LA1305: Projektdokumentation

BetterThanLasVegas

|  |  |  |  |
| --- | --- | --- | --- |
| **Datum** | **Version** | **Änderung** | **Autor** |
| 31.03.23 | 0.1.0 | Erste Version | Zumstein |
| 28.04.23 | 0.2.0 | Zweite Version | Niklaus |
| 05.05.23 | 0.3.0 | Dritte Version | Göhl |
| 12.05.23 | 0.4.0 | Vierte Version | Röllin |
| 26.05.23 | 1.0.0 | Projektabschluss | Niklaus |

# Informieren

## Ihr Projekt

Wir machen ein Casino, bei welchem man eine Slotmachine und ein Zahlenratespiel spielen kann.

## Quellen

## Anforderungen

|  |  |  |  |
| --- | --- | --- | --- |
| **Nummer** | **Muss / Kann?** | **Funktional? Qualität? Rand?** | **Beschreibung** |
| 1 | Muss | Funktional | Das Programm generiert beim Random Number Game eine zufällige Zahl zwischen 1 und 100 |
| 2 | Muss | Funktional | Das Programm sollte bei den Benutzereingaben reagieren |
| 3 | Muss | Funktional | Das Programm sollte beim Random Number Game dem Benutzer Hinweise geben |
| 4 | Muss | Funktional | Wenn der Benutzer beim Random Number Game die Zahl richtig geraten hat, dann gibt es eine Benutzerausgabe |
| 5 | Muss | Funktional | Der Benutzer kann bei den Spielen Jetons gewinnen oder verlieren |
| 6 | Muss | Funktional | Der Benutzer kann nach dem Random Number Game entscheiden, ob und wie viele Runden er wieder spielen möchte |
| 7 | Muss | Funktional | Der Benutzer kann am Anfang auswählen, welches Spiel er spielen möchte |
| 8 | Muss | Funktional | Das Programm generiert bei der Slotmachine eine zufällige Reihenfolge an 3 Zahlen |
| 9 | Muss | Funktional | Wenn die drei Zahlen bei der Slotmachine gleich sind, gewinnt der Spieler |
| 10 | Muss | Funktional | Wenn die drei Zahlen bei der Slotmachine nicht gleich sind, verliert der Spieler |
| 11 | Muss | Qualität | Das Programm sollte in C# OOP und mit den Designpattern Observer und Factory Pattern eingebaut werden. |

## Testfälle

|  |  |  |
| --- | --- | --- |
| **Nummer** | **Eingabe** | **Erwartete Ausgabe** |
| 1.1 | Das Programm nimmt die von Visual Studio generierte Zahl | Die Zahl sollte zwischen 1-100 liegen und zufällig sein. |
| 2.1 | Der Benutzer gibt eine Zahl über 100 ein | Das Programm sagt dem Benutzer, dass der Benutzer eine Zahl unter 100 und über 1 eingeben sollte |
| 2.2 | Der Benutzer gibt eine Zahl unter 1 ein | Das Programm sagt dem Benutzer, dass der Benutzer eine Zahl unter 100 und über 1 eingeben sollte |
| 2.3 | Der Benutzer gibt eine Kommazahl oder ein Buchstabe ein | Das Programm sagt dem Benutzer, dass der Benutzer eine gültige Zahl eingeben sollte |
| 3.1 | Der Benutzer gibt eine Zahl höher als die generierte Zahl ein | Das Programm teilt dem Benutzer mit, dass die gesuchte Zahl tiefer ist. |
| 3.2 | Der Benutzer gibt eine Zahl kleiner als die generierte Zahl ein | Das Programm teilt dem Benutzer mit, dass die gesuchte Zahl höher ist. |
| 4.1 | Der Benutzer gibt die richtige Zahl ein | Das Programm teil dem Benutzer mit, dass die Zahl erraten wurde. |
| 5.1 | Der Benutzer verliert | Das Programm zieht dem Spieler entweder 50 Jetons (RandomNumberGame) oder die eingesetzten Jetons (Slotmachine) ab. |
| 5.2 | Der Benutzer gewinnt | Das Programm gibt dem Spieler entweder 100 Jetons (RandomNumberGame) oder die eingesetzten Jetons\*10 (Slotmachine). |
| 6.1 | Der Benutzer gibt nach/vor dem Spiel ein, dass er 4 weitere Runden spielen will. | Das Programm speichert dies und läuft 4 weitere Runden. |
| 6.2 | Nach dem Spiel entscheidet der Benutzer, dass er weiterspielen möchte | Der Code läuft weiter und das Programm fragt dem Benutzer nach, wie viele Runden er spielen möchte |
| 6.3 | Nach dem Spiel bricht der Benutzer das Programm ab | Der Code geht wieder zum Anfang und fragt dem Benutzer, welches Spiel er spielen möchtee |
| 7.1 | Der Benutzer wählt die Slotmachine/Random Number Game am Anfang des Games | Das Programm lädt die Slotmachine/Random Number Game |
| 8.1 | Der Benutzer startet die Slotmachine und gibt die Anzahl Jetons ein | Das Programm generiert bei der Slotmachine eine zufällige Reihenfolge an 3 Zahlen |
| 9.1 | Die 3 Zahlen sind gleich | Der Benutzer gewinnt und kriegt 10\* die Anzahl eingesetzte Jetons |
| 10.1 | Die 3 Zahlen sind nicht gleich | Der Benutzer verliert und ihm werden die Anzahl eingesetzte Jetons abgezogen |
| 11.1 | - | - |

1. Planen

|  |  |  |
| --- | --- | --- |
| **Nummer** | **Frist** | **Zeit (geplant)** |
| 1.1 | 28.04 | 20 min |
| 2.1 | 28.04 | 40 min |
| 2.2 | 28.04 | 40 min |
| 2.3 | 28.04 | 40 min |
| 3.1 | 28.04 | 50 min |
| 3.2 | 28.04 | 50 min |
| 4.1 | 28.04 | 50 min |
| 5.1 | 12.05 | 110 min |
| 5.2 | 12.05 | 110 min |
| 6.1 | 12.05 | 120 min |
| 6.2 | 12.05 | 50 min |
| 6.3 | 12.05 | 110 min |
| 7.1 | 05.05 | 130 min |
| 8.1 | 05.05 | 80 min |
| 9.1 | 05.05 | 80 min |
| 10.1 | 05.05 | 80 min |
| 11.1 | 26.05 | ~300 min insgesamt |
| **Total:** |  | **1460** |

1. Entscheiden

Wir haben uns entschieden, mit C# OOP zu arbeiten, damit die Klassen übersichtlicher sind und wir der Code nicht zu kompliziert wird, damit jeder von uns den Code versteht. Auch haben wir uns wegen der LB 426 entschieden, zwei Patterns (Observer und Factory Pattern) einfügen, damit wir die LB besser meistern können und die Implementation der Patterns schon geübt haben.

1. Realisieren

|  |  |  |  |
| --- | --- | --- | --- |
| Nummer | Frist | Zeit (geplant) | Zeit (effektiv) |
| 1.1 | 28.04 | 20 min | 15 min |
| 2.1 | 28.04 | 40 min | 40 min |
| 2.2 | 28.04 | 40 min | 30 min |
| 2.3 | 28.04 | 40 min | 50 min |
| 3.1 | 28.04 | 50 min | 40 min |
| 3.2 | 28.04 | 50 min | 40 min |
| 4.1 | 28.04 | 50 min | 40 min |
| 5.1 | 12.05 | 110 min | 120 min |
| 5.2 | 12.05 | 110 min | 120 min |
| 6.1 | 12.05 | 120 min | 120 min |
| 6.2 | 12.05 | 50 min | 50 min |
| 6.3 | 12.05 | 110 min | 100 min |
| 7.1 | 05.05 | 130 min | 140 min |
| 8.1 | 05.05 | 80 min | 90 min |
| 9.1 | 05.05 | 80 min | 70 min |
| 10.1 | 05.05 | 80 min | 70 min |
| 11.1 | 26.05 | ~300 min insgesamt | ~300 min insgesamt |
| Total: |  | 1460 | 1435 |

1. Kontrollieren

## Testprotokoll

|  |  |  |  |
| --- | --- | --- | --- |
| **Nummer** | **Datum** | **Resultat** | **Durchgeführt** |
| 1.1 | 26.05.23 | **OK** | Niklaus |
| 2.1 | 26.05.23 | **OK** | Niklaus |
| 2.2 | 26.05.23 | **OK** | Niklaus |
| 2.3 | 26.05.23 | **OK** | Niklaus |
| 3.1 | 26.05.23 | **OK** | Niklaus |
| 3.2 | 26.05.23 | **OK** | Niklaus |
| 4.1 | 26.05.23 | **OK** | Niklaus |
| 5.1 | 26.05.23 | **OK** | Niklaus |
| 5.2 | 26.05.23 | **OK** | Niklaus |
| 6.1 | 26.05.23 | **OK** | Niklaus |
| 6.2 | 26.05.23 | **OK** | Niklaus |
| 6.3 | 26.05.23 | **OK** | Niklaus |
| 7.1 | 26.05.23 | **OK** | Niklaus |
| 8.1 | 26.05.23 | **OK** | Niklaus |
| 9.1 | 26.05.23 | **OK** | Niklaus |
| 10.1 | 26.05.23 | **OK** | Niklaus |
| 11.1 | 26.05.23 | **OK** | Niklaus |

1. Auswerten

Wir haben dieses Projekt parallel zu einem Modul über Designpatterns gemacht, was uns sehr geholfen hat. Am Anfang hatten wir ein Problem mit GitHub, das haben wir aber dann später gefixed. Sonst lief das Projekt gut, wir hatten auch sehr viel extra Zeit zur Verfügung, um das Projekt noch abzuschliessen.