

Hochschule -
Fakultät IV – Technische Informatik
Modul: Informatik
Professor: -

Portfolio

von
Sebastian Schramm Matrikel-Nr. -

3. Februar 2021

Inhaltsverzeichnis

1 Deckblatt

Deckblatt mit Ihrem vollständigen Namen, Matrikelnummer, Semester, Studiengang und Ihre unterschriebene Erklärung, dass Sie das Portfolio selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt haben. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen.

2 Persönlichkeit der Informatik

Schreiben Sie einen Aufsatz über eine Persönlichkeit der Informatik. Dieser Aufsatz sollte maximal 1500 Zeichen lang sein (ca. eine DIN A4 Seite).

Konrad Zuse und seine Errungenschaften Konrad Ernst Otto Zuse war ein deutscher Bauingenieur, Erfinder und Unternehmer der Zuse KG. Konrad Zuse wurde am 22. Juni 1910 in Deutsch-Wilmersdorf geboren und starb am 18. Dezember 1995 in Hünfeld. Zuse baute 1941 den ersten funktionstüchtigen, vollautomatischen, programmgesteuerten und frei programmierbaren, in binärer Gleitkommarechnung arbeitenden Rechner und somit den ersten funktionsfähigen Computer der Welt (Wiki, 2021). Aufgrund der monotonen und mühseligen Berechnungen im Bauingenieurwesen, wollte Zuse diese Arbeit automatisieren. Er entschloss sich 1935, seinen Beruf als Statiker zu kündigen und widmete sich der Entwicklung eines mechanischen Gehirns. 1937 wurde der erste mechanische Rechner Z1 fertiggestellt und basierte auf dem Binärsystem. Zusätzlich besaß der Z1 ein Ein-/Ausgabewerk, ein Rechenwerk, ein Speicherwerk und ein Programmwerk, das die Programme von gelochten Kinofilmstreifen ablas (Wiki, 2021). Zuse entwickelte Methoden auf der Grundlage von Mantisse und Exponenten, dies ermöglichte dem Z1 mit Gleitkommazahlen zu arbeiten (vgl. Wiki, 2021). Aufgrund der Unzuverlässigkeit des Z1, musste Zuse eine Alternative für die Schaltelemente finden. Dadurch entstand der Z2 welcher nun Relais verwendet. 1941 wurde der Z3 vollendet welcher der erste elektrisch programmierbare Computer der Welt war, welcher aber 1943 bei einem Bombenangriff zerstört wurde (vgl. Mark und Eniac, 2000).

- ¹ https://de.wikipedia.org/w/index.php?title=Konrad_Zuse&oldid=204389120
- ² <https://homecomputermuseum.de/historie/konrad-zuses-z1-z4/>

3 Wandeln Sie die folgenden Dezimalzahlen in das Hexadezimalsystem um: 2989, 57005, 48879.

- $2989 : 16 = 186 \text{ | Rest } 2989 - 16 \cdot 186 = 13$
- $186 : 16 = 11 \text{ | Rest } 186 - 16 \cdot 11 = 10$
- $11 : 16 = 0 \text{ | Rest } 11$

2989 = BAD

- $57005 : 16 = 3562 \text{ | Rest } 57005 - 16 \cdot 3562 = 13$
- $3562 : 16 = 222 \text{ | Rest } 3562 - 16 \cdot 222 = 10$
- $222 : 16 = 13 \text{ | Rest } 222 - 16 \cdot 13 = 14$
- $13 : 16 = 0 \text{ | Rest } 13$

57005 = DEAD

- $48879 : 16 = 3054 \text{ | Rest } 48879 - 16 \cdot 3054 = 15$
- $3054 : 16 = 190 \text{ | Rest } 3054 - 16 \cdot 190 = 14$
- $190 : 16 = 11 \text{ | Rest } 190 - 16 \cdot 11 = 14$
- $11 : 16 = 0 \text{ | Rest } 11$

48879 = BEEF

4 Wie zeigen Sie, die Äquivalenz der folgenden Ausdrücke mit XOR

$$(x \vee y) \wedge \neg(x \wedge y)$$

x	y	$(x \vee y)$	$\neg(x \wedge y)$	$a \wedge b$
0	0	0	1	0
0	1	1	1	1
1	0	1	1	1
1	1	1	0	0

$$\neg((x \wedge y) \vee (\neg x \wedge \neg y))$$

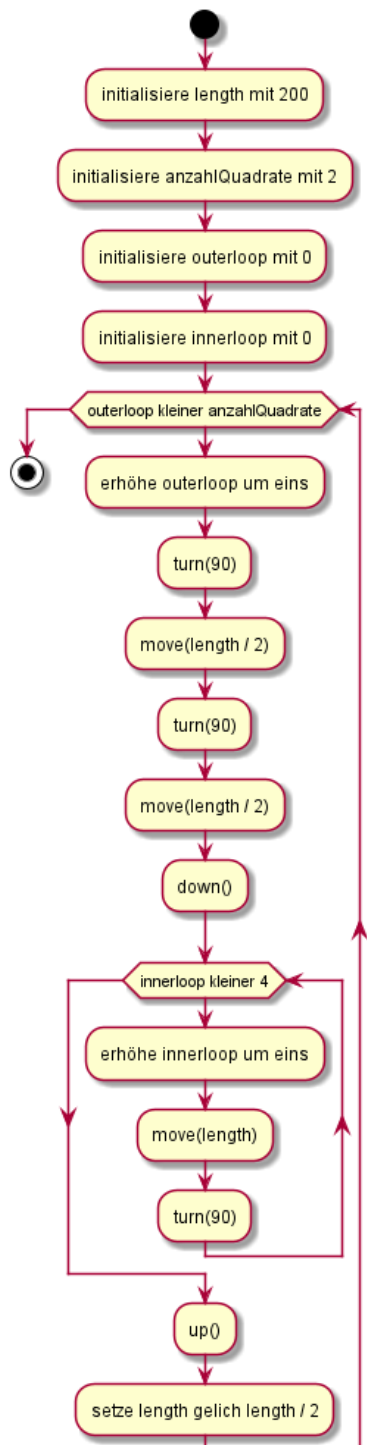
x	y	$(x \wedge y)$	$(\neg x \wedge \neg y)$	$\neg(a \vee b)$
0	0	0	1	0
0	1	0	0	1
1	0	0	0	1
1	1	1	0	0

$$\neg(x \wedge y) \wedge (x \vee y)$$

x	y	$\neg(x \wedge y)$	$(x \vee y)$	$a \wedge b$
0	0	0	1	0
0	1	1	1	1
1	0	1	1	1
1	1	1	0	0

5 zweiKonzentrischeQuadrate

Damit dieses Programm funktioniert gehen dir davon aus, dass wir oben Links starten. Also bei 0,0 und unser Startrichtung nach rechts(y) verläuft. Zusätzlich kann sich das Programm merken, in welche richtung sich der Kopf zuletzt bewegt hat.



6 IntStack

Schreiben Sie eine Java-Klasse IntStack, die einen ADT Stack implementiert. Die GrösSe (Kapazität) des Stacks soll fix sein und bei der Erzeugung eines IntStack-Objekts festgelegt werden. Intern sollen die Daten in einem int-Array abgelegt werden. Geben Sie den Source-Code ab.

6.1 Main.java

```

1 package aufgabe_08;
2

```

```

3 public class Main {
4
5     public static void main(String[] args) {
6         IntStack stack = new IntStack(32);
7         stack.print();
8         System.out.println(stack.empty());
9         stack.push(12);
10        stack.push(2);
11        for (int i = 0; i < 12; ++i)
12            stack.push((int) (Math.random()*100));
13        stack.print();
14        System.out.println(stack.empty());
15        System.out.println(stack.full());
16        stack.pop();
17        stack.pop();
18        stack.pop();
19        stack.pop();
20        stack.print();
21        for (int i = 0; i < 12; ++i)
22            stack.push((int) (Math.random()*100));
23        System.out.println(stack.empty());
24        System.out.println(stack.full());
25        stack.print();
26        for (int i = 0; i < 12; ++i)
27            stack.push((int) (Math.random()*100));
28        System.out.println(stack.empty());
29        System.out.println(stack.full());
30        stack.print();
31
32        for (int i = 0; i < 40; ++i)
33            stack.pop();
34
35        stack.push(3);
36        stack.print();
37    }
38 }

```

6.2 IntStack.java

```

1 package aufgabe_08;
2
3 public class IntStack {
4     private int[] stack;
5     private int stackPos;
6
7     public IntStack(int size) {
8         init(size);
9         stackPos = -1;
10    }
11
12    /**
13     * Legt ein neues stack array mit der gröSse size an
14     * @param size
15     */
16    private void init(int size) {
17        this.stack = new int[size];
18    }
19
20    /**
21     * Prüft ob die stackPos gleich -1 ist und gibt true zurück,
22     * andernfalls false
23     * @return

```

```

24     */
25     public boolean empty() {
26         return (stackPos == -1);
27     }
28
29     /**
30     * Prüft ob die stackPos gröSSer gleich stack.length ist und
31     * gibt true zurück, andernfalls false
32     * @return
33     */
34     public boolean full() {
35         return (stackPos >= stack.length-1);
36     }
37
38     /**
39     * Fügt eine Zahl zum stack hinzu
40     * @param number
41     */
42     public void push(int number) {
43         //Prüft ob der stack voll ist
44         if (!full())
45             stack[++stackPos] = number;
46     }
47
48     /**
49     * Entfernt eine Zahl vom stack
50     */
51     public void pop() {
52         //Prüft ob der stack leer ist
53         if (!empty())
54             stack[stackPos--] = 0;
55     }
56
57     /**
58     * Gibt die letzte Zahl aus dem stack zurück
59     * @return
60     */
61     public int top() {
62         return stack[stackPos];
63     }
64
65     /**
66     * Gibt alle Zahlen die sich im stack befinden aus
67     */
68     public void print() {
69         for (int tmp : stack)
70             System.out.print(tmp + ", ");
71         System.out.println();
72     }
73 }

```

7 Aritmetischer Ausdruck

Preoder: $*+a/bc-d*ef$

Inorder: $a+b/c*d-e*f$

Postorder: $abc/+def*-*$