Musterlösung (live, ohne Gewähr)

Probe-Klausur im Modul: Grundlagen der Informatik (GDI)

Prüfer: Prof. Dipl.-Ing. Klaus Knopper

Datum: 23.06.2015
Bearbeitungszeit: 180 Minuten

Name, Vorname: _____ **Matrikelnr.:** _____

Bewertungstabelle:

Aufgabe:	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
max.:	**10**	**5**	**10**	**25**	**15**	**15**	**10**	**10**
erreicht:								
$\sum$ erreichbar:			**$\sum$ erreicht:**			**Note:**		

Erlaubte Hilfsmittel: Bücher, Skripte, Mitschriften, Übungen, Altklausuren, nicht-programmierbarer Taschenrechner, Schreibzeug.

Hinweise zur Bearbeitung dieser Klausur: Kontrollieren Sie bitte, ob Sie alle 10 Aufgabenblätter vollständig erhalten haben. Die Aufgaben lassen sich alle direkt auf dem jeweiligen Aufgabenblatt lösen.
Viel Erfolg!

1 Begriffe der Informatik

1.1 Was versteht man unter einem *Compiler*? (2 Punkte) Worin unterscheidet er sich von einem *Interpreter*? (2 Punkte)

Antwort: Übersetzt von Quellcode in ein maschinenlesbares Format (direkt ausführbar von einer realen oder virtuellen Maschine)

Interpreter führt Quelltext "direkt" aus, ohne dass ein maschinenabhängiges Format erzeugt werden muss.

1.2 Was ist ein *Algorithmus*? (4 Punkte)

Antwort:

Ein "Kochrezept": Folge von Anweisungen, die den Weg vom Problem zum Ziel beschreiben. "Schritt-für-Schritt-Anleitung"

1.3 Was ist ein *Code*? (2 Punkte)

(Zutreffendes bitte ankreuzen. Nur eine Antwort ist richtig.)

- ◯ Ein mathematisches Rätsel
- ◯ Eine Erfindung von Leonardo DaVinci
- ☒ Eine Abbildung zwischen Binärwerten und Symbolen oder Zahlen
- ◯ Eine Programmiersprache

Summe erreichbare Punkte für Teilaufgabe 1: 10 Punkte

2 Grundlagen

2.1 (5 Punkte)

(Bitte bei jeder Antwort ein Kreuz in das richtige Feld einfügen. Richtig angekreuzte Antworten ergeben einen Punkt. Falsch angekreuzte Antworten führen zu einem Punkt Abzug. Nicht angekreuzte Antworten werden nicht bewertet. In der Gesamtwertung trägt die Aufgabe keine negativen Punkte bei.)

	Trifft zu	**Trifft nicht zu**
Jede Chomsky-Grammatik benötigt mindestens ein Nichtterminal- und ein Terminalsymbol.	X	○
Jede von einer Chomsky-Grammatik erzeugte Sprache besteht nur aus Worten, die aus einem einzigen Startsymbol ableitbar sind.	X	○
Es gibt in EBNF beschreibbare Sprachen, für die kein Syntaxdiagramm erstellt werden kann.	○	X
Der Mechanismus des Überladens betrifft nur *Attribute* einer Klasse.	○	X
Beim Überladen einer Methode kann die Anzahl der Parameter beliebig geändert werden.	X	○

Summe erreichbare Punkte für Teilaufgabe 2: 5 Punkte

3 Zahlendarstellungen

3.1 Gegeben sei die folgende Binärzahl (Dualzahl mit der üblichen Interpretation als vorzeichenlose Zahl) 1010101_2. Wie sieht die dezimale und die oktale Darstellung der Zahl aus? Geben Sie jeweils auch die Schritte zur Berechnung an. (6 Punkte)

Antwort: `1 + 0*2 + 1*4 + 0*8 + 1*16 + 0*32 + 1*64 = 85`

`85_10 = 125_8 = 5 + 8*2 + 64*1`

3.2 Wie sieht die binäre Darstellung der Zahl 41_{10} als 8 Bit Dualzahl aus? (4 Punkte)

Antwort: `1 * 32 + 0 * 16 + 1 * 8 + 0 * 4 + 0 * 2 + 1 * 1`

`-> 00101001`

Summe erreichbare Punkte für Teilaufgabe 3: 10 Punkte

4 Grammatik

4.1 Überprüfen Sie, ob die folgenden Worte mit den vorgegebenen EBNFs erzeugt werden können und markieren die entsprechenden Stellen mit Ja oder Nein. (12 Punkte)

	{ab|a}	{a}b[a]	{aaab}[aba]{aab}
aba	Ja.	Ja.	Ja.
aab	Ja.	Ja.	Ja.
aaab	Ja.	Ja.	Ja.
ε	Ja.	Nein.	Ja.

4.2 Gegeben Sei die folgende Grammatik $G = (N, T, P, S)$ mit $T = \{a, b\}$. Bestimmen Sie N und P möglichst minimal so, dass die folgende Sprache erzeugt wird:
$L(G) = \{w \mid w = ba^{2n}b^n$ mit $n \in N_0\}$. (13 Punkte)

usw.
baaaaaaaabbbb (für n=4)

N = { X }
P = { S -> bX, X -> aaXb, X-> }

5 Algorithmen: Arrays, Funktionen, Zahlendarstellung

5.1 Gegeben sei das folgende Java-Fragment. In diesem werden Dual- und Oktalzahlen als Arrays gespeichert, konvertiert und ausgegeben.

```
public static void main(String[] args) {
 int[] dual =  {1, 0, 1, 1}; // 8 + 2 + 1 = 11
 int[] oktal = {1, 0, 3, 4}; // 1*8^3 + 3*8 + 4 = 512 + 24 + 4 = 540
 System.out.println("Dualzahl: "  + toString(dual));
 System.out.println("Oktalzahl: " + toString(oktal));
 System.out.println("Dualzahl Wert: "  + toInt(dual, 2));
 System.out.println("Oktalzahl Wert: " + toInt(oktal, 8));
 int[] dual2 = octToDual(oktal);
 System.out.println("Oktalzahl Dualwert: " + toString(dual2));
}
```

5.1.1 Geben Sie die Implementierung der Methode `toString(...)` an, die ein Array in einen entsprechenden String für die Ausgaben aufbereitet. Im Beispiel oben sollte folgendes ausgegeben werden:

```
Dualzahl: 1011
Oktalzahl: 1034
```

(5 Punkte)

```
public static String toString(int[] array){
 // Das ist einfach: jedes Array-Element in String
 //                  konvertieren (automatisch)
 //                  und anhängen, von links nach rechts.
 String s = "";
 for(int i=0; i<array.length; i++) s += array[i];
 return s; // Und das Resulat zurückgeben
}
```

5.1.2 **Geben Sie die Implementierung der Methode `toInt(...)` an, die als Argument neben dem Array die zugrundeliegende Basis erhält und den Wert im Dezimalsystem zurückliefert. Im Beispiel oben sollte folgendes ausgegeben werden:**

```
Dualzahl Wert: 11
Oktalzahl Wert: 540
```

(5 Punkte)

```
public static int toInt(int[] array, int basis){
 int dez = 0; // Anfangswert 0
en,
 // allerdings ist der erste Wert im Array hier die höchste Stelle!
 for(int i=0; i<array.length; i++)
  dez += array[i] * hoch(basis,array.length-i-1);
  // oder: statt hoch() -> (int) Math.pow(basis,array.length-i-1)
 return dez;
}
```

```
// Hilfsfunktion für toInt() (s.o.), es geht natürlich auch ohne.
public static int hoch(int x, int y){ // Berechnet x^y in int
 int ergebnis = 1; // Anfangswert 1 (da Multiplikation)
 // Multipliziere ergebnis y mal mit x
 for(int i=0; i<y; i++) ergebnis *= x;
 return ergebnis;
}
```

5.1.3 **Geben Sie die Implementierung der Methode `octToDual(...)` an, die zu einer gegebenen Array-Repräsentation einer Oktalzahl eine entsprechende Dualzahl erzeugt. Im Beispiel oben sollte folgendes ausgegeben werden:**

```
Oktalzahl Dualwert: 001000011100
```

(5 Punkte)

```
 public static int[] octToDual(int[] octal){
lzahl
  // (wegen 8 = 2^3)

  // octal-Arrays entsprechen, daraus ergibt sich die Rechenvorschrift.
  int[] dual = new int[octal.length * 3]; // Platz reservieren
ibt
                                          // es 3 Stellen im dual-Array!
   dual[i*3+2] =  octal[i] % 2; // Klar, oder?
eilung durch 2
   dual[i*3+0] = (octal[i] / 2 / 2) % 2; // Nächste Dual-Stelle...
  }
  return dual; // Ergebnis zurückgeben
 }
```

6 Rekursion

6.1 Gegeben sei die folgende Klasse Test. Was wird beim Aufruf der main-Methode auf dem Bildschirm ausgegeben? (6 Punkte)

```
public class Test {
 private static String tuWas (int b, int x) {
  String erg = "";
  if (x >= b) erg += tuWas (b, x / b);
  return (erg += (x % b));
 } // tuWas

 public static void main (String [] args) {
  System.out.println(tuWas(8, 71));
  System.out.println(tuWas(3, 38));
  System.out.println(tuWas(2, 19));
 } // main
}// tuWasTest
```

Tipp: Stellen Sie eine Tabelle auf für verschiedene Eingabewerte, und setzen Sie die Ergebnisse, die sich ohne weitere Rekursion berechnen lassen, für die rekursiv aufgerufene Funktion ein.

Antwort: Es wird die Zahlendarstellung von x zur Basis b ausgegeben.

Hier: 107, 1102, 10011

6.2 Bestimmen Sie die Worst-Case-Komplexität des Rechenaufwands. Welche Ordnung ergibt sich bezüglich der Rechenkomplexität? (4 Punkte)

Antwort: "logarithmisch" O(log n) (da sich die Funktion für jede Stelle selbst aufruft, und das solange, bis die letzte Stelle erreicht ist, bei verdopplung der Eingabewerte steigt die Anzahl der Selbstaufrufe aber weniger als proportional -> log n)

6.3 Ersetzen Sie die Funktion `tuWas(...)` durch eine iterative Lösung! (5 Punkte)

```
private static String tuWasIterativ (int b, int n) {

   String s = "";
   while(x > 0){
 gen
 Basis
   }
   return s;


} // tuWasIterativ
```

Summe erreichbare Punkte für Teilaufgabe 6: 15 Punkte

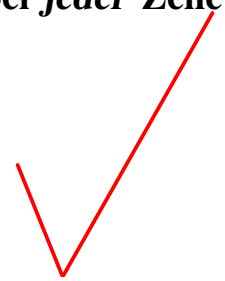
7 Programmieren und Programmverständnis

7.1 Gegeben seien die folgenden Variablenbelegungen:

```
int x = 1; int y = 4; int z = 3;
```

Welchen Wert haben die Variablen nach der Ausführung der einzelnen alternativen Zeilen? Der Ausgangswert bei *jeder* Zeile sei die oben angegebene Belegung. (5 Punkte)

	x	y	z
`x = x++;`	1	4	3
`x *= ++x;`	2	4	3
`x = y++ + --z;`	6	5	2
`y *= y += z++;`	1	28	4
`x -= (z-- - --y);`	1	3	2



7.2 Lassen sich die folgenden Codebeispiele ausführen? Falls nein, erläutern Sie kurz den Fehler. Falls ja - was wird ausgegeben? (5 Punkte)

Code	Ausgabe oder Erklärung

`double t = 8.8;`
`int i = t;`
`System.out.println(i);`
	Geht nicht, weil keine Zuweisung von double -> int erlaubt (ohne cast)

`int i = 4;`
`float f = i;`
`System.out.println(f);`
	4

`int zahl;`
`int zahl2 = zahl;`
`System.out.println(zahl2);`
	Zuweisung zahl2=zahl nicht erlaubt, da zahl nicht initialisiert ist!

`int[] a1 = {1,2,3,4,5,6};` `System.out.println(a1[0]);`	1
`int[] a2 = {1,2,3,4,5,6};` `System.out.println(a2[6]);`	Geht nicht, da das Array nur index 0..5 besitzt.

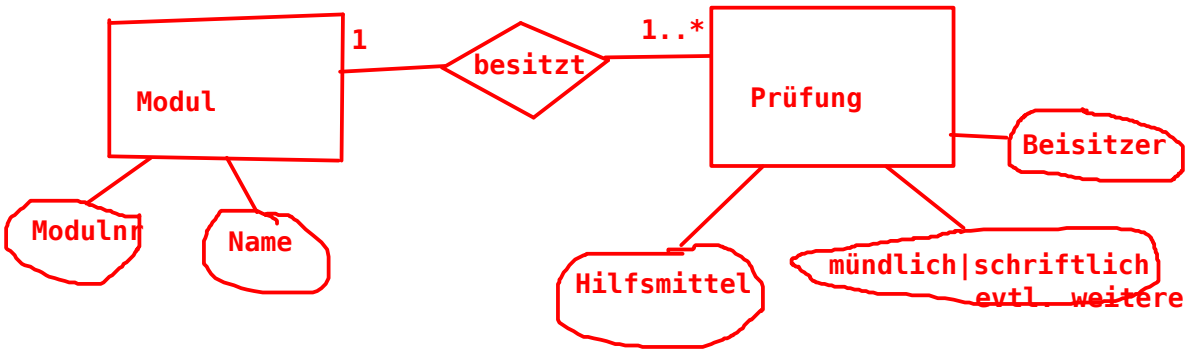
Summe erreichbare Punkte für Teilaufgabe 7: 10 Punkte

8 Objektorientierung

8.1 Ziel sei die Modellierung eines einfachen Systems zur Verwaltung von Prüfungen. Jede `Prüfung` gehört zu genau einem `Modul`. Zu jedem Modul kann es mehrere Prüfungen geben. Es muss aktuell zwischen `mündlichen` und `schriftlichen` Prüfungen unterschieden werden. Weitere Prüfungsarten können später folgen.

- Jedes Modul soll über einen Namen und eindeutige **`Modulnr`** verfügen, die von 1000 ab aufsteigend vergeben werden soll.
- Mündliche Prüfungen verfügen noch über einen **`Beisitzer`**. Dieser wird nach Erzeugung der Prüfung - erst kurz vor dem eigentlichen Termin festgelegt.
- Bei schriftlichen Prüfungen ist die Angabe von **`Hilfsmittel`**n möglich. Die zulässigen Hilfsmittel werden bei der Erzeugung einer Prüfung angegeben und dürfen danach nicht mehr verändert werden können.

8.1.1 Modellieren Sie in einem ERM-Diagramm alle für die beschriebene Anwendung notwendigen Klassen und ihre Beziehungen zueinander. (4 Punkte)



8.1.2 Definieren Sie alle vier für die Verwaltung der Module und Prüfungen notwendigen Klassen in Java. Es müssen nur die Signaturen und Implementierungen der Methoden angegeben werden, die zum Aufbau aller Daten des vollständigen Prüfungsplans erforderlich sind - also keine zur nachträglichen Bearbeitung oder Löschung. Implementieren Sie auch einen `Konstruktor` für die Klasse `Prüfung`, der beim Aufruf zwischen einer `mündlichen` und `schriftlichen` Prüfung unterscheiden kann. (6 Punkte)

```
class Modul{
 String name;
zen
t!
 public final int modulnr;          // Die individuelle Modulnummer

ummer festlegen
  this.name = name; // Namen des Moduls auf den Parameter setzen
en
 }

legen
  pruefungen = p;
 }

}

class Pruefung{

 Modul modul;  // Jede Prüfung gehört zu genau einem Modul
 enum Modus { muendlich, schriftlich }; // Später evtl. auch andere

 final String hilfsmittel;  // Wird einmalig festgelegt
tzt
 final Modus modus;               // Art der Prüfung

 public Pruefung(Modus modus, String hilfsmittel, String beisitzer){
  this.modus = modus;
  this.hilfsmittel = hilfsmittel;
  if(modus == Modus.muendlich) this.beisitzer = beisitzer;
 }

}
```

Summe erreichbare Punkte für Teilaufgabe 8: 10 Punkte

Summe erreichbare Punkte insgesamt: 100 Punkte