Software Architektur & Design

Programmierung eines Flipperautomaten mithilfe von Entwurfsmustern.

Verwenden sie die in der Lehrveranstaltung vorgenommenen Entwurfsmuster, um einen Ablauf eines Flipperautomaten zu simulieren.
Die Angaben müssen nicht exakt so umgesetzt werden und dienen lediglich als Inspiration.

Erfinden sie ein Szenario nach ihrer Fantasie!

Zustand

Der Flipper befindet sich anfangs im *NoCredit*-**Zustand**.
Nach Einwurf einer Münze wechselt dieser in den *Ready*-Zustand und bleibt dort, bis der Start-Knopf gedrückt wird.
Der Start-Knopf versetzt den Automaten in den *Playing*-Zustand.
Zu jedem Zeitpunkt können weitere Münzen eingeworfen werden, welche den Kredit erhöhen.

Wird der Start-Knopf im *NoCredit*-Zustand gedrückt, so erscheint eine Meldung, dass kein Kredit vorhanden ist.

Wird der Start-Knopf während des *Playing*-Zustandes gedrückt, so erscheinen die Autoren der Software.

Ist eine Kugel 3-mal verloren gegangen, so wechseln sie in den *End*-State, bei welchem sie ein Spiel gewinnen können. Danach wechselt der Automat, je nach Kredit, in den *No-Credit*- bzw. *Ready-Zustand*.

Befehl und Kompositum

Erstellen sie ein paar *Flipper-Elemente* (z. B. *Rampe*, *Target*, *Bumper, SlingShot, Hole, Kicker, etc.*), welche sie beliebig oft instanziieren können.

Verwenden sie das *Command-Pattern* (***Befehl*** bzw. *Kommando*), um zu spezifizieren, was passiert, wenn ein gewisses Element getroffen wird (z. B. wenn die *hit*-Methode aufgerufen wird).
Verwenden sie auch das ***Kompositum***-*Muster* zusammen mit dem *Kommando-Muster*, um komplexere Befehle (Makro-Befehle) zu erstellen.
Zum Beispiel können sie ein Hole derart konfigurieren, sodass ein Befehl für die Punktevergabe zuständig ist und ein weiterer Befehl den Spieler bzw. die Spielerin zwischen 1, 2 und 3 wählen lässt, wobei es beim Erraten Zusatzpunkte gibt.

Adapter

Verwenden sie den objektbasierten Adapter und binden sie ein inkompatibles Flipperelement oder einen inkompatiblen Befehl (Command) in den Flipper ein.

Vermittler (Mediator)

Verwenden sie einen (oder mehrere) Vermittler, in welchem sie spezifizieren, wie die Flipper-Elemente miteinander zusammenarbeiten.

Zum Beispiel könnte sich eine Rampe öffnen, wenn alle Targets einer Gruppe getroffen wurden.
Ebenfalls könnten alle Targets einer Gruppe wieder rauf gehen, nachdem das letzte Target der Gruppe getroffen wurde.

Visitor (Besucher):

Verwenden sie das ***Visitor Pattern (Besucher)*** und implementieren sie 2 Besucher, welche eine Liste von abstrakten Flipper-Elementen durchlaufen kann um Funktionen der Konkreten Flipper-Elemente aufrufen zu können.

Ein *ResetVisitor* soll dabei alle Elemente in den Anfangszustand versetzen – Rampen werden z. B. geschlossen und die Anzahl der Durchläufe auf 0 gesetzt, Targets werden hochgefahren und die Leds ausgeschalten, etc.

Ein *PunkteVisitor* soll durch die Elemente gehen und Punkte nach einem gewissen Schema, welches sie frei erfinden dürfen, berechnen. Zum Beispiel werden die Anzahl der Durchläufe mit einem Wert multipliziert, welcher vom Level oder von sonstigen Zuständen des Flippers abhängt!

Rufen sie den Punkte-Visitor nach jedem Verlust der Kugel auf und geben sie die Punkte aus.

Abstrakte Fabrik:

Verwenden sie die ***Abstrakte Fabrik***, um die Ausgabe auf dem Flipper in unterschiedlichem Format darzustellen.
Zum Beispiel könnte beim Wechsel in den *Playing*-Zustand – je nach Kugel - folgende Ausgabe erscheinen:

```
########     ###    ##       ##           ##
##     ##   ## ##   ##       ##          ####
##     ##  ##   ##  ##       ##            ##
########  ##     ## ##       ##            ##
##     ## ######### ##       ##            ##
##     ## ##     ## ##       ##            ##
########  ##     ## ######## ########    ######
```

Folgender Link ist bei der Generierung von ASCII-Schriften hilfreich: http://patorjk.com/software/taag

Verwenden sie 2 Schriftfamilien und erstellen sie Ausgaben für das Flipper-Display. (Z. B. *Press Start, Game Over, Ball 1, Ball 2, Ball 3*, …) Die Auswahl der Schriftfamilie soll gesteuert werden können.

Singleton

Verwenden sie das **Singleton**-Muster, wo es ihnen als geeignet erscheint.

Weitere Ideen (optional):

Erstellen sie einen weiteren Mediator, mit welchem ein anderer Spielablauf bzw. ein anderes Verhalten der Elemente möglich ist.

Kombinieren sie den Mediator mit dem Besucher-Muster (Visitor).

Erstellen sie häufig verwendete Elemente mithilfe des Prototype-Musters.