Fachinformatiker für Anwendungsentwicklung

Dokumentation zur Projektarbeit

Desktopapplikation Spiel Hangman

Prüfungsbewerber:

Martin Spille

XXX XXX

XXXXX Bremen

Inhaltsverzeichnis

1 Einleitung

1.1 Projektumfeld

Comcave College ist ein großer Bildungsträger mit Sitz in Dortmund und Niederlassungen vielen Großstädten Deutschlands.

1.2 Projektziel

Ziel des Projekts ist die Entwicklung des bekannten Buchstabenspiels Hangman als Desktopapplikation für Windows und Linux zur Unterhaltung.

2 Projektplanung

2.1 Projektphasen

Das Projekt wird im Zeitraum vom 31.08.2022 – 06.09.2022 durchgeführt. Die tägliche Arbeitszeit beläuft sich durchschnittlich auf 8 Stunden.

Tabelle 1 zeigt die grobe Zeitplanung

Projektphase	Geplante Zeit in Stunden
Planung	2
Implementierungshase	30
Kontroll- und Testphase	1
Erstellen der Dokumentation	7
Pufferzeit	0
Gesamt	40

2.2 Ressourcenplanung

Folgende Ressourcen wurden für das Projekt verwenden.
Neben dem Arbeitsplatz und dem von Comcave gestelltem Rechner, verwende ich als Technologie die Programmiersprache Java und die Entwicklungsumgebung Eclipse.

2.3 Entwicklungsprozess

Welches Modell ??
Wasserfall…, V, …

3 Analysephase
3.1 Wirtschaftlichkeitsanalyse
3.1.1 Projektkosten

Neben den anfallenden Personalkosten des Entwicklers und der Projektleitung müssen auch die Aufwendungen für die verwendeten Ressourcen, eingeplant werden.

Für den Projektleiter wird ein Stundensatz von 25,00€ und für den Auszubildenden Entwickler 10,00€ festgelegt.

Für die Ressourcennutzung werden pauschal 20€ eingeplant.

Die Durchführungszeit beträgt 40std.

Die Gesamtkosten, die während des Projekts anfallen, belaufen sich auf:

Vorgang	Zeit	Kosten pro Stunde	Kosten
Entwicklungskosten	34 h	10,00 €	340,00€
Fachgespräch	3 h	10,00€+25,00€	105,00€
Abnahmetest	1 h	10,00€+25,00€	35,00€
Code Review	2 h	10,00€+25,00€	70,00€
Ressourcennutzung	40 h	0,50€	20,00€
Gesamtkosten	40 h		570,00€

3.1.2 Amortationsdauer

Auszugehen ist von einem Verkaufspreis von 3€ und 50 Verkäufen pro Monat.

$$3{,}00€ * 50stk = 150{,}00€\, pro\ Monat\ Umsatz$$

$$\frac{570{,}00€}{150{,}00\ €} = 3{,}8\ Monate$$

Die Amortisationsdauer beträgt 3,8 Monate. Nach 4 Monaten ist davon auszugehen, dass Gewinn erzielt wird.

3.2 Anwendungsfälle

Die Anwendungsfälle wurden in einem Anwendungsfalldiagramm nach UML zusammengefasst. Das Anwendungsfalldiagramm ist im Anhang zu finden.

3.3 Lastenheft

Ein Auszug aus dem Lastenheft befindet sich im Anhang.

4 Entwurfsphase

4.1. Zielplattform

Desktop Applikation für Windows und Linux. Dementsprechend wurde die Programmierspreache Java gewählt.

4.2 Architekturdesign

Für die Architektur von der Hangmann App wurde die Anwendungsarchitektur MVC gewählt. MVC eignet sich Ideal für ein Projekt der Größe, um die GUI und die Daten mittels des Controllers zu verknüpfen.

Für die Ausrichtugn der Komponenten in der Gui eignet sich das Framework WindowsBuilder.

4.3 Geschäftslogik

Implementierte Methoden wurden im Vorfeld mittels Pseudocode skizziert, um die Fehleranfälligkeit im Code zu minimieren.

Um die Struktur zu erfassen, wurde ein UML Klassendiagramm erstellt.

Dieses befindet sich im Anhang.

4.4 Pflichtenheft

Ein Beispiel für das auf dem Pflichtenheft aufbauende Pflichtenheft befindet sich im Anhang

5 Implementierungsphase

5.1. Implementierung der Benutzeroberfläche

Die Benutzeroberfläche wurde Anhand von im Vorfeld erstellten Mockups in Java umgesetzt. Hierfür implementierte ich ein Frame, welches Zugriff auf alle Panels (Ansichten) des Spiels hat und zwischen ihnen wechselt.

5.2. Implementierung der Geschäftslogik

Auszüge aus der Klasse Model und der Methode printGuess() befinden sich im Anhang.

6 Fazit

6.1 Soll- / Ist-Vergleich

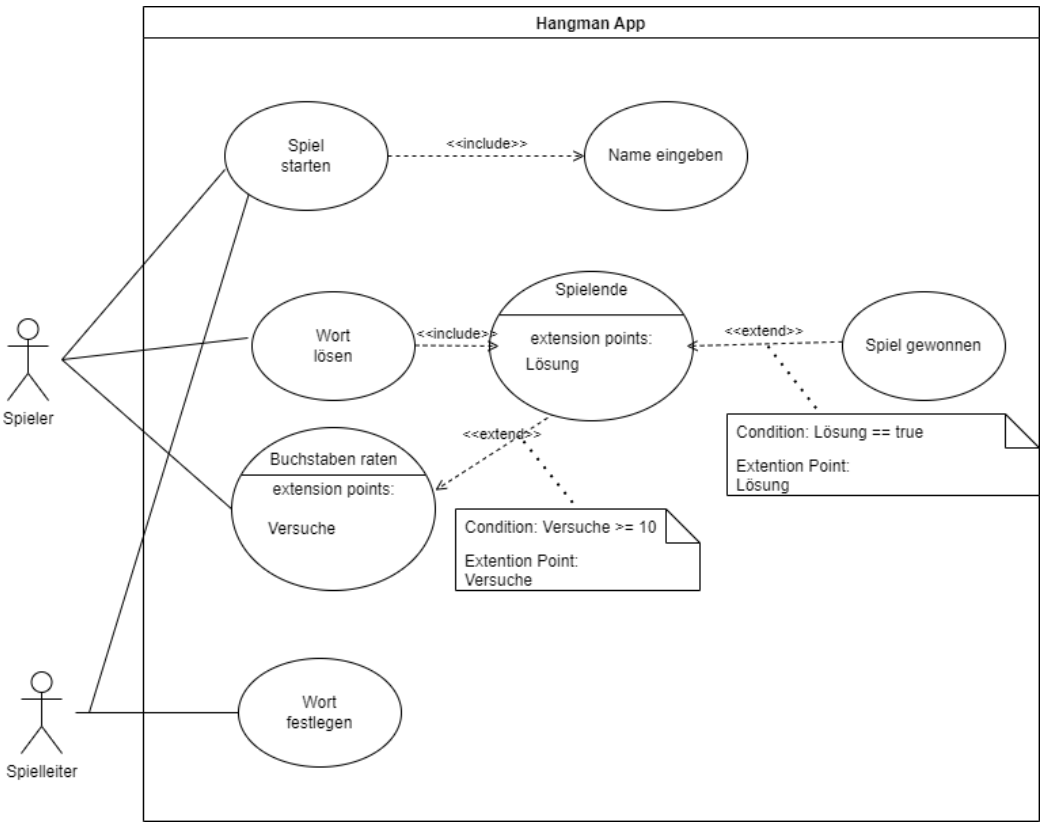
Das in der Aufgabenstellung beschriebene Projektziel wurde weitestgehend umgesetzt.

In zukünftigen Iterationen wird das Zeichnen des Hangmans nach falsch geratenen Buchstaben umgesetzt.

Das fehlerhafte Verhalten der Applikation ab einem zweiten Spiel wird behoben.

A Anhang

A1 Anwendungsfalldiagramm



A2 Lastenheft

Es folgt ein Auszug aus dem Lastenheft mit Fokus auf die Anforderungen

1.1 1. Anforderung

Nr. / ID	3.1	**Nichttechnischer Titel**	Benutzeroberfläche		
Quelle		**Verweise**		**Priorität**	

1.1.1 Beschreibung

Ein Programm mit grafischer Benutzeroberfläche in der das Spiel dargestellt wird.

1.1.2 Wechselwirkungen

Ihr Text

1.1.3 Risiken

Auflösung des Monitors könnte zu groß/zu klein sein

1.1.4 Vergleich mit bestehenden Lösungen

-

1.1.5 Schätzung des Aufwands

10h

1.2 2. Anforderung

Nr. / ID	3.2	**Nichttechnischer Titel**	Darstellung eines Wortes		
Quelle		**Verweise**		**Priorität**	

1.2.1 Beschreibung

Die Darstellung eines Wortes in einzelnen Buchstaben, die verdeckt dargestellt werden.

1.2.2 Wechselwirkungen

Ihr Text

1.2.3 Risiken

Das Wort könnte zu lang sein, d.h zu viele Buchstaben haben.

- Größe der Platzhalter skalieren nach Anzahl Buchstaben.

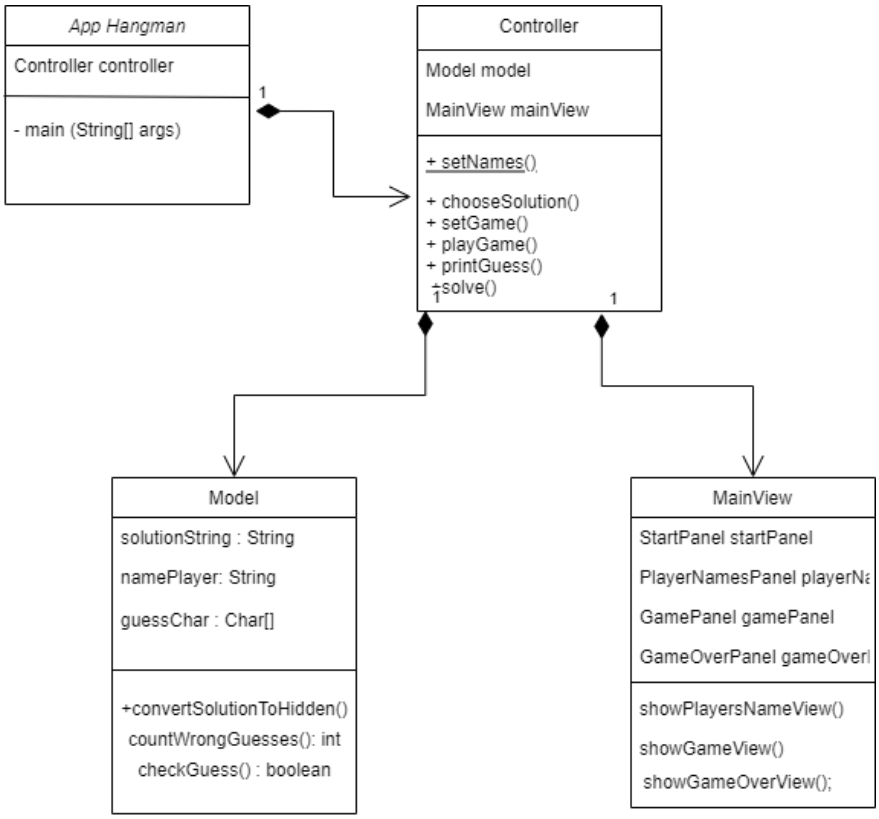
1.2.4 Vergleich mit bestehenden Lösungen

-

1.2.5 Schätzung des Aufwands

6h

A3 Klassendiagramm



A4 Pflichtenheft

fehlt

A5 Auszug aus der Klasse Model

Public class Model

```
	public String solutionString;
	public String hiddenSolution;
	public String solutionGuess;
	private String namePlayer;
	private String nameHost;
	private int wrongGuessesInt;
	public char[] revealSolutionChar;
	public char[] solutionChar;
	public char guessChar[];
	public boolean isGuessCorrect;
	private StringBuilder collectWrongLetters = new StringBuilder();

//Methoden
public void convertSolutionToHidden() {

		for(int i=0;i< this.solutionString.length();i++) {
			this.hiddenSolution += " _";
			revealSolutionChar = hiddenSolution.toCharArray();
		}
	}

	public int countWrongGuesses() {
		if(checkGuess()==false) {
			this.wrongGuessesInt++;
		}
		return wrongGuessesInt;
	}

	public boolean checkGuess() {
		isGuessCorrect = false;

		for(int i=0; i<solutionChar.length;i++) {

			if(guessChar[0] == solutionChar[i]) {
				this.revealSolutionChar[2*i+1]=solutionChar[i];
				isGuessCorrect = true;
			}
		}

		return isGuessCorrect;
	}
```

A6 Auszug aus der Methode printGuess()

```
public void printGuess() {

//richtige Buchstaben ausgeben

                if(this.getModel().checkGuess()==true) {

        this.getMainView().getGamePanel().getLabelCorrectLetter().setText(String.valueOf(th
        is.getModel().revealSolutionChar));

                }else {

//falsche Buchstaben ausgeben


        this.getModel().getCollectWrongLetters().append(this.getModel().guessChar);

        this.getMainView().getGamePanel().getLabelWrongLetter().setText(this.getModel().ge
tCollectWrongLetters().toString());


//Anzahl Versuche reduzieren


        this.getMainView().getGamePanel().getLabelCountGuesses().setText("Verbleibende
Versuche: "+(3-this.getModel().getCountWrongGuesses()));


//Prüft ob Spieler alle Versuche verbraucht hat

                        if(this.getModel().getCountWrongGuesses()==3) {

                                this.startGameOver(null);


        this.getMainView().getGameOverPanel().getLabelWhoWon().setText(this.getModel().
getNameHost()+" gewinnt.");

                        }

                }

        }
```