Hochschule -

Fakultät IV – Technische Informatik
Modul: Informatik
Professor: -

Portfolio

von

Sebastian Schramm Matrikel-Nr. -

4. 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 Z1ein 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 Programmiergare Computer der Welt war, welcher aber 1943 bei einem Bombenangriff zerstört wurde (vgl. Mark und Eniac, 2900).

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 |Rest 2989 -16 * 486 = 13
- 186 : 16 = 11 |Rest 186 -16 * 11 = 10
- 11 : 16 = 0 |Rest 11

2989 = BAD

- 57005 : 16 = 3562|Rest 57005 -16 * 3562= 13
- 3562 : 16 = 222 |Rest 3562 -16 * 222 = 10
- 222 : 16 = 13 |Rest 222 -16 * 13 = 14
- 13 : 16 = 0 |Rest 13

57005 = DEAD

- 48879 : 16 = 3054|Rest 48879 -16 * 3054= 15
- 3054 : 16 = 190 |Rest 3054 -16 * 190 = 14
- 190 : 16 = 11 |Rest 190 -16 * 11 = 14
- 11 : 16 = 0 |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 UML

Schreiben Sie einen Aufsatz zum Thema UML. Dieser Aufsatz sollte maximal 3000 Zeichen lang sein (ca. zwei DIN A4 Seiten).

UML Währen der Blüte der Objekt orientierten Programmierung in den 1990er Jahren, kamen vermehrt Vorschläge für eine Modellierungssprache. Dazu haben sich die drei Grady Booch, Ivar Jacobson und James Rumbaugh. Sie hatten schon ihre eigenen Modellierungssprachen entworfen, nun haben sich zusammensetzten und daraus ein Konzept der UML entwickelten. Die UML oder auch Unified Modeling Language ist eine Objektorientierte Sprache und besitzt Notationen zur Beschreibung von Softwaresystemen. Der Grundgedanke hierbei, ist eine einheitliche Notation für alle Softwaresysteme. Sie dient zur Spezifikation, Konstruktion, Dokumentation und Visualisierung von Software-Teilen und anderen Systemen (UML, 2020). In der UML gibt es Notaktionselemente, aus denen sich verschiedene Diagramme erzeugen lassen. Im Jahre 1999 stieSS OMG die Entwicklung von UML 2.0 an. Dabei verdeutlicht jedes Diagramm eine bestimmte Perspektive auf das zu modellierende System und kann dem Entwickler, oder Nutzer einen guten Überblick auf die Software verschaffen. Es gibt verschiedene Modellierungssprachen, wie das Aktivitätsdiagramm welches die Ablaufmöglichkeiten eines Systems beziehungsweise Anwendungsfalls mit Aktionen darstellt. Wobei eine Aktion einen einzelnen schritt darstellt und mit Kanten wird der Kontrollfluss angegeben. Bei einem Anwendungsfalldiagramm oder auch Use Case Diagram, wird das Verhalten eines Systems aus der Benutzersicht dargestellt. Diese beschreiben aber auch die geplante Funktionalität eines Systems. AnschlieSSend wird noch ein Akteur benötigt, dieser kann zum Beispiel eine Person oder ein System sein und gibt an, was dieses System tun soll. (vgl. UML, 2021) Bei einem Klassendiagramm werden die Beziehungen zwischen den einzelnen Klassen verdeutlich sowie, welche Attribute und Methoden es in der jeweiligen klasse vorhanden sind. Zusätzlich ist noch ersichtlich um welche Datentypen es sich handelt und welche Attribute/Methoden public, private oder protected sind. Mit der Generalisierung wird angegeben in welcher Beziehung die jeweiligen Klassen zueinander stehen, dies wird mit einer durchgezogenen Linie und einem an einen der beiden enden dargestellt. Der Pfeil gibt an von welcher Klasse geerbt wird. (vgl. Klassendiagramm, 2021) Ein Zustandsdiagramm beschreibt den Lebenszyklus der Objekte einer Klasse und ist eine graphische Darstellung eines Zustandsautomaten welche auf dem Konzept der endlichen Automaten basieren. Mithilfe eines Zustandsdiagrammes kann dargestellt werden, in welchem zustand sich das betrachtete Objekt befindet und welches Verhalten diese Objekt in einer aktiven Klasse modelliert. (vgl. Endlicher_Automat, 2021)

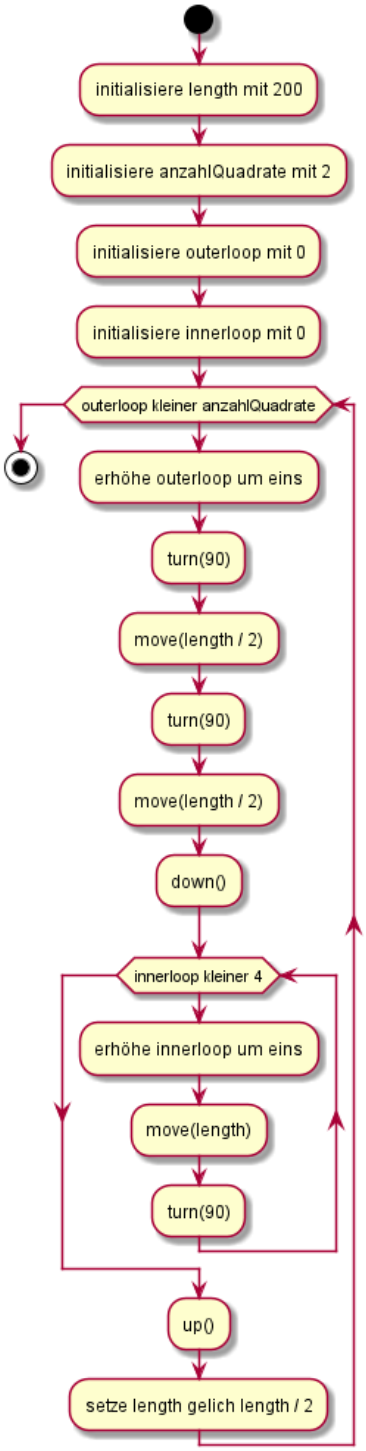
1 https://de.wikipedia.org/wiki/Unified_Modeling_Language
2 https://de.wikipedia.org/wiki/Klassendiagramm
3 https://de.wikipedia.org/wiki/Endlicher_Automat

6 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.

```
int length = 200;
int anzahlQuadrate = 2;

wiederhole (anzahlQuadrate) {
  turn(90);
  move(length / 2);
  turn(90);
  move(length / 2);
  down();
  wiederhole (4) {
    move(length);
    turn(90);
  }
  up();
  length = length / 2;
}
```



7 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.

7.1 Main.java

```
1 package aufgabe_08;
2
```

```java
public class Main {

    public static void main(String[] args) {
        IntStack stack = new IntStack(32);
        stack.print();
        System.out.println(stack.empty());
        stack.push(12);
        stack.push(2);
        for (int i = 0; i < 12; ++i)
            stack.push((int) (Math.random()*100));
        stack.print();
        System.out.println(stack.empty());
        System.out.println(stack.full());
        stack.pop();
        stack.pop();
        stack.pop();
        stack.pop();
        stack.print();
        for (int i = 0; i < 12; ++i)
            stack.push((int) (Math.random()*100));
        System.out.println(stack.empty());
        System.out.println(stack.full());
        stack.print();
        for (int i = 0; i < 12; ++i)
            stack.push((int) (Math.random()*100));
        System.out.println(stack.empty());
        System.out.println(stack.full());
        stack.print();

        for (int i = 0; i < 40; ++i)
            stack.pop();

        stack.push(3);
        stack.print();
    }
}
```

7.2 IntStack.java

```java
package aufgabe_08;

public class IntStack {
    private int[] stack;
    private int stackPos;

    public IntStack(int size) {
        init(size);
        stackPos = -1;
    }

    /**
     * Legt ein neues stack array mit der grösSe size an
     * @param size
     */
    private void init(int size) {
        this.stack = new int[size];
    }

    /**
     * Prüft ob die stackPos gleich -1 ist und gibt true zurück,
     * andernfalls false
     * @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 }
```

8 Aritmetischer Ausdruck

Preoder: *+a/bc-d*ef
Inorder: a+b/c*d-e*f
Postorder: abc/+def*-*