**Portfolio**

Für 4. Blockwoche

**Inhalt**

[**Bücher:** 1](#_Toc128565310)

[Schauspieler: 3](#_Toc128565311)

[**Filme:** 5](#_Toc128565312)

[**Clash of Clans** 6](#_Toc128565313)

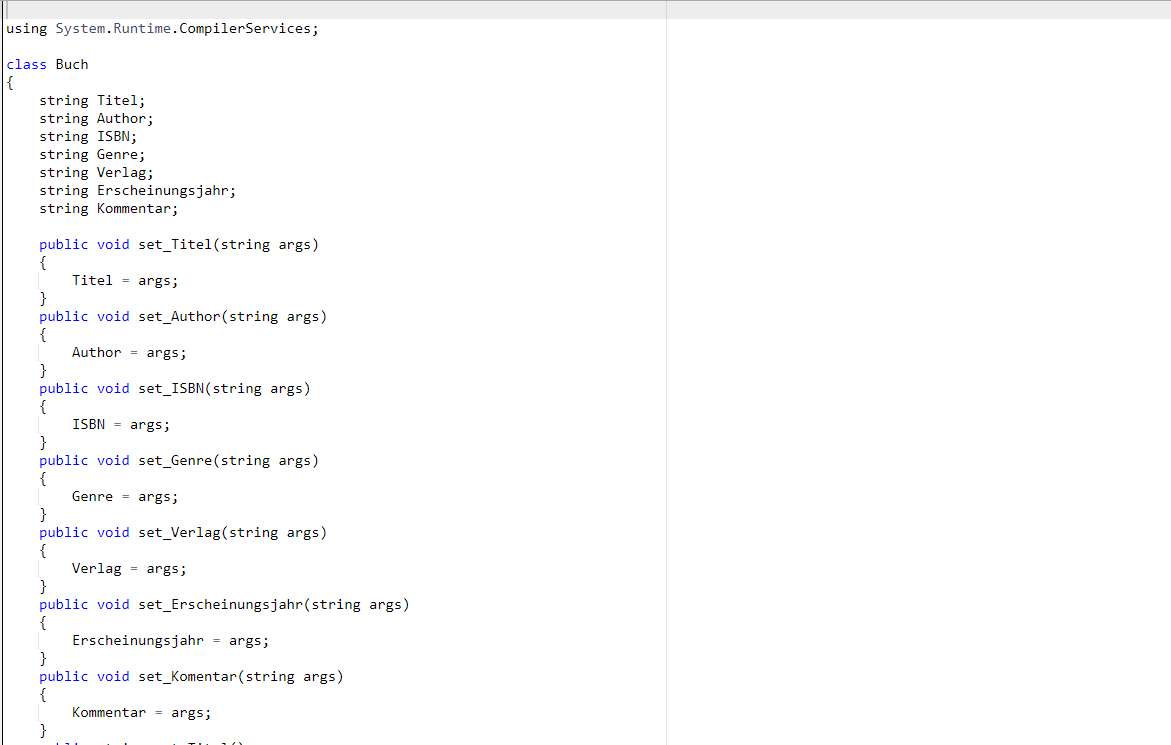
[**Taschenrechner:** 7](#_Toc128565314)

[**Bruch Rechner:** 10](#_Toc128565315)

# **Bücher:**

Bei der ersten Aufgabenstellung war die Aufgabe zu erfüllen das man im Code ein Titel, Author, ISBN, Genre, Verlag, Erscheinungsjahr und ein Kommentar definieren und abrufen konnte mit Klassen. Der Code war in 4 Klassen aufgeteilt einmal ein Konstruktor wo Titel und Author gesetzt und ausgegeben werden kann, einmal wo Titel, Author, ISBN und Erscheinungsjahr eingegeben und Ausgegeben werden kann, einmal wo die Aufgabe bei allen erfüllt werden soll und einmal wo nichts definiert werden soll.

Jetzt geht es erstmal um den unten gezeigten Code abschnitt.



Als erstes wird eine Klasse namens Buch erstellt diese Klasse wird Benötigt um von außen aus oder in der Datei auf diese Klasse zugreifen zu können. Wichtig ist hier zu beachten das diese nicht als Private gekennzeichnet ist. Sobald die Klasse als Private gekennzeichnet ist kann man diese dann nur noch in der Datei aufrufen und nicht mehr von Außerhalb.

In der Klasse werden nun die ganzen Werte eingetragen und vordefiniert. Hier wird dann jeweils zugegriffen.

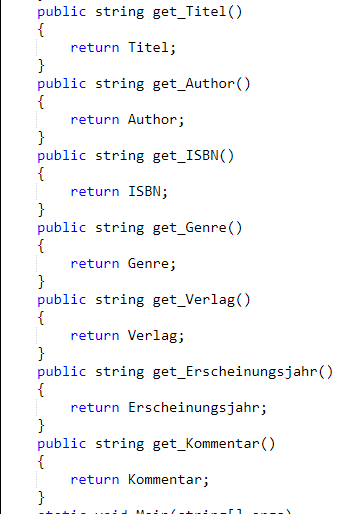
Als nächstes werden die Konstruktoren gesetzt. Hier wird eine „public void“-Methode verwendet. Das public sagt dem Programm das die Methode im ganzen Programm aufrufbar ist und das void sagt dem Programm das nichts zurückgegeben wird. Diese Methode vom Konstruktor wird zumeist verwendet um Konstruktoren zu erstellen in denen die oben genannten Variablen gespeichert werden.

Sobald das gemacht ist kann man schon dem Konstruktor einen Namen geben und die jeweiligen Datenwerde. Bei den wie oben genannten Konstruktoren wo nur Sachen gespeichert werden aber nichts ausgegeben wird bzw. in der Fach Sprache ein setter genutzt wird immer als Name z.b. **set\_**Title oder **set**Title verwendet. Das set am Anfang sagt den jeweiligen Entwicklern das hier Daten gesetzt werden aber nicht Ausgegeben werden.

Als nächstes muss definiert werden was für Daten gesetzt werden sollen, das macht man indem man hinter der Variable eine Klammer macht und dann mit dem jeweiligen Identifier eine variable definiert die dann genutzt wird in dem Konstruktor. In dem Bild wird als Identifier immer „string“ verwendet „string“ sagt dem Konstruktor das hier Normaler Text eingegeben werden soll.

Danach wird im Konstruktor der oben definierte Variable mit der vom Entwickler angegebenen Daten definiert.

Als nächstes geht es mit dem so genannten getter weiter.



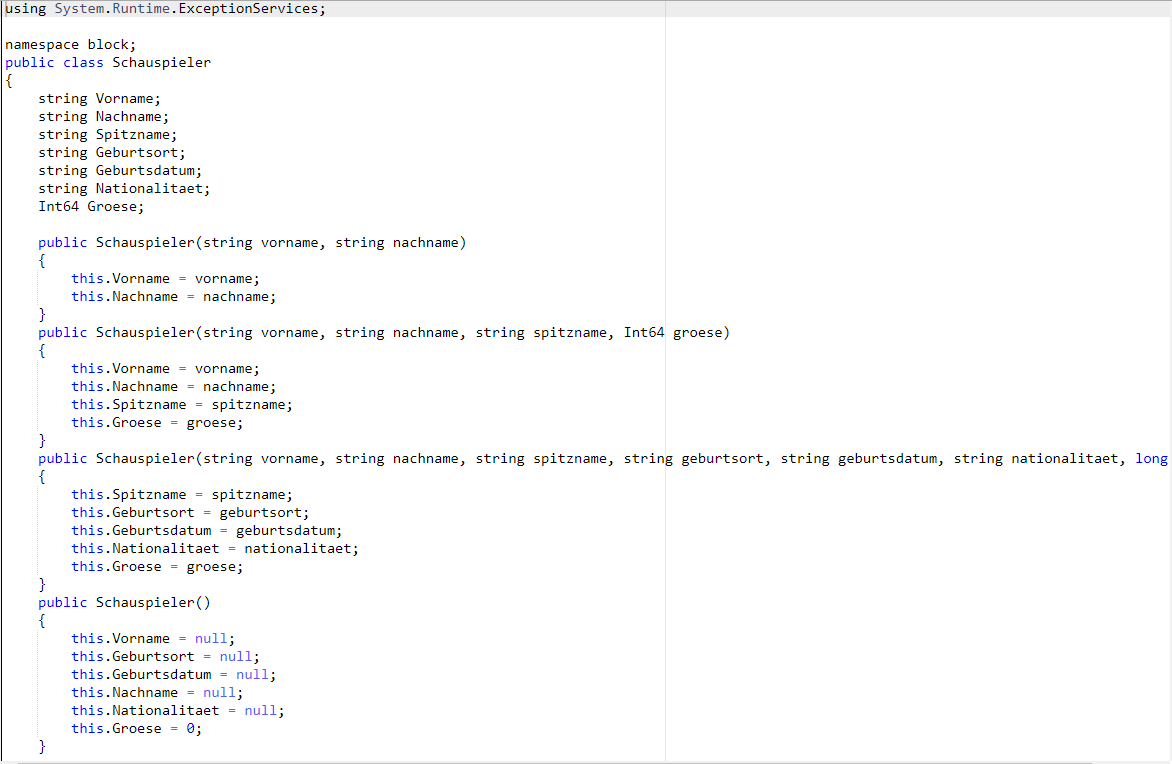
Der getter funktioniert im endeffekt genau so wie der setter allerdings sollten hier keine Datentypen oder werte in der Klammer definiert werden sollte es sich nur um die Ausgabe einer Variable gehen. Und anstatt das im Konstruktor eine Variable gesetzt wird wird hier die jeweilige Variable ausgegeben.

Bedienung:

Hier muss einfach nur die Klasse in die jeweilige Datei importiert werden und dann der Konstruktor definiert werden.

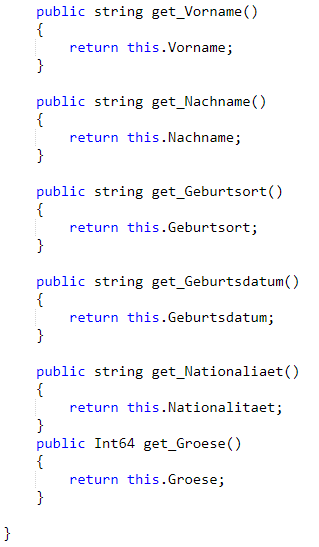
# Schauspieler:

Bei der zweiten Aufgabenstellung musste eine Klasse erstellt werden wo man Schauspieler definieren konnte.



Wie bei der letzten Aufgabe war hier fast das gleich zu machen nur mit anderen Variablen und einer anderen Verwendungszweck und einer anders definierten Variable. Oben wo die Variablen definiert sind findet man ein neune Datentyp. Mit dem Int64 Datenbank sagen wir einfach nur dem Projekt das in der Datei nur Zahlen im 64Bit System definiert werden können. Hier ist Int64 eigentlich nicht wirklich nötig zu definieren da man nie an die jeweilige grenze Stoßen wird wo auch ein sInt ausreicht. Ansonsten ist hier beim setter auch alles gleich wie bei der Aufgabe von oben. Das this in dem Programm sagt einfach nur das die Variable von oben genommen werden soll.

Auch beim getter ist hier alles genau wie bei der Aufgabe von oben weswegen hier alles eigentlich klar sein sollte.



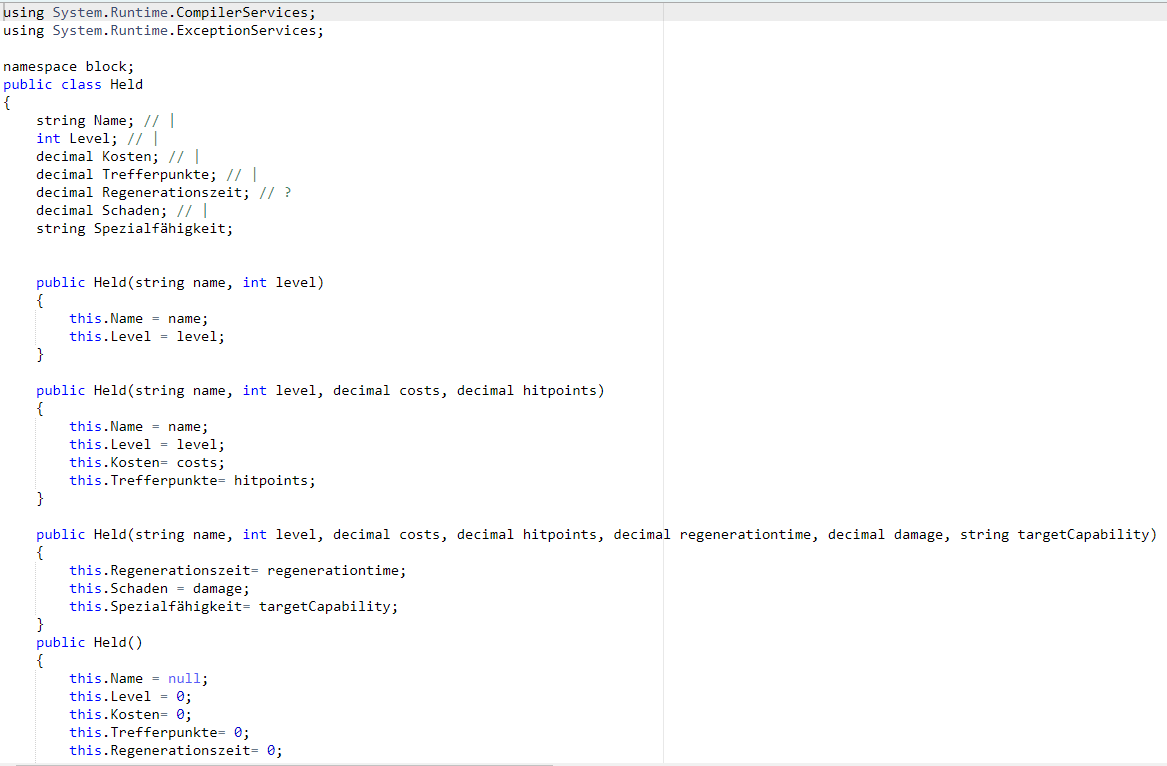
# **Filme:**

In der Aufgabe werden die letzten Zwei aufgaben zusammen Kombiniert und in einer Datei Verfasst. Man kann diese auch in mehrere Dateien machen was das ganze Übersichtlicher machen würde. Aber ansonsten ist hier wieder alles gleich

# **Clash of Clans**

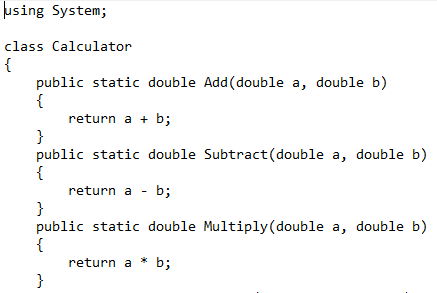
Hier wird ein Spiel Namens Clash of Clans als Aufgabe genutzt. Hier müssen wir Daten von diesem Spiel als Klassen nutzen.

Hier ist auch wieder alles gleich nur wieder bei den Variablen gibt es hier wieder ein Unterschied. Und zwar werden hier decimal auch als Datentypen verwendet. Decimal wird hier verwendet da in dem richtigen Spiel hier Daten verwendet werden die nach kommastellen haben was bei normalen Int oder String Lösungen nicht wirklich möglich ist, sondern nur über zwischen Schritte.

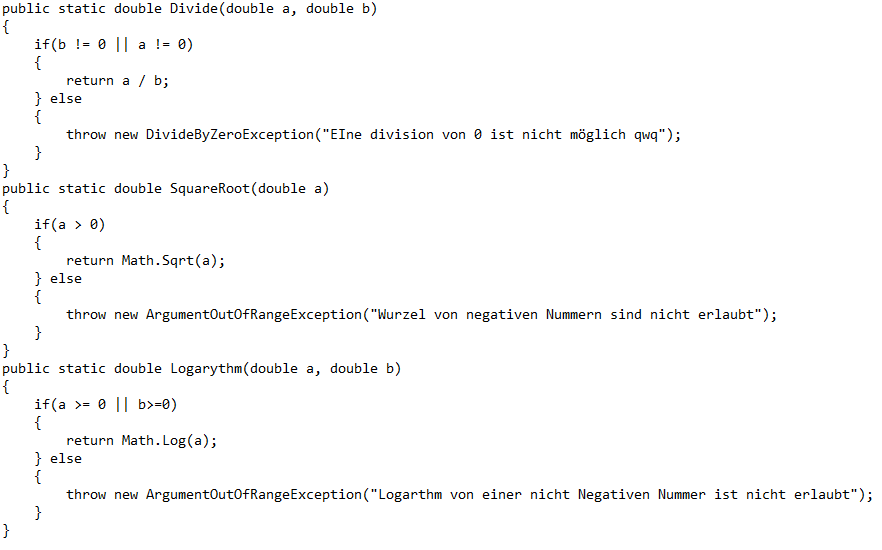


# **Taschenrechner:**

Bei dem Taschenrechner mussten wir mehrere Sachen erfüllen und zwar musste der Taschenrechner nicht nur Plus und Minus können, sondern auch Dividiert, Wurzel Rechnen, Logarithmus und vieles mehr.

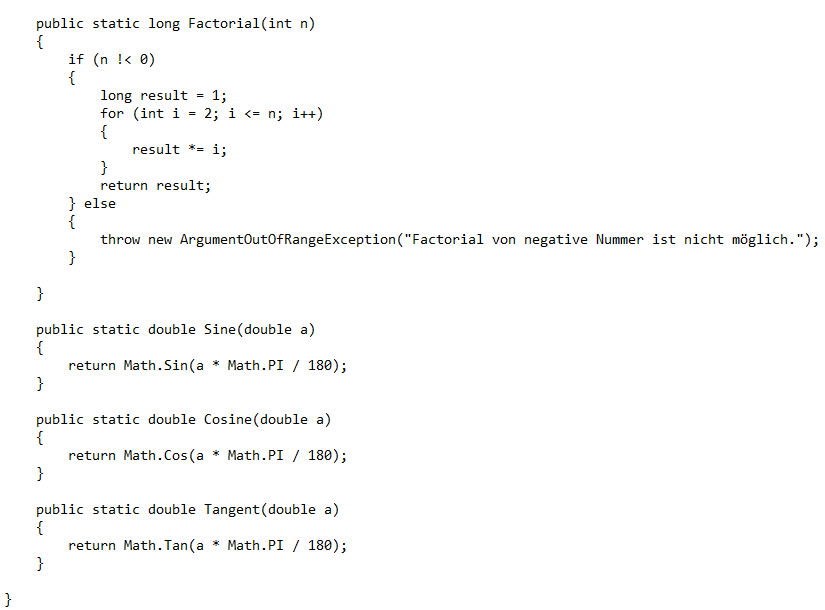


In der oben gezeigten Aufgabe ist hier der Konstruktor mit double definiert. Double wird für genutzt um bei Gleitkommerzahlen den genaustmöglichsten Wert zu bekommen. Hier ist es einfach nur wichtig das bei der jeweiligen Eingabe zwei Werte angegeben werden.



In der Aufgabe wird bei Divide als erstes gefragt ob b oder a nicht gleich 0 ist. Wenn das der Fall sein sollte dann wird eine DivideByZeroException geworfen mit einem bestimmten gesetzten Text.

Bei Square Root wird anders als bei den anderen nur eine Eingabe benötigt. Wenn hier a größer ist als 0 dann soll mit der von .NET definierten Sqrt Funktion die Wurzel berechnet werden und dann ausgegeben werden. Bei der Logarithmus Methode wird hier wieder gefragt ob a und b nicht kleiner ist als 0 und dann mit der schon von .NET definierten Math.Log Funktion der Wert ausgerechnet werden andererseits wird ein ArgumentOutOfRangeException geworfen.



In dem Screenshot werden long Methoden Verwendet da die Variable die Ausgabe lang sein soll.

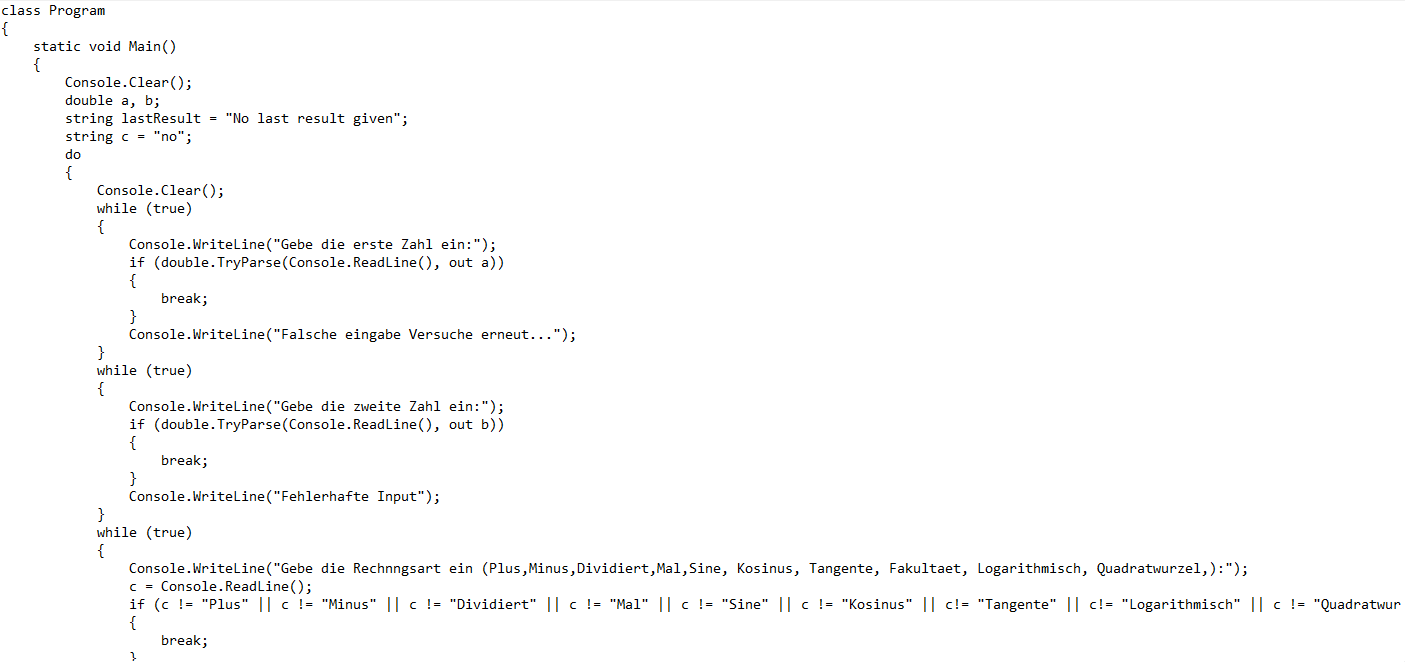
Als erstes wird hier überprüft ob n nicht kleiner ist als 0, wenn das der Fall ist dann wird eine Variable erstellt Namens Result und dem Wert 1 gegeben. Dann wird eine for Schleife erstellt wo eine i Variable erstellt drinnen und bei jeder Ausführung plus eins gerechnet wird und dann die Schleife erst dann beenden lässt bis i kleiner oder gleich ist wie n. Wenn das dann fertig ist wird die Ausgabe Modulo i gerechnet.

Dann wird result ausgegeben. Sollte bei der if Methode nicht der Fall sein dann wird ein Fehler mit dem ArgumentOutOfRangeException geworfen.

Bei Sin wird mit der .NET-Funktion a mit PI-Dividiert durch 180 gerechnet.

Bei Cosin wird auch wieder Math.Cos verwendet und die dann drinnen a mal PI dividiert durch 180 gerechnet.

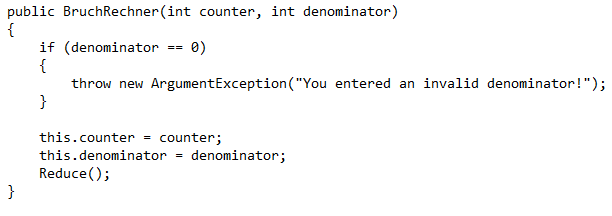
Bei Tan wird auch wieder dasselbe gerechnet allerdings am Anfang mit Math.Tan



Hier fängt das Programm jetzt richtig an. Als erstes wird die Konsole gelöscht um unnötige Texte die Davor waren zu entfernen. Dann wird eine double Variable Namens a und b erstellt, eine lastResult Variable mit dem Standard Wert „No last result given“ und eine Variable mit c und dem Standard wert namens no

Dann wird eine do while schleife erstellt das erst dann endet, wenn man am ende des Programms die Taste J drückt. Dann wird die Konsole wieder gelöscht und eine weitere while Schleife erstellt. Dann wird gefordert die erste und die Zweite Zahl zu schreiben, wenn hier ein falscher wert eingegeben wird dann soll eine Fehler Meldung Ausgegeben werden wo die Meldung zurück kommt das die Eingabe falsch war und es erneut versuchen soll. Dann soll man eingaben was ausgerechnet werden soll und dann mit einem Switch case überprüft Was ausgerechnet werden soll. Und dann gefragt ob nochmal gerechnet werden soll.

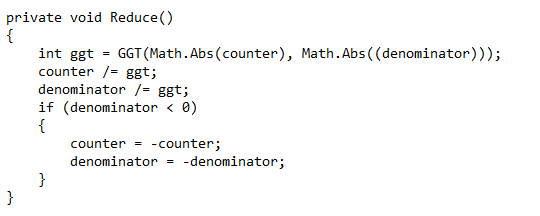
# **Bruch Rechner:**



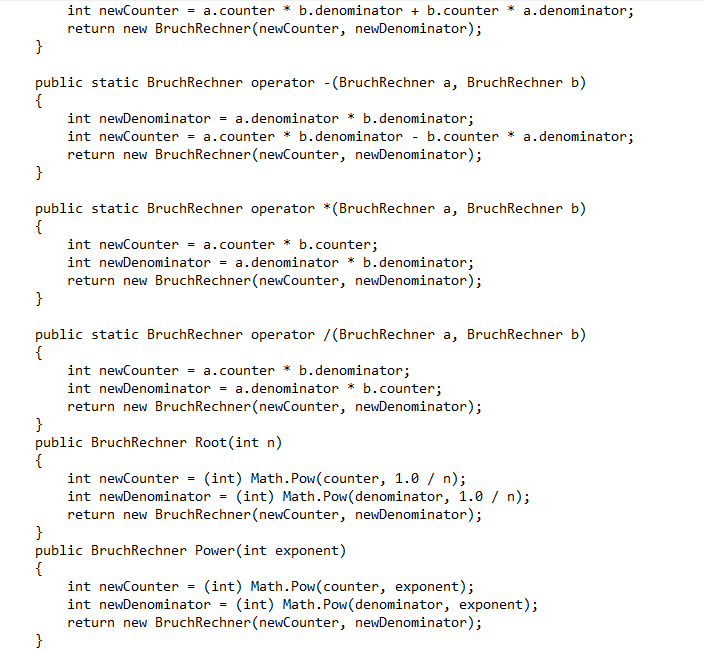
Bei dem Bruch Rechner wird wie bei allen in der Klasse zwei Variablen vor Definiert Namens counter und denominator. Diese zwei Variablen sind Numerisch angegeben.

Dann wird eine Klasse Namens BruchRechner erstellt wo counter und denominator als Eingabe wert Verpflichtend ist.

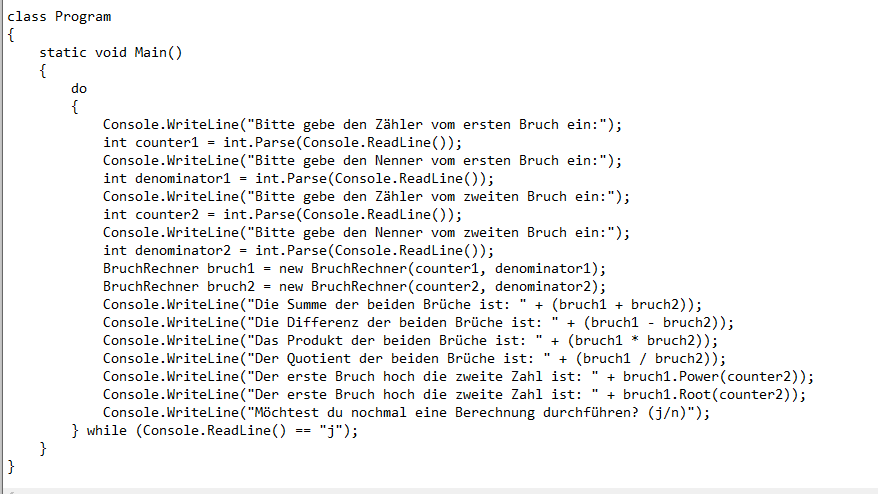
Dann wird gefragt ob denominator gleich null ist. Wenn ja wird direkt ein ArgumentException Fehler geworfen. Wenn das nicht der fall ist wird der Counter und denominator mit den angegebenen Daten gesetzt. Und dann Reduce() aufgerufen.



Die Reduce variable ist dafür da das der Größte gemeinsame Teiler gerechnet wird hier wird immer Math.Abs vom counter gerechnet und den vom denominator. Dann wird counter Dividiert gleich ggt gerechnet und das selbe bei denominator. Wenn der denominator kleiner als 0 ist dann soll counter gleich minus counter gerechnet werden und dass selbe auch wieder bei denominator.



Dann werden mit den definierten Klassen die Daten ausgerechnet diese basieren auch wieder mit den Werten von der vorherigen Aufgabe mit.



Dann zu aller Schluss werden dann die jeweiligen Daten auch wieder ausgegeben.

# **Zahlensystem um Rechner**

