Modul Softwaretechnologie – Projektdokumentation – Einarmiger Bandit

Projekt von: Alexander Werner und Ole Hofmann

Zielsetzung:

Zu Beginn wurde als Ziel die Entwicklung eines Spielautomaten im Stil eines einarmigen Banditen definiert. Das System sollte sowohl einen Hebel zum Spielstart als auch einen Münzeinwurf mit Bank implementiert haben. Die Gewinnwahrscheinlichkeit sollte sich dynamisch an die Anzahl der in Folge verlorenen Spiele anpassen. Die drei Symbole zur Gewinnanzeige werden durch an Motoren befestigte Quadrate mit Vier verschiedenen Seitenfarben realisiert.

Softwarearchitektur:

In Abb. 1 ist ein Pseudo-Klassendiagramm der Softwarearchitektur abgebildet. Das Projekt nutzt das Design Pattern “Dependency Injection”, wobei alle Instanzen und Abhängigkeiten zentral in der ‘Main’ Klasse erstellt werden. Die Hauptlogik des Programms stellt die ‘GameInteractor’ Klasse dar. Dort wird der Spielablauf geregelt und die einzelnen Komponenten angesteuert. Das Projekt läuft in zwei Threads. Der Haupt-Thread begleitet den ‘GameInteractor’ und wartet zu Beginn blockierend auf den Starttaster. Der Sekundär-Thread kontrolliert in der ‘CoinButtonInput’ Klasse den Taster zur Geldannahme. Dieser entprellt und übermittelt die Informationen threadsicher über die ‘InMemoryGateway’ Klasse an den ‘GameInteractor’. Zusätzlich wird der ‘BrickOutputController’ angesteuert, um die Anzeige der verbleibenden Spiele zu aktualisieren. Nach der Betätigung des Starttasters prüft der ‘GameInteractor’, ob ausreichend Spiele eingezahlt wurden, dekrementiert den Spiel-Zähler und generiert abhängig der zuvor infolge verlorenen Spiele eine Farbfolge für die Motoren. Diese Farbfolge wird mittels der aktuellen Motorposition in Drehwinkel umgerechnet. Handelt es sich bei der generierten Farbfolge um einen Gewinn, also drei identische Farben, startet der ‘GameInteractor’ die Gewinn-Ausgabesequenz. Andernfalls wird eine Verlustsequenz ausgeführt. Beide Sequenzen umfassen sowohl Bild und Tonausgabe über den EV3.

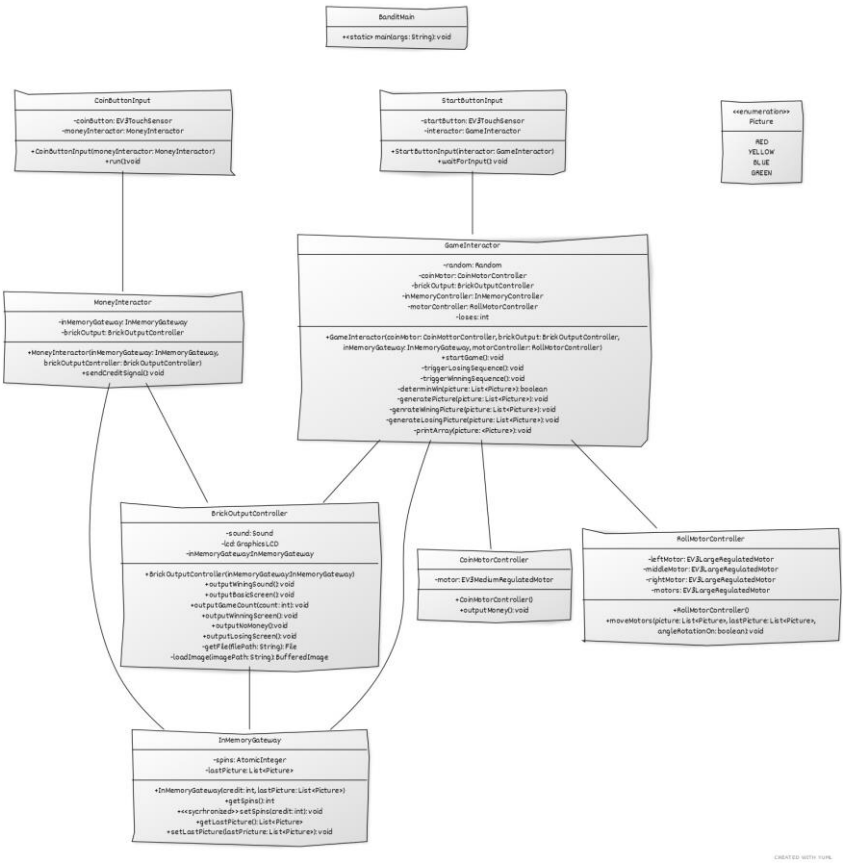

Abb. 1 Vereinfachtes Klassendiagramm | gezeichnet mit Hilfe von: https://yuml.me/

Implementierung von Theorie Themen:

Viele der theoretisch im Unterricht behandelten Themen haben Anwendung in diesem Projekt gefunden:

- Es wurde zur Dokumentation ein UML-Klassendiagramm sowie mehrere Flussdiagramme erstellt,
- Alte Code Abschnitte wurden mit Refactoring überarbeitet,
- Bei der Entwicklung wurden agile Methoden angewandt,
- Softwaretechnisch sind Designpatterns genutzt worden und Threads wurden implementiert.

Grammatik überarbeitet mit ChatGPT (https://chatgpt.com/)