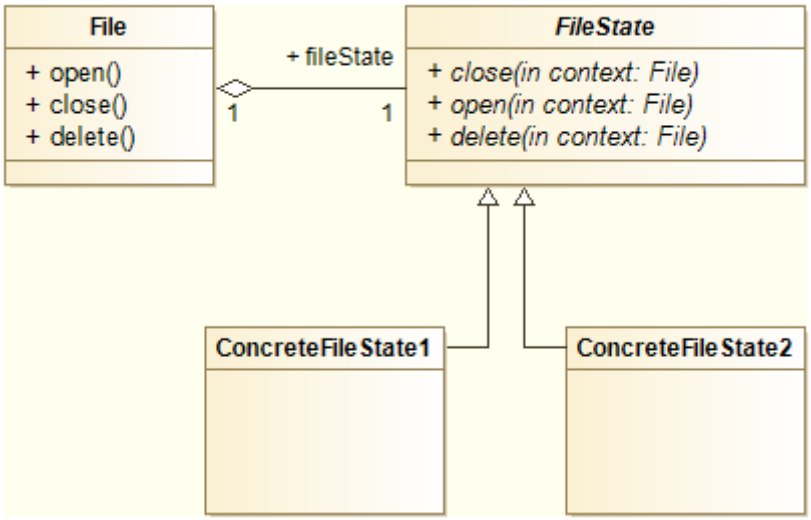
Aufgaben 3. Halbtag

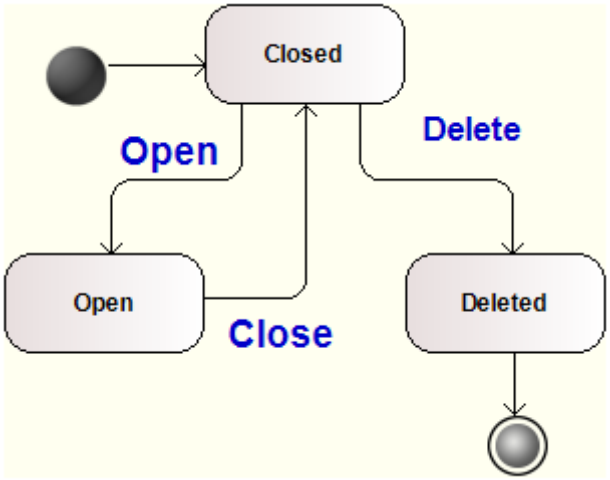
Aufgabe 1: State Pattern

Gegeben ist folgendes Klassendiagramm:



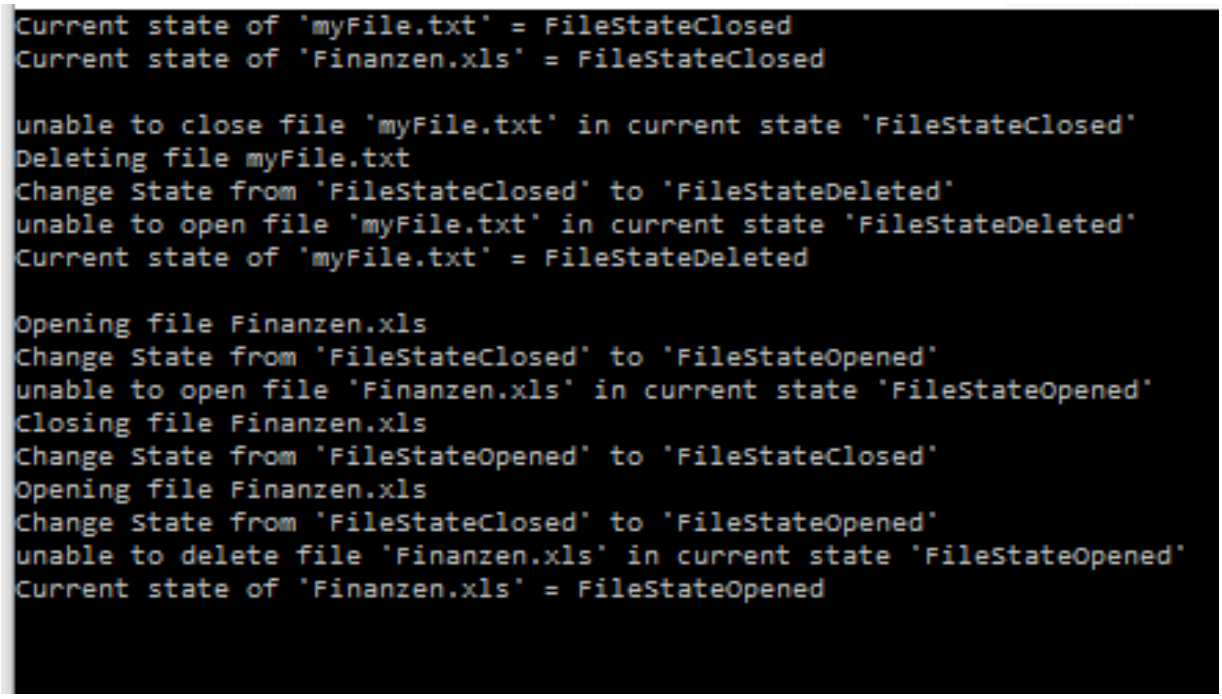
Bemerkung: Das Klassendiagramm ist allgemein gehalten bzw. die Klassen „ConcreteFileStateX" sind Platzhalter für die konkreten States gemäss Zustandsdiagramm. FileState soll als abstrakte Klasse modelliert werden.

Ein File kann in den Zuständen Open, Closed und Deleted sein, analog folgendem State Diagramm:



Modellieren Sie den Sachverhalt mit Hilfe des State Patterns.

Ergänzen Sie dazu den Code von Aufgabe 1 so, dass Ausgaben auf der Konsole wie folgt gemacht werden:



Erstellen Sie Unit Tests, welche das Verhalten verifizieren!

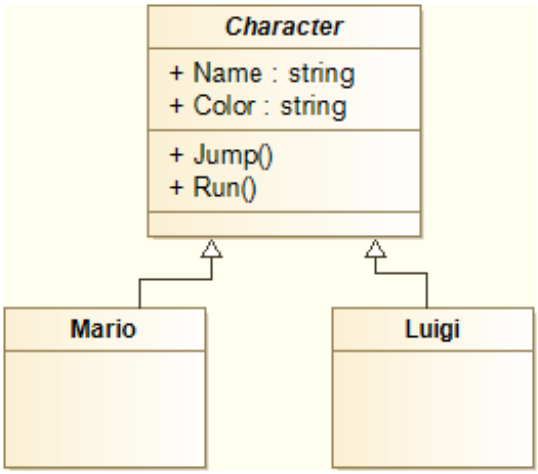
Aufgabe 2: Super Mario und Luigi

Von Nintendo wurde das legendäre Super Mario Game entwickelt. Wir müssen für eine **neue** Spielversion die Spielcharaktere etwas aufpeppen. Die bestehenden Klassen dürfen dabei **nicht** geändert werden! Erweitern Sie daher das Programm mit neuen Klassen (Main Programm wurde bereits erweitert) so, dass die neuen Charaktere verwendet werden können. Verwenden Sie für diese Problemstellung ein geeignetes Design Pattern!

```
static void Main(string[] args)
{
    Character mario = new Mario();
    Character luigi = new Luigi();

    mario.Run();
    mario.Jump();

    luigi.Run();
    luigi.Jump();

    Console.WriteLine();

    // Verhalten mit neuen Spielcharakteren

    Character marioWithYoshi = new CharacterWithYoshi(mario);
    marioWithYoshi.Run();
    marioWithYoshi.Jump();

    Character specialLuigi = new CharacterWithSpecialJump(luigi);
    specialLuigi.Run();
    specialLuigi.Jump();

    Console.ReadLine();
}
```

```
Red-Blue Mario is running
Red-Blue Mario is jumping
Green-Blue Luigi is running
Green-Blue Luigi is jumping

Yoshi!!! Red-Blue Mario is running
Yoshi and Mario is jumping
Green-Blue Luigi is running
Luigi is jumping with a back salto
```

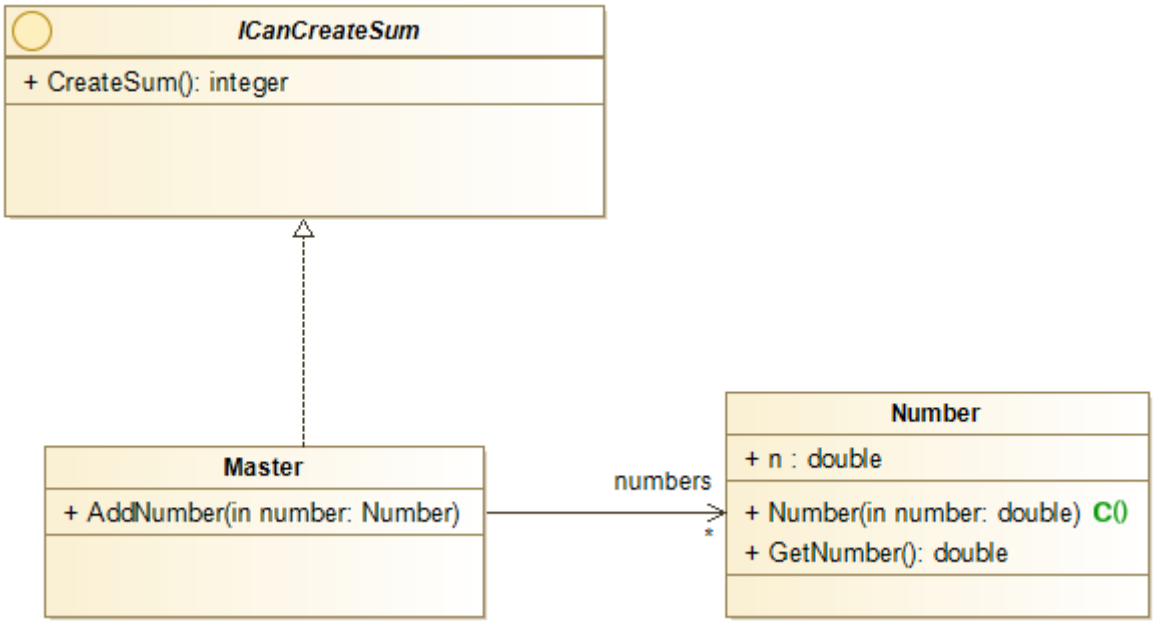
Klassendiagramm der bestehenden Klassen:



Aufgabe 3: UML to C#

Implementieren Sie folgendes UML Diagram:

Klasse «Master» implementiert das Interface «ICanCreateSum». Das Interface hat eine Methode «CreateSum», welche eine Summe berechnen soll und das Resultat als Integer zurückgibt (Gemäss Interface Beschreibung sollen allfällige Kommastellen aufgerundet werden!).
Die Klasse Master soll dies implementieren in dem sie über alle Elemente «numbers» vom Typ «Number» iteriert und die Summe bildet.



Erstellen Sie Unit Tests, welche das Verhalten verifizieren!

Bewertung (Aufgabe 1, 3):
- Funktionalität
- Unit Tests vorhanden wo anwendbar
- Coding Style, Lesbarkeit
- Einfachheit der Lösung

Abgabe:

Zeit: Samstag 22. Juni 2019, 20:00
Format: E-Mail an gisler.michael@gmx.ch, Solution als ZIP Datei
Zeitlicher Verzug: < 1 Tag → 0.5 Note, jeder weitere Tag zusätzliche 0.5 Note Abzug