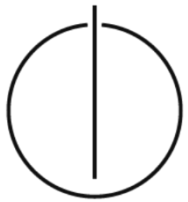


TECHNISCHE UNIVERSITÄT MÜNCHEN
FAKULTÄT FÜR INFORMATIK
Praktikum: Grundlagen der Programmierung

Programmierpraktikum

Woche 04 (17.11.2016)

Stefan Berktold
s.berktold@tum.de

Präsenzaufgaben

Heutige Übersicht

Aufgabe 4.1 ***Quiz***
Wahr/Falsch-Fragen zu Datentypen und Ausdrücken

Aufgabe 4.2 ***Programmieraufgabe (MiniJava**, erweitert)*
Vokalersetzung in Strings mit den Methoden `length()` *und* `charAt()` *(→ Datentyp* `char`*)*

Aufgabe 4.3 ***Programmieraufgabe (MiniJava**, erweitert)*
Vertauschung der Groß-/Kleinschreibung in Strings (Typecasting: `char` ↔ `int`*)*

Aufgabe 4.4 ***Quiz***
Fragen zu implizitem Typecasting für primitive Datentypen *(siehe Folie 98)*

Vorbereitung zu Hausaufgabe 4.5

Hausaufgaben

Abgabe: 21.11.2016 05:00 Uhr

Aufgabe 4.5 [7.5] ***Seiteneffekte***

Umwandlung von komplexe in mehrere (nacheinander ausgeführte) einfache Ausdrücke

Aufgabe 4.6 [6] ***Programmieraufgabe (MiniJava**, erweitert**)***

Verschlüsselung von Strings (Texten) nach Cäsar (einfacher als es sich anhört)

Aufgabe 4.7 [8+2] ***Programmieraufgabe (MiniJava**, erweitert**)***

String-Parsing: Hexadezimalzahlen aus einem String in einen Integer umwandeln

Aufgabe 4.1

Java-Quiz (1/2)

	Wahr	Falsch

Folgende Programmstücke sind semantisch äquivalent:
`int x = read(); int y = x/x;`
und
`int x = read(); int y = 1;`
	☐	☒

Der Ausdruck `0.3 == 0.1 + 0.1 + 0.1` evaluiert zu `true`.	☐	☒
Mit dem 32-bit Datentyp `float` lassen sich größere Zahlen darstellen als mit dem 64-bit Datentyp `long`.	☒	☐
Jeder 32-bit `Integer` kann in einem 32-bit `Float` dargestellt werden.	☐	☒

Aufgabe 4.1

Java-Quiz (2/2)

	Wahr	Falsch
Der Ausdruck `(x + y) - y == x` evaluiert unter der Annahme, dass `x` und `y` vom selben Zahlentyp sind, immer zu `true`.	☐	☒

Angenommen `x` und `y` sind vom Typen `int`, dann sind folgende Ausdrücke semantisch äquivalent
`x + 10 > y + 10`
und
`x > y`
	☐	☒

Der Ausdruck `2 + 5 + ">=" + 1 + 1` evaluiert zu `"7>=11"`.	☒	☐
Gegeben sei der Ausdruck `3*5+7`. Ist der Syntax-/Ableitungsbaum, der mithilfe der MiniJava-Grammatik erzeugt werden kann, eindeutig?	☐	☒

Datentyp char und String

Konkatenation (= Verknüpfung) von Strings und Charactern

```
String s;               // Variable s mit dem Typ String deklarieren
s = "Hallo";            // Variable initialisieren (Wertzuweisung)
s = s + " Welt";        // Stringkonkatenation ->  s == "Hallo Welt"

char c1 = 'A';          // Variable c1 vom Typ char mit 65 (= 'A') initialisieren
char c2 = 66;           // Variable c2 vom Typ char mit 66 (= 'B') initialisieren

s = s + c1;             // ->  s == ?   "Hallo WeltA"
s = s + c1 + c2;        // ->  s == ?   "Hallo WeltAAB"
s += c2;                // ->  s == ?   "Hallo WeltAABB"
s = s + c1 + 2;         // ->  s == ?   "Hallo WeltAABBA2"
s += (char) 65;         // ->  s == ?   "Hallo WeltAABBA2A"
s += 65;                // ->  s == ?   "Hallo WeltAABBA2A65"
s = s + (c1 + 2);       // ->  s == ?   "Hallo WeltAABBA2A6567"
s += '65';              // ->  s == ?   Fehler! char darf nur ein Zeichen sein.
```

Datentyp char und String

Rechnen mit Charactern, Funktionen length() und charAt()

```
String s = "Student";                   // Variable s initialisieren
int len = s.length();                   // Variable len mit der Länge des Strings init.
                                        // ->  len == ?    len == 7

char c0 = s.charAt(0);                  // ->  c0 == ?    c0 == 'S' == 83
char cN = s.charAt(len-1);              // ->  cN == ?    cN == 't' == 116

if (c0 >= 'A' && c0 <= 'Z') {           // äquivalent zu c0 >= 65 && c0 <= 90
       c0 += 1;                         // ->  c0 == ?    c0 == 'T' == 84
}                                       // ->  s == ?     s == "Student"

int alphabet = 'z' - 'a' + 1;           // ->  alphabet == ?    alphabet == 26

s = "Hallo";                            // ->  len == ?    len == 7
```

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Aufgabe 4.2

Vokalersetzung (Auszug)

Schreiben Sie ein Programm `Vokalersetzung`, das im unten gegebenen Text alle Vokale durch einen vom Benutzer eingegebenen Vokal ersetzt. Achten Sie darauf, dass die Groß- und Kleinschreibung nach der Ersetzung mit der vor der Ersetzung übereinstimmt. Umlaute (Ä/ä, Ö/ö, usw.) sollen dabei nicht als Vokale betrachtet werden. Beispiel: Das Wort `"Exenmeister"` wird bei Eingabe von O oder o zu `"Oxonmoostor"`.

Verwenden Sie keine Bibliotheksfunktionen der Klasse `String` außer `char charAt(int i)` und `int length()`. Erläuterungen zu diesen Funktionen finden Sie in der Angabe.

Hinweis: Eine `Integer`-Expression `e` können Sie wie folgt in einen Character umwandeln: `(char) e`. Zum Beispiel `int i = 64; char c = (char) i;` speichert in der `Character`-Variable `c` das Zeichen `'A'`. Machen Sie sich mit der ASCII-Tabelle vertraut.

Das Code-Gerüst (**`Vokalersetzung.java`**) finden Sie auf *Moodle* (in der Angabe).

Aufgabe 4.2 – Orientierung (Tipps)

Vokalersetzung

```
public class Vokalersetzung extends MiniJava {
    public static void main(String[] args) {
```

1. Vokal vom Benutzer einlesen (eingeben lassen) und als Groß- und Kleinbuchstabe abspeichern (in Variablen). Falsche Eingaben abfangen!
2. Deklariere einen leeren String für die Ausgabe (den veränderten Text).
3. Über alle Zeichen des Textes iterieren und Vokale durch den zuvor eingelesenen Vokal ersetzen (Groß-/Kleinschreibung beachten!). Eventuell geändertes Zeichen an den Ausgabestring hinten anhängen.
4. Veränderten Text ausgeben (write).

```
    }
}
```

Aufgabe 4.2 – Lösung

Vokalersetzung

Meinen Lösungsvorschlag zur ähnlich formulierten Aufgabe aus Blatt 01 des WS 15/16 gibt es unter http://tutor16.stecrz.de (PW: tutor16). Dieser Lösungsvorschlag bezieht sich auf die damalige Aufgabenstellung und dient daher nur als Orientierung!

Die offiziellen Lösungen zu allen Präsenzaufgaben sind voraussichtlich ab

18.11.2016 18:00 Uhr

auf *Moodle* verfügbar (und dürfen nicht vorab ausgehändigt werden).

Aufgabe 4.3

Rechtschreibschwäche (Auszug)

Schreiben Sie ein MiniJava-Programm, das einen `String` einliest und jeden Kleinbuchstaben durch den jeweiligen Grossbuchstaben und umgekehrt ersetzt. Alle anderen Zeichen sollen nicht ersetzt werden. Geben Sie den konvertierten `String` via der Methode `write(String s)` aus.

Beispiel: `"Hello Students!"` wird umgewandelt in `"hELLO sTUDENTS!"`.

Folgende Methoden können Sie zusätzlich verwenden:

- `int length()`
  Um die Länge eines Strings zu bestimmen rufen Sie die Methode `length()` auf.
- `char charAt(int index)`
  Um auf ein Zeichen innerhalb eines Strings zuzugreifen, können sie die Methode `charAt(i)` verwenden, welche ein `Character` an der Position `i` zurück liefert.

Verwenden Sie nur die Methoden aus dem Angabentext oder welche MiniJava bereit stellt.

Aufgabe 4.3 – Lösung

Rechtschreibschwäche (Auszug)

Die offiziellen Lösungen zu allen Präsenzaufgaben sind voraussichtlich ab

18.11.2016 18:00 Uhr

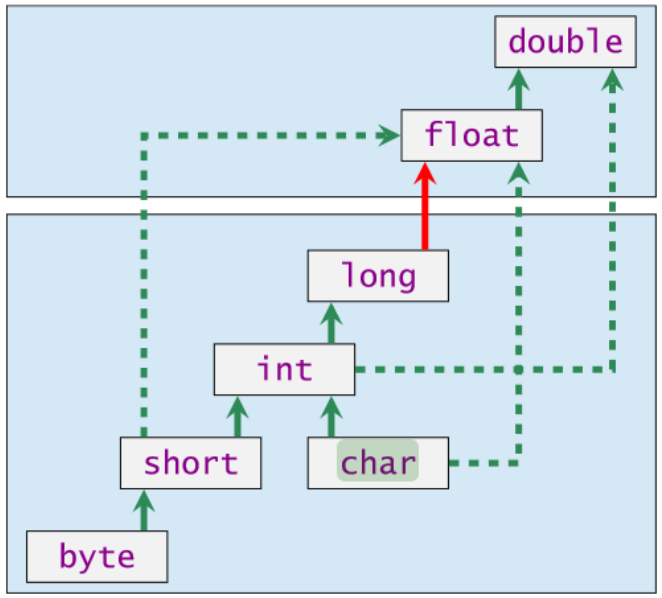
auf *Moodle* verfügbar (und dürfen nicht vorab ausgehändigt werden).

Erinnerung

Typkonvertierung

Erlaubte implizite Typecasts:

Beispiel:
```
char a = 5;
int b = a;
```



Explizite Typecasts:

Beispiel:
```
double a = 2.5;
int b = (int) a;
```

→ bspw. **nicht** möglich (*Compiler-Fehler*):
```
int c = a;
```

Aufgabe 4.4

Funktionsaufrufe (1/3)

Gegeben sind die folgenden Funktionssignaturen:

```
double max(double a, int b)
double max(int a, double b)
double max(double a, float b)
```

1. Vergleichen Sie alle Funktionen miteinander. Welche ist spezifischer als die andere? Welche sind unvergleichbar?

   Hinweis: Eine Funktion *f* ist ***spezifischer*** als eine andere Funktion *g*, wenn gilt: Für alle möglichen Parameter von *f* könnte auch *g* aufgerufen werden, nicht aber andersrum. Falls keine Funktion spezifischer ist als die andere, dann sind die Funktionen ***unvergleichbar***.

   Lösung: `double max(double a, int b)` ist spezifischer als `double max(double a, float b)`, andere Paare unvergleichbar.

Aufgabe 4.4

Funktionsaufrufe (2/3)

Gegeben sind die folgenden Funktionssignaturen:

```
double max(double a, int b)
double max(int a, double b)
double max(double a, float b)
```

2. Betrachten Sie folgenden Funktionsaufruf: `max(3, x)`

   Welche primitiven numerischen Typen kann die Variable **x** haben, sodass der Aufruf mit den oben gegebenen Funktionen gültig ist?

   Lösung: Die Variable **x** kann lediglich den Typen `double` haben, da bei allen anderen Typen immer zwei unvergleichbare Funktionen zur Wahl stehen.

Aufgabe 4.4

Funktionsaufrufe (3/3)

Gegeben sind die folgenden Funktionssignaturen:

```
double max(double a, int b)
double max(int a, double b)
double max(double a, float b)
```

3. Welche der folgenden Funktionsaufrufe sind gültig?
   Welche der Funktionen wird bei einem gültigen Aufruf verwendet?

- `max(3.0, 5.0);` – ungültiger Aufruf, da es keine passende Funktion gibt
- `max(3f, 3);` – ruft `max(double a, int b)` auf
- `max(3f, 3.0);` – ungültiger Aufruf, da es keine passende Funktion gibt
- `max(3.0, 4f);` – ruft `max(double a, float b)` auf
- `max(5.0, 2);` – ruft `max(double a, int b)` auf

Vorbereitung zu HA 4.5

Seiteneffekte

Beispiel:

```
c = --b + e++ - 2;
```

Lösung:

```
b = b - 1;   // b -= 1
c = b;
c = c + e;   // c += e
e = e + 1;   // e += 1
c = c - 2;   // c -= 2
```