Abschlussprüfung Winter 2023

Fachinformatiker für Anwendungsentwicklung

Dokumentation zur betrieblichen Projektarbeit

Simple Calculator

Ein simpler Taschenrechner mit Grundfunktionen

Abgabedatum: Düsseldorf, den 16.03.2022

Prüfungsbewerber:

Philipp Pick

Gumbertstrasse 148

40229 Düsseldorf

Ausbildungsträger:

Comcave College

Immermannstrasse 65

40210 Düsseldorf

Ausbildungsbetrieb:

Philipp Pick

Gumbertstrasse 148

40229 Düsseldorf

Inhalt

Inhalt

Simple Calculator
Ein simpler Taschenrechner mit Grundfunktionen

Inhalt

Abkürzungsverzeichnis

GUI	-	Graphical User Interface
OS	-	Operating System, Betriebssystem
MVC	-	Model-View-Controller-Pattern

Einleitung

Projektumfeld

Das Projekt wird im Zuge einer Übungs-Projektarbeit im Kurs „Projektarbeit – 300426“ angefertigt. Geleitet wird der Kurs von Herrn Markus Szyska (im Weiteren Dozent genannt) bei geschätzt 20 Teilnehmern. Der Kurs soll auf die Anfertigung der für den IHK-Abschluss wichtigen Projektarbeit vorbereiten. Dazu soll in einem etwas geringeren Umfang genauso vorgegangen werden, als wäre es die echte Projektarbeit.

Kunde ist der Dozent. Die Software muss dem Kunden bis zum Mittag des 16.03.2022 vorliegen.

Projektziel

Als Übungsprojekt soll ein Taschenrechner mit den Grundfunktionen eines handelsüblichen Taschenrechners programmiert werden.

Zu den Grundfunktionen zählen die Grundrechenarten (Addition, Subtraktion, Multiplikation, Division), eine Löschfunktion als auch eine Rückschritt-Funktion, die es erlaubt, das letzte eingegebene Zeichen zu löschen.

Als Eingabe soll ein digitales Tastenfeld per GUI zur Verfügung gestellt werden. Weitere Eingabemethoden sind laut Kundenangaben nicht vorgesehen. Außerdem muss auch nicht die Punkt-Vor-Strich-Rechnung beachtet werden.

Zum Schluss soll ein ausführbares Programm stehen, das eine Schlange von Rechenoperationen korrekt berechnen und ausgeben kann (ohne Punkt-Vor-Strich).

Projektbegründung

Das Projekt entsteht im Zuge eines Übungsprojektes. Darum steht vor allem die Übung und die erhaltene Erfahrung an erster Stelle der Sinnhaftigkeit des Projektes. Jedes OS hat integrierte Taschenrechner-Apps, die ein Vielfaches an Funktionalität im Vergleich zu diesem simplen Taschenrechner bieten.

Dennoch ist es eine sinnvolle Aufgabe für angehende Anwendungsentwickler, da es nicht um reine Programmierung von Funktionen geht. Auf simple Weise werden hier vom Programmierer auch die Lösung mathematischer und logischer Probleme verlangt.

Projektschnittstellen

Aufgrund des begrenzten Umfangs des Projektes gibt es keine technischen Schnittstellen zu anderen Systemen. Nutzer der Anwendung ist am Ende der Dozent, welcher für die abschließende Bewertung der Arbeit zuständig ist. Optional ist zudem eine Präsentation des Projektes im gesamten Kurs, wobei der Zuhörerkreis wahrscheinlich nicht sehr viel größer ist, als in einem Vier-Augen-Gespräch.

Projektabgrenzung

Nicht Teil dieses Projektes ist die Implementierung komplizierterer Funktionen, wie die Punkt-Vor-Strich-Rechnung. Allerdings ist die das GUI darauf ausgelegt, diese Funktionalität zu einem späteren Zeitpunkt zu integrieren.

Projektplanung

Projektphasen

Das Projekt wird im Zeitraum vom 10.03.2022 bis einschließlich 16.03.2022 angefertigt. Das ergibt einen Zeitraum von fünf Werktagen, was bei einer täglichen Arbeitszeit von acht Stunden einen Projektumfang von 40 Stunden ergibt. Die Tabelle „Zeitplanung (kurz)“ gibt einen Eindruck davon, wie der Umfang der einzelnen Phasen für dieses Projekt veranschlagt wurden. Eine ausführliche Übersicht dieser Phasen lässt sich zudem dem Anhang „Detaillierte Zeitplanung“ entnehmen.

Zeitplanung (kurz)	
Projektphase	**Stunden**
Analyse	**7**
Entwurf	**7**
Implementierung	**13**
Qualitätskontrolle	**3**
Dokumention	**10**
Gesamtaufwand	**40**
Vorgegebener Aufwand	**40**

Ressourcenplanung

Die Übersicht „Verwendete Ressourcen“ im Anhang gibt eine genaue Auflistung aller verwendeter Ressourcen wieder. Zu diesen Ressourcen zählen sowohl Soft- und Hardware, als auch Personal. Bei der Auswahl von Software wurde nur kostenfreie genutzt. Darunter fällt viel Software aus OpenSource-Projekten, was zu einer Minimierung der Projektkosten beigetragen hat.

Entwicklungsprozess

Für die Entwicklung des Projektes musste zunächst ein passender Entwicklungsprozess gefunden werden. Aufgrund der begrenzten Zeit, des geringen Umfangs und des geringen Personals wurde auf das klassische Wasserfallmodell gesetzt. Das Wasserfallmodell ist ein lineares Modell, dass kaskadierend vorgeht. Die Phasen werden durchlaufen und erst mit Abschluss einer Phase, wird die nächste Phase eröffnet. Dies erfordert eine strukturierte Analyse und einen detaillierten Entwurf für das Projekt, da eine abgeschlossene Phase als beendet gilt. Die nachfolgenden Kapitel zeigen die einzelnen Phasen und ihre Durchführung auf.

Analysephase

Ist-Analyse

Zum Zeitpunkt der Analyse ist kein Programm vorhanden und es besteht nur der Wunsch nach einem Taschenrechner, der die Grundrechenarten beherrscht. Das Projekt dient rein zur Übung und Vorbereitung.

Wirtschaftlichkeitsanalyse

Hier soll untersucht werden, ob sich das Projekt in einem echten Umfeld rentieren würde.

Make/Buy-Entscheidung

Es gibt bereits kostenlose und vorinstallierte Taschenrechner mit weitaus größerem Funktionsumfang. Prinzipiell wäre jede investierte Einheit von Zeit oder Geld eine Verschwendung. Da es hier aber um die Gewinnung von Erfahrung geht, kann die Entscheidung nur auf „Make“ fallen. Denn wer kauft, der lernt nichts.

Projektkosten

Die imaginären Kosten des Projektes würden sich aus Personal- und Ressourcenkosten zusammensetzen. Da das Projekt in den eigenen vier Wänden mit eigener Hardware ausgeführt wird, werden Büro-/Wohnungsmiete, Hardware und Energiekosten zu einem pauschalen Stundensatz von 15€ für Ressourcennutzung zusammengefasst. Außerdem wurde bereits bei den Ressourcen darauf geachtet, dass keine neuen Lizenzen oder kostenintensive Programme verwendet werden müssen. Deshalb fallen weiter für das Projekt nur noch (fiktive) Personalkosten an. Da der Umschüler keinen Aufschluss über seine Finanzen geben möchte, wird mit einem fiktiven Stundenlohn in Höhe des gesetzlichen Mindestlohns von 12 € gerechnet. Für den Dozenten wird ein durchschnittlicher Stundenlohn in Höhe von 18,75€ angenommen. Daraus folgen Gesamtkosten von 1181,25€ für das gesamte Projekt.

Kostenaufstellung			
Vorgang	**Zeit**	**€ / h**	**Kosten**
Entwicklung	40	12€ + 15€	1.080,00 €
Abnahme	3	18,75€ + 15€	101,25 €
		Gesamt	*1.181,25 €*

Amortisationsdauer

Das Projekt bietet keine monetären Vorteile. Möglicherweise wäre eine Zeitersparnis denkbar, die sich durch die Verwendung des Taschenrechners erreichbar wäre. Diese Zeitersparnis aber Hochzurechnen ist im Zuge dieses Projektes eigentlich nicht möglich. Gehen wir einmal davon aus, dass aufgrund des geringen Umfanges der Funktionalität dennoch eine messbare Zeitersparnis möglich wäre, setzen wir diese auf 5 Minuten am Tag fest. Wir rechnen mit 200 effektiven Arbeitstagen pro Jahr.

Berechnung der Amortisation (Rundungswerte):

5 min/d x 200 d/a = 1000 min/a =

(1000/60) h/a x (18,75 € + 15 €) = 395,84 €/a

1181,25 € / 395,84 €/a = 2,98 a

Das Projekt hat sich damit in knapp 3 Jahren amortisiert. Dies ist die Zeit, die die Anwendung eingesetzt werden muss, damit Anschaffungskosten und Zeitersparnis sich ausgleichen. Dies bleibt jedoch eine fiktive Rechnung, da die kostenlosen Taschenrechner diese Zeitersparnis in gleichem Umfang erbringen und über weitaus größeren Funktionsumfang verfügen.

Nutzwertanalyse

Die Vorteile, die sich durch die gewonnene Erfahrung und die Vorbereitung auf die Projektarbeit niederschlagen, wiegen die fehlenden monetären Vorteile vollkommen auf. Durch diese Dokumentation erspart sich der Entwickler bereits viel Arbeit bei der Erstellung vieler Dokumente zur Dokumentation der eigentlichen Projektarbeit, da die meisten Dokumente als Vorlage genutzt werden können.

Anwendungsfälle

Mit dem Taschenrechner soll es möglich sein eine Kette von Grundrechenoperationen auszuführen, die nicht die Punkt-Vor-Strich-Regel beachten. Dazu muss der Nutzer Zahlen und Operatoren eingeben können. Zudem soll der Nutzer in der Lage sein, eine Kette komplett oder nur das zuletzt eingegebene Zeichen zu löschen. Das Use-Case-Diagramm, das im Anhang zu finden ist, verdeutlicht die mögliche Nutzung des Taschenrechners.

Qualitätsanforderungen

Es werden keine besonderen Qualitätsanforderungen an das Projekt gestellt. Der Entwickler hat freie Hand in der Umsetzung der Software.

Lastenheft/Fachkonzept

Zu Beginn stehen viele interessante Funktionen, die man von einem Taschenrechner erwarten würde. Durch die begrenzte Zeit des Projektes, lassen sich aber nur wenige dieser erwünschten Funktionen tatsächlich in so kurzer Zeit implementieren. Darum musste zwischen Muss und Wunsch der Anforderungen unterschieden werden.

Folgende Punkte sind als Muss zu sehen:

- Grundrechenarten: Multiplikation, Subtraktion, Addition und Division
- Eine funktionierende GUI
- Eingabe über GUI
- Berechnung einer Kette von Rechenoperationen
- Korrekturmöglichkeiten der Kette durch Löschung
- Korrekturmöglichkeiten der Kette durch Entfernung des letzten Zeichens

Folgende Punkte sind Wünsche:

- Punkt-Vor-Strich beachten

- Automatische Überprüfung der Korrektheit einer Kette
- Eingabe über Tastatur
- Nutzung der Klammerzeichen bei der Erstellung der Kette

Die Anforderungen wurden in einem Lastenheft zusammengetragen, dass in Auszügen im Anhang zu finden ist. Im Zuge des Projektes konnten nicht alle der Wünsche erfüllt werden. Nur die automatische Prüfung konnte in der verbleibenden Zeit realisiert werden. Die verpflichtenden Muss-Erwartungen wurden jedoch vollumfänglich implementiert.

Entwurfsphase

Zielplattform

Die Programmiersprache für das Projekt konnte frei gewählt werden. Da sich der Entwickler am besten auf die Programmierung mit JAVA versteht und das Projekt als Desktopanwendung umgesetzt werden sollte, konnte es in JAVA geschrieben werden. Durch die intensive Auseinandersetzung mit JAVA im Kurs konnte zudem das GUI widget toolkit Swing zur Gestaltung der Benutzeroberfläche verwendet werden.

Architekturdesign

Das Projekt folgt dem Model-View-Controller-Pattern (MVC). Dabei werden Funktionalitäten auf die drei Muster verteilt, damit das Projekt an Übersichtlichkeit gewinnt. Der Controller steuert die Anwendung. Er verbindet View und Model miteinander und versorgt sie mit den notwendigen Informationen. Das Model beinhaltet alle für die Grundfunktionen notwendigen Daten und Logiken. Es führt die Berechnungen durch und gibt die Resultate an den Controller weiter. Dieser reicht die Resultate dann an den View weiter, der die Resultate dem Nutzer anzeigt. Der View ist die einzige Teil, der dem Nutzer direkt zugänglich ist. Über den View kann der Nutzer in unserem Fall auch die Daten eingeben, die der View wiederum an den Controller weitergibt, damit er sie an das Model weiterreicht.

Da die drei Teile voneinander getrennt sind, können sie einzeln ausgetauscht und angepasst werden, ohne dass die anderen Teile betroffen wären.

Entwurf der Benutzeroberfläche

Die Eingabe von Daten soll primär über die Benutzeroberfläche stattfinden. Darum ist eine übersichtliche und leicht verständliche Oberfläche von höchster Priorität. Das Design ist zunächst zu vernachlässigen. Zunächst wurde ein Mockup der Oberfläche erstellt, welches zunächst zur Visualisierung einer möglichen Bedienoberfläche diente. Für die Bedienoberfläche mussten die zehn Ziffern, als auch die Operatoren, das Komma und auch die Sonderfunktionen leicht erreichbar sein. Dafür wurde sich an bestehenden Designs von Taschenrechnern und des Numpads der Tastatur orientiert. Außerdem musste eine Anzeige sowohl die Zeichenkette, als auch das Ergebnis für den Nutzer visualisieren. Die Klammern wurden bereits für eine spätere Implementierung eingefügt.

Das Mockup wurde im Anhang zur Ansicht hinterlegt. Für das Mockup wurde der WindowBuilder in Eclipse verwendet, da hierdurch bereits ein erster Schritt hin zum Prototypen gegangen werden konnte. Denn der

WindowBuilder generiert im Hintergrund bereits Java-Code, der zur Erstellung des Prototypen herangezogen werden kann.

Datenmodell

Für dieses Projekt wurde keine Speicherung von Daten benötigt. Daten werden nur zur Laufzeit gespeichert und bei Beendigung des Programms vergessen.

Geschäftslogik

Da die Software in keinen bestehenden Arbeitsfluss integriert werden muss, fällt es weg, darauf Rücksicht nehmen zu müssen. Stattdessen soll sich hier auf die Beschreibung der wichtigsten Bereiche der Geschäftslogik konzentriert werden. Im Anhang befindet sich ein Klassendiagramm, dass die genaue Aufteilung der Klassen versinnbildlicht.

Die Klasse App enthält die Mainmethode und dient allein dem Programmstart. Alles Weiteren wird über den Controller geregelt, der bei Erstellung seinerseits zunächst Model und View instanziiert. Ab jetzt kann er als Vermittler der beiden Klassen fungieren.

Wichtigster Bestandteil des Projektes ist die Klasse MainModel, welche alle Logiken zur Berechnung der Ketten besitzt. Hier werden zudem alle wichtigen Daten gespeichert und verwaltet. Die wichtige Zeichenkette wird hier „erinnert“ und bei Eingabe von Befehlen modifiziert. Die Kette wird hier in ihre Bestandteile zerlegt und schließlich werden die Bestandteile einer Rechenoperation unterzogen, die dann zu einem Ergebnis führt. Dieses Ergebnis landet über den Controller im View, welches ein Child besitzt, dass das gesamte UserInterface ist. Die Viewklasse selbst ist hier nur das Fenster, das den Raum für das UserInterface bietet. Das UserInterface erhält Ergebnis und Zeichenkette um sie anzuzeigen. Außerdem finden sich hier die Buttons, welche zur Befehlseingabe verwendet werden können.

Maßnahmen zur Qualitätssicherung

Zur Sicherung der Qualität sollen zunächst manuelle Tests der einzelnen Komponenten während des Schreibens des Codes durchgeführt. Hierbei kommt das White-Box-Testing zum Tragen. Zur Abnahme wird dann ein Black-Box-Testing mittels automatisierter JUnit-Tests. Dabei wird sich vor allem auf die korrekte Berechnung der Ergebnisse und die korrekte Bearbeitung der Zeichenkette durch Eingabe und Verarbeitung geprüft.

Ein weiterer wichtiger Aspekt der Qualitätssicherung ist die Nutzung von GitHub zur Versionsverwaltung. Das gesamte Projekt wird über ein offenes Repository verwaltet, wodurch Änderungen an Dateien schnell und reparabel durchgeführt werden können. Ein Beispiel für die Versionierung und die übersichtliche Dokumentation durch GitHub liegt im Anhang unter Versionierung ab.

Pflichtenheft/Datenverarbeitungskonzept

Das Pflichtenheft ist in Auszügen im Anhang zu finden. Hier sind die konkreten Anforderungen an die Umsetzung des Projektes aufgeführt.

Implementierungsphase

Implementierung der Datenstruktur

War in diesem Projekt nicht notwendig.

Implementierung der Geschäftslogik

Die Implementierung der Geschäftslogik bedeutete den größten Aufwand für das Projekt. Zur Implementierung und Realisierung der Klassen aus dem Klassendiagramm wurde die IDE Eclipse verwendet.

Die gesamte Geschäftslogik befindet sich in der Klasse MainModel. Zunächst wurde die korrekte Erstellung der Zeichenkette (equation) implementiert. Dafür wurden Methoden wie addToEquation(), deleteLastEntry() oder deleteEquation() implementiert. Sobald die Zeichenkette korrekt erstellt werden konnte, wurde die Zerlegung der Zeichenkette in die Rechenoperationen implementiert. Hierfür wurde die Funktion buildDoubleNumbers() geschrieben, welche den String in seine einzelnen Zahlen splittet. Danach wurde die Berechnung mit calculateEquation() implementiert. Einen Auszug des Codes liegt im Anhang unter Geschäftslogik ab.

Implementierung der Benutzeroberfläche

Wie bereits in der Entwurfphase beschrieben, wurde bereits das Mockup mittels WindowBuilder erstellt, welches bereits Code für eine Java Swing-Oberfläche generiert. Dieser Code wurde zur Grundlage der Benutzeroberfläche in der UserInterface-Klasse genommen. Das Design konnte so übernommen werden.

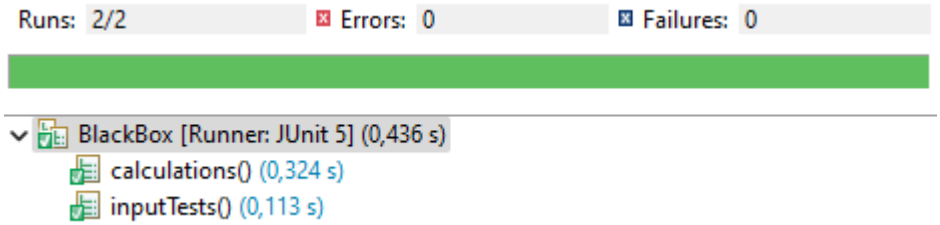
Als nächstes wurde der Code neu strukturiert und die Funktionalität der Buttons mittels ActionListeners hergestellt. So kann die Klasse mit dem Controller kommunizieren und ermöglicht Eingaben durch den User. Wie typisch für das MVC-Pattern werden die Button-Events nicht direkt in der View verarbeitet, sondern zunächst an das Model weitergereicht.

Abnahmephase

Zum Testen der Software wurden mit JUnit5 Black-Box-Tests geschrieben. Die Tests setzten sich aus den Bereichen „Calculations“ und „Input Tests“ zusammen. Im ersten Bereich wurden Testberechnungen durchgeführt. Im zweiten Bereich wurde die Eingabe der Zeichenkette auf seine Funktionalität geprüft.

```
Calculations
--- simple addition
--- simple subtraction
--- simple multiplication
--- simple division
--- mixed operators
Input Tests
--- simple input-
--- delete equation and get command
--- do not allow two symbols in row
--- remove last char from equation
--- do not allow symbol at the end when calculation is started
```

In beiden Bereichen konnte die Software die Tests bestehen. Für einen Auszug der geschriebenen JUnit-Tests reicht ein Blick in den Anhang unter Black Box Tests.



Eine offizielle Abnahme der Software hat nicht stattgefunden.

Einführungsphase

Die Einführung des Programmes verlief reibungslos. Mehr ist zu diesem Punkt hier nicht zu sagen.

Dokumentation

NICHT MEHR GESCHAFFT!

Fazit

Soll-/Ist-Vergleich

Die folgende Tabelle zeigt anschaulich, dass die Zeitplanung des Projektes Großteils eingehalten werden konnte. Lediglich eine Umverteilung von einem halben Arbeitstag konnte von der Implementierung auf die Dokumentation geschoben werden.

Zeitplanungsabgleich			
Phase	**Planung**	**Realisation**	**Abweichung**
Analyse	7	7	0
Entwurf	7	7	0
Implementierung	13	9	-4,
Abnahme	3	3	0
Dokumention	10	14	+4,
Gesamt	*40*	*40*	*0*

Gewonnene Erkenntnisse

Das Schreiben dieser Projektdokumentation hat sehr viel länger gedauert, als zunächst angenommen. Das Projekt an sich war sehr interessant und konnte hervorragend dem Zweck der Vorbereitung der anstehenden Projektarbeit dienen.

Es hat sich gezeigt, dass selbst ein vermeintlich simpler Taschenrechner sehr komplexe Strukturen enthält. Gerade das Auflösen des Strings (d.h. der Zeichenkette) zu konkreten Zahlen und den dazugehörigen Operatoren war komplexer als zunächst angenommen. Das hat mir gezeigt, dass eine gute Planung insbesondere dadurch gut wird, dass man lieber einen zu großen als zu kleinen Zeitpuffer einplanen sollte. Das mag in einem echten Projekt häufig unbeliebt sein, aber im Endeffekt hinterlässt man so einen besseren Eindruck als ein Entwickler, der Launches immer wieder nach hinten verschiebt.

Ausblick

Das Projekt kann durchaus erweitert werden, da es einige Wunschanforderungen gibt, die noch nicht implementiert werden konnten. So könnte man auch die Eingabe per Tastatur, die Punkt-Vor-Strich-Regel und Klammerzeichen implementieren.

Literaturverzeichnis

Nicht notwendig

Eidesstattliche Erklärung

Anhang

Detaillierte Zeitplanung

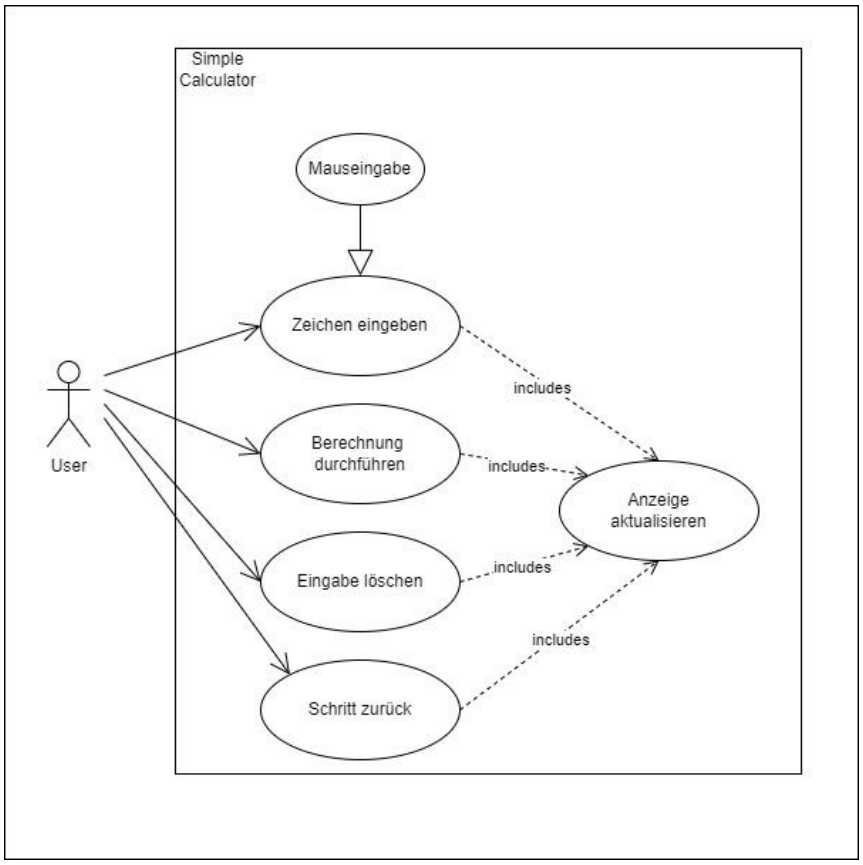
Zeitplanung (detailiert)	
Projektphase	**Stunden**
Analyse	**7**
Durchführung Ist-Analyse	2
Wirtschaftlichkeits- Amortisationsrechnung	2
Erstellung Use-Case-Diagramm	1
Erstellung Lastenheft mit Anforderungen	2
Entwurf	**7**
MockUp der Bedienoberfläche erstellen	1
Planung der Geschäftslogik	3
Pflichtenheft erstellen	3
Implementierung	**13**
Bedienoberfläche gestalten (View)	2
Bedienoberfläche Funktionalität etablieren	1
Implementierung des Models	6
Implementierung des Controllers	2
Funktionstests	2
Abnahme	**3**
Anwendertest	1
Kundenabnahme	2
Dokumention	**10**
Erstellung der Entwicklerdokumentation	1
Erstellung der Benutzerdokumentation	1
Erstellung der Projektdokumentation	8
Gesamtaufwand	**40**
Vorgegebener Aufwand	40

Verwendete Ressourcen

Simple Calculator
Ein simpler Taschenrechner mit Grundfunktionen

Anhang

Use-Case-Diagramm



Auszug aus dem Lastenheft

LASTENHEFT

3. ANFORDERUNGSBESCHREIBUNG

Folgende Anforderungen werden gestellt und genauer beschrieben:

- Grundrechenarten implementieren
- Rückschritt implementieren
- Eingabe leeren
- GUI für Eingabe und Anzeige
- Eingabe über Tastatur (Wunsch)

3.1 Grundrechenarten implementieren

3.1.1 Beschreibung

Die Applikation sollte die vier Grundrechenarten beherrschen (Addition, Subtraktion, Multiplikation und Division).

3.1.2 Wechselwirkungen

Keine bekannt. Untereinander sollte auf die Punkt-vor-Strich-Rechnung geachtet werden.

3.1.3 Risiken

Die Berechnung könnte zu falschen Ergebnissen führen, darum intensive Testung nötig.

3.1.4 Vergleich mit bestehenden Lösungen

Gleiche Funktion wie bei jedem Taschenrechner.

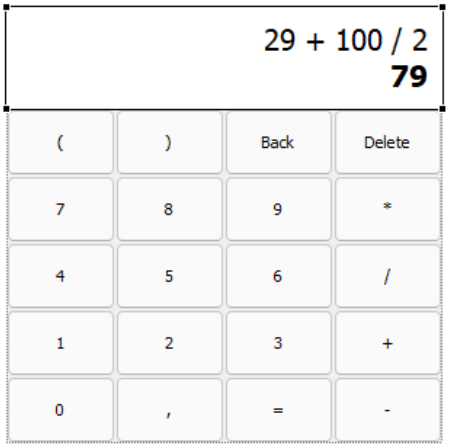
3.1.5 Schätzung des Aufwands

Kein großer Aufwand der Programmierung. Sobald die IDE steht sollte es in 1h locker zu schreiben sein.

Simple Calculator
Ein simpler Taschenrechner mit Grundfunktionen

Anhang

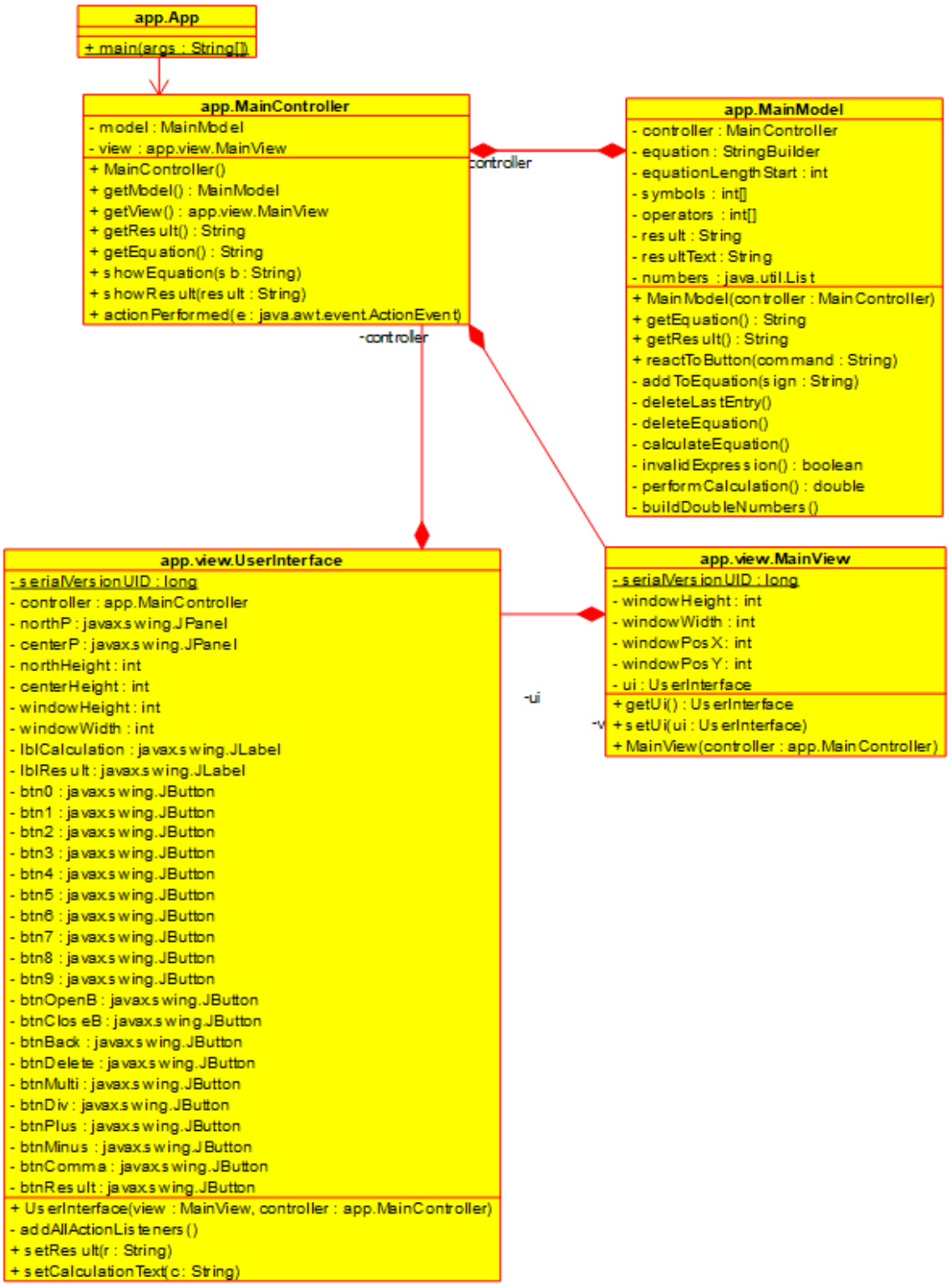
Mockup



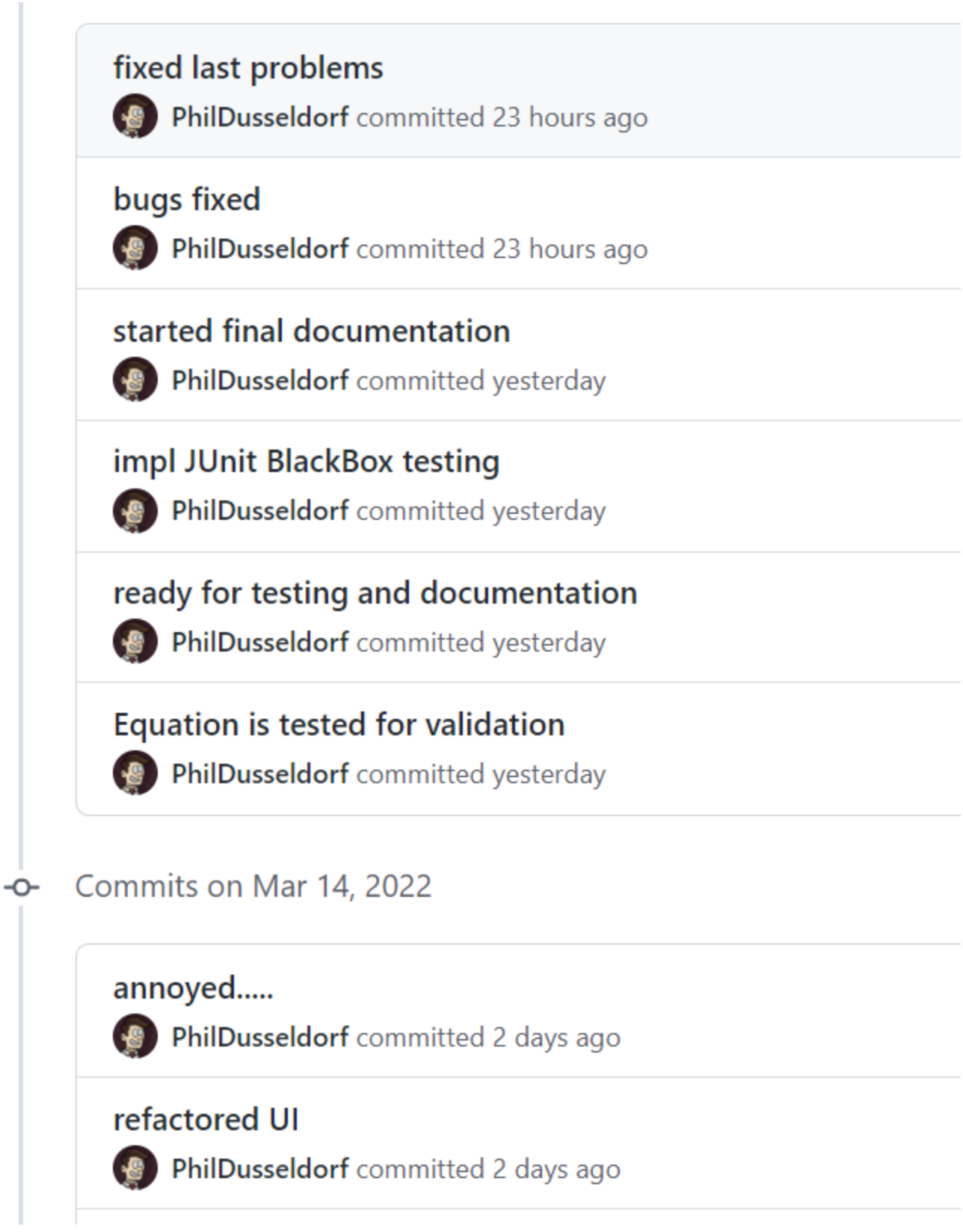
Simple Calculator
Ein simpler Taschenrechner mit Grundfunktionen

Anhang

Klassendiagramm



Versionierung



Geschäftslogik

```
85
86      private void calculateEquation() {
87          // empty lists
88          numbers.clear();
89
90          // perform calculations if expression is valid
91          if (!invalidExpression()) {
92              // get all numbers in equation
93              buildDoubleNumbers();
94              // calculate the result
95              double res = performCalculation();
96              equation.delete(0, equation.length());
97              result = String.valueOf(res);
98              controller.showResult(result);
99          } else {
100             controller.showResult("No valid expression");
101         }
102     }
103
104     private boolean invalidExpression() {
105         // ATTENION: more checks would be possible/necessary
106         // checks if last char of equation is an operator
107         String expression = equation.toString();
108         return Arrays.stream(operators).anyMatch(op -> op == (int)(expression.charAt(expression.length()-1)));
109     }
110
111     private double performCalculation() {
112         double res = numbers.get(0);
113         for (int i = 0; i < equation.length(); i++) {
114             char c = equation.charAt(i);
115             int intChar = (int)c;
116             // if an op is found a calculation is performed
117             if (Arrays.stream(operators).anyMatch(op -> op == intChar)) {
118                 switch (c) {
119                     case '+':
120                         res += numbers.get(1);
121                         numbers.remove(0);
122                         break;
123                     case '-':
124                         res -= numbers.get(1);
125                         numbers.remove(0);
126                         break;
127                     case '*':
128                         res *= numbers.get(1);
129                         numbers.remove(0);
130                         break;
131                     case '/':
132                         res /= numbers.get(1);
133                         numbers.remove(0);
134                         break;
135                 }
136             }
137         }
138         return res;
139     }
140
```

Simple Calculator
Ein simpler Taschenrechner mit Grundfunktionen

Anhang

Blackbox-Tests

```
21    @Test
22    public void inputTests() {
23        System.out.println("--- simple input-");
24        String testEquation = "1+1";
25        createInput(testEquation);
26        assertEquals(testEquation, controller.getEquation());
27
28        System.out.println("--- delete equation and get command");
29        controller.getModel().reactToButton("Delete");
30        assertEquals("type in an equation", controller.getEquation());
31
32        System.out.println("--- do not allow two symbols in row");
33        testEquation = "12+-1";
34        createInput(testEquation);
35        assertEquals("12-1", controller.getEquation());
36
37        System.out.println("--- remove last char from equation");
38        createInput("/31");
39        controller.getModel().reactToButton("Back");
40        assertEquals("12-1/3", controller.getEquation());
41
42        System.out.println("--- do not allow symbol at the end when calculation is started");
43        createInput("*");
44        controller.getModel().reactToButton("=");
45        assertEquals("Result", controller.getResult()); // "Result" is the blanco text
46    }
47
48    @Test
49    public void calculations() {
50        System.out.println("--- simple addition");
51        String testEquation = "1+1";
52        createInput(testEquation);
53        controller.getModel().reactToButton("=");
54        assertEquals("2.0", controller.getResult());
55
56        System.out.println("--- simple subtraction");
57        testEquation = "123-23";
58        createInput(testEquation);
59        controller.getModel().reactToButton("=");
60        assertEquals("100.0", controller.getResult());
61
62        System.out.println("--- simple multiplication");
63        testEquation = "11*23";
64        createInput(testEquation);
65        controller.getModel().reactToButton("=");
66        assertEquals("253.0", controller.getResult());
67
68        System.out.println("--- simple division");
69        testEquation = "21/7";
70        createInput(testEquation);
71        controller.getModel().reactToButton("=");
72        assertEquals("3.0", controller.getResult());
```