# NIM-Projekt - Vorgehensweise

Einleitung

Es soll ein lauffähiges NIM-Spiel entsprechend der zuvor definierten Anforderungen realisiert werden. Die Umsetzung soll in den Kaufmännischen Schulen Tecklenburger Land des Kreises Steinfurt im Zeitraum von 09.09.2019 bis 04.11.2019 innerhalb von 40 Unterrichtsstunden erfolgen. Die Hintergrundlogik wird objektorientiert in C# mit dem Framework .NET Core 3 in Visual Studio 2019 programmiert. Die Oberfläche wird mit BLAZOR gebaut. Zur Projektverwaltung wird GitHub verwendet.

Aufgrund des geringen Umfanges des Projektes wird als Vorgehensmodell das Wasserfall-Modell gewählt. Da es sich um ein neues eigenständiges Projekt handelt, entfällt die Ist-Analyse.

Soll-Planung

Das Programm muss ein Spielfeld mit mindestens vier Reihen haben, von welchen jeweils in einem Zug mindestens ein, maximal drei, Elemente entnommen werden können. Dabei ist die Gewinnbedingung, dass der Spieler noch einen Zug ausführen kann.

Darüber hinaus, aufgrund gruppenseitiger Motivation, soll das Programm möglichst leicht erweiterbar und anpassbar sein – in der Oberfläche sowie in den Spiel-Einstellungen (Anzahl Reihen, Gewinnbedingung, etc.). Das Spiel kann von mehreren Spielern, mit oder ohne computergesteuerte Spieler, gespielt werden. Ein computergesteuerter Spieler soll im Idealfall das Spiel immer gewinnen.

Entwurf

Zunächst wird das NIM-Spiel durchstrukturiert und in Klassen unterteilt. Es gibt die zentrale Spiel-Klasse und diverse Klassen um die Spielumgebung zu definieren [siehe Anhang Abb.1]. Für die komplexeste Klasse, die Spiel-Klasse, wird zudem vor der Implementierung ein Zustandsdiagramm erstellt [siehe Anhang Abb.2].

Es wird im Internet recherchiert, welcher Algorithmus für den computergesteuerten Spieler am erfolgreichsten ist. Da verschiedene Varianten der Spieler-Logik gefunden werden, werden auch mehrere Versionen des computergesteuerten Spielers programmiert.

Implementierung

Bei der Implementierung werden zunächst die Hintergrundlogik und die Oberfläche gemäß MVVM aufgeteilt. Die eigentliche Logik ist damit soweit von der Oberfläche getrennt, dass an diesen unabhängig voneinander, parallel gearbeitet werden kann.

Hintergrundlogik

Zunächst werden die notwendigen Klassen gemäß UML-Diagrammen erstellt. Außer der Spiel-Klasse und der computergesteuerten Spieler-Klasse, ist die Realisierung der Klassen unkompliziert. Es handelt sich mehrheitlich um Klassen mit einfachen Eigenschaften und zugehörigen Zugriffsmethoden, teilweise mit einer Implementierung des IEquatable-Interfaces.

Das Spiel-Objekt übernimmt die Kernfunktionen des Programmes. Als zentrale Einheit generiert diese die notwendigen Objekte und steuert den Programmfluss sowie den Datenfluss.

Das Regel-Objekt beschreibt zulässige Züge, Spielfeldaufbau, Gewinnbedingung und Spieleranzahl. Bei Initialisierung bezieht sich die Regel-Klasse auf die Rahmenbedingungen aus der Aufgabenstellung. Anhand dessen wird das Spiel-Objekt initialisiert.

Es werden in der Hintergrundlogik alle Objekte, bis auf das Spiel-Objekt, als immutable Objekte realisiert. Alle Objekte werden demnach immer neu erstellt und ersetzt bei Veränderungen.

Oberfläche

In der Oberfläche wird nur auf die Objekte der Hintergrundlogik, die dargestellt werden müssen, zugegriffen. Mit dem Ziel der Anpassbarkeit, wird die Oberfläche in Blazor ist so strukturiert, dass diese in allen Browsern läuft und die Grafiken der dargestellten Elemente ausgetauscht werden können.

Computergesteuerter Spieler

Das Spieler-Objekt definiert eindeutig die Spieler. Ein computergesteuerter Spieler erbt von der Spieler-Klasse und überschreibt die Methode zur Zug-Entscheidung. Es wird zunächst mit der Logik „Immer den erstbesten Zug ausführen“ oder „einen zufälligen Zug ausführen“ zum initialen Testen, seitens Computer, gespielt.

Es wird ein computergesteuerter Spieler, der nach dem „XOR-Prinzip“, bei dem für den aktuellen Zustand des Spieles der beste Zug gewählt wird, in Erwägung gezogen. Dies ist der beliebteste Algorithmus für das NIM-Spiel gemäß Internetrecherche.

Letztlich arbeitet der computergesteuerte Spieler des NIM-Spiels nach dem „Min-Max-Prinzip“. Es wird immer für das gesamte Spiel ermittelt, inwiefern Züge vorteilhaft sind. Aufgrund der geringen Anzahl an möglichen Zügen in dem Spiel, ist dieser Algorithmus sehr effizient. Hier ist anpassbar, anhand eines Faktors [-1,1], ob ein entsprechend guter oder schlechter Zug auswählt wird.

Überprüfung

Nach dem erfolgreichen Verknüpfen der Programmteile, werden für alle Klassen Tests geschrieben, um deren Korrektheit zu gewährleisten. Es wird zudem mit verschiedenen Personen getestet, ob der computergesteuerte Spieler, immer, soweit möglich, gewinnt. Alle oben genannten Kriterien werden soweit erfüllt.









