Portfolio Dokumentation

1. Bücher

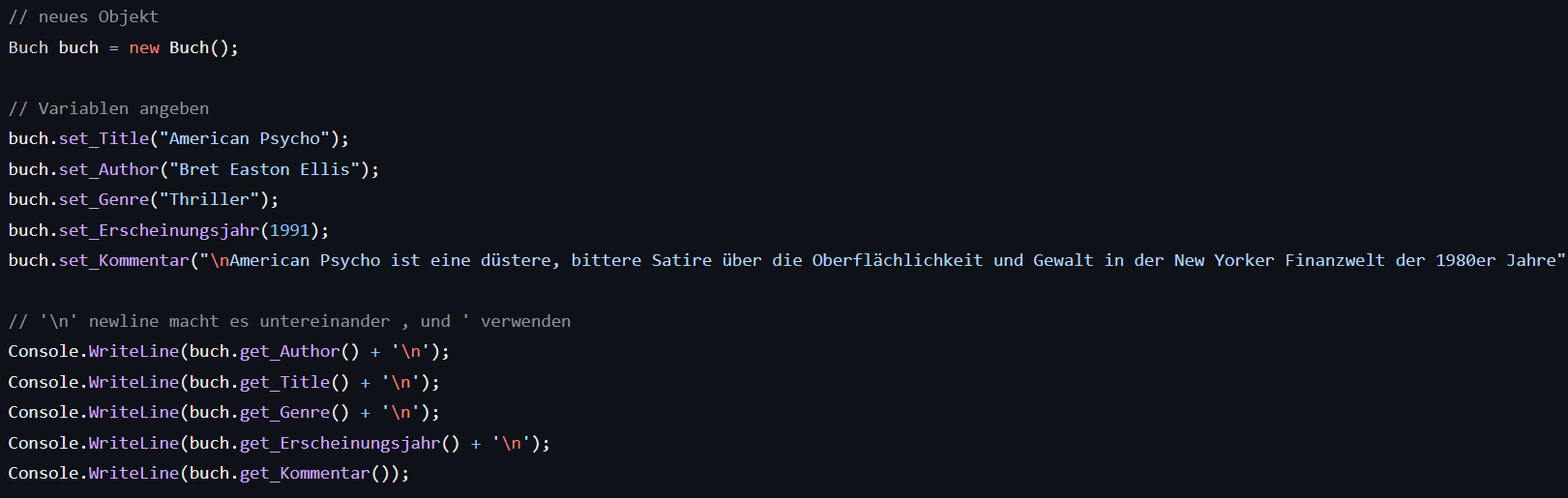
**1. Logik / Gedankengang:**

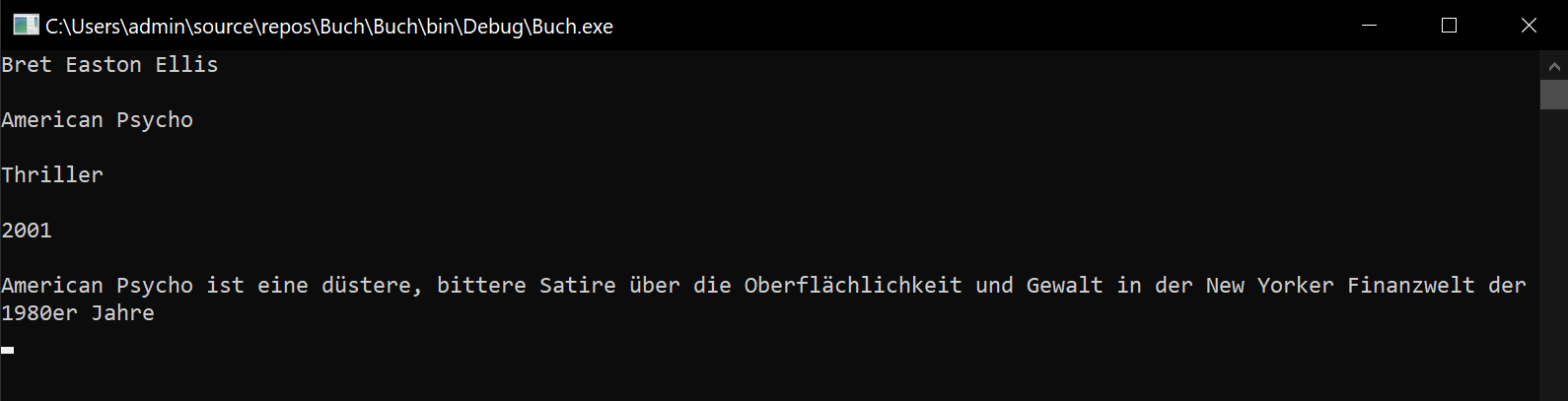
* Die Buchklasse ist ein Werkzeug, um quasi Informationen über Bücher zu speichern und zu organisieren.
* Wir können verschiedene Details über ein Buch speichern, wie den Titel, den Autor, die ISBN-Nummer und mehr.
* Die "getter" und "setter" helfen uns, auf diese Informationen zuzugreifen und sie zu ändern, wenn nötig.

**2. Vorgehensweise:**

* Programmieren: Wir haben die Buchklasse schrittweise aufgebaut und dabei verschiedene Konstruktoren für unterschiedliche Szenarien implementiert.
* Debuggen: Nachdem wir die Klasse erstellt haben, haben wir sie getestet, um sicherzustellen, dass sie wie erwartet funktioniert.

**3. Beispielcode:**

* Im Beispielcode erstellen wir ein Buchobjekt und setzen einige Informationen wie den Titel, den Autor und das Erscheinungsjahr.
* Dann zeigen wir diese Informationen in der Konsole an.



**4. Anmerkungen:**

* Die Buchklasse ist nützlich, um Informationen über Bücher zu organisieren und zu verwalten.
* Die Verwendung von "getter" und "setter" ermöglicht eine kontrollierte Datenzugriff und -änderung.

2. Schauspieler

**1. Logik / Gedankengang:**

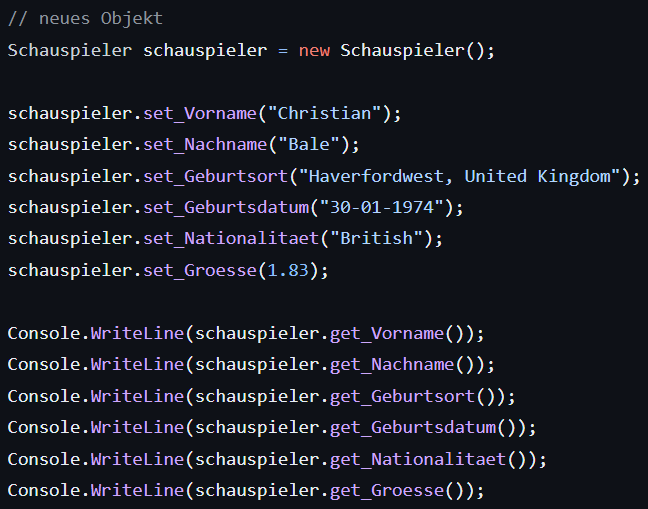
* Die Schauspielerklasse dient dazu, Informationen über Schauspieler zu verwalten, ähnlich wie bei der Buchklasse für Bücher.

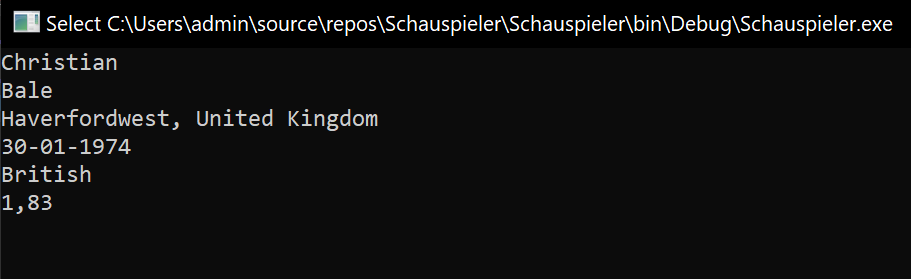
**2. Vorgehensweise:**

* **Programmieren:** Wir haben die Schauspielerklasse mit ähnlichen Attributen wie in der Buchklasse erstellt.
* **Debuggen:** Hier mussten wir nicht viel machen, da es quasi gleich aufgebaut war wie die Klassen und Objekte von der ersten Bücher Aufgabe.

**3. Beispielcode:**

* Ähnlich wie im Buchbeispiel haben wir Konstruktoren implementiert.
* Wir haben dann einige Informationen über einen Schauspieler festgelegt und sie in der Konsole angezeigt.

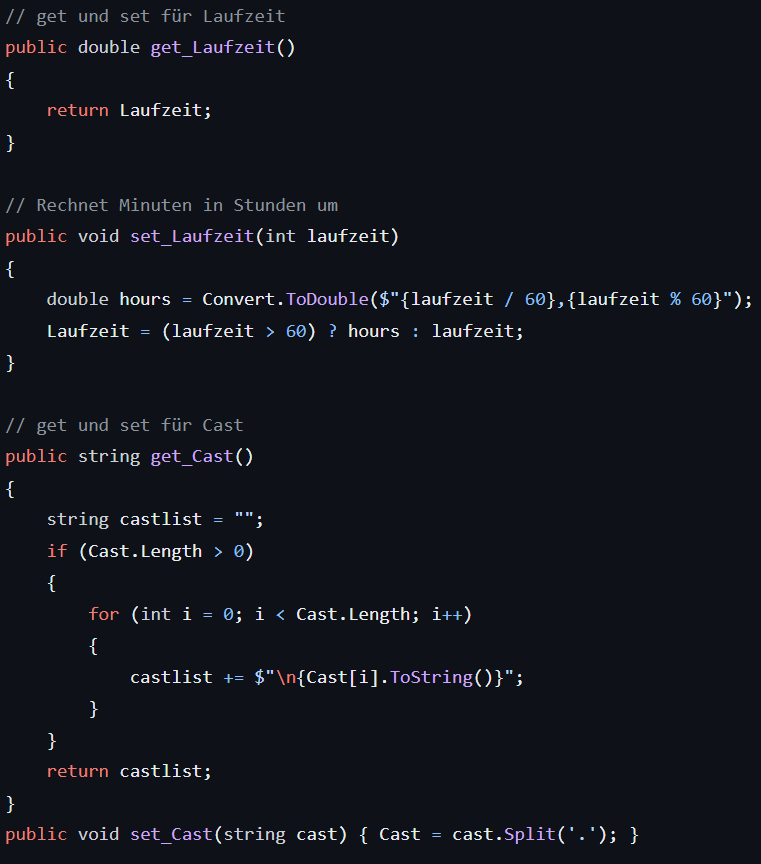


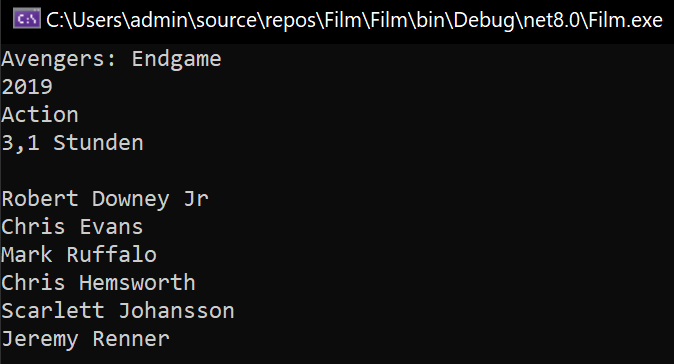
****

**4. Anmerkungen:**

* Die Schauspielerklasse bietet eine strukturierte Möglichkeit, Informationen über Schauspieler zu speichern und zu verwalten, ähnlich wie bei der Buchklasse für Bücher.

3. Film

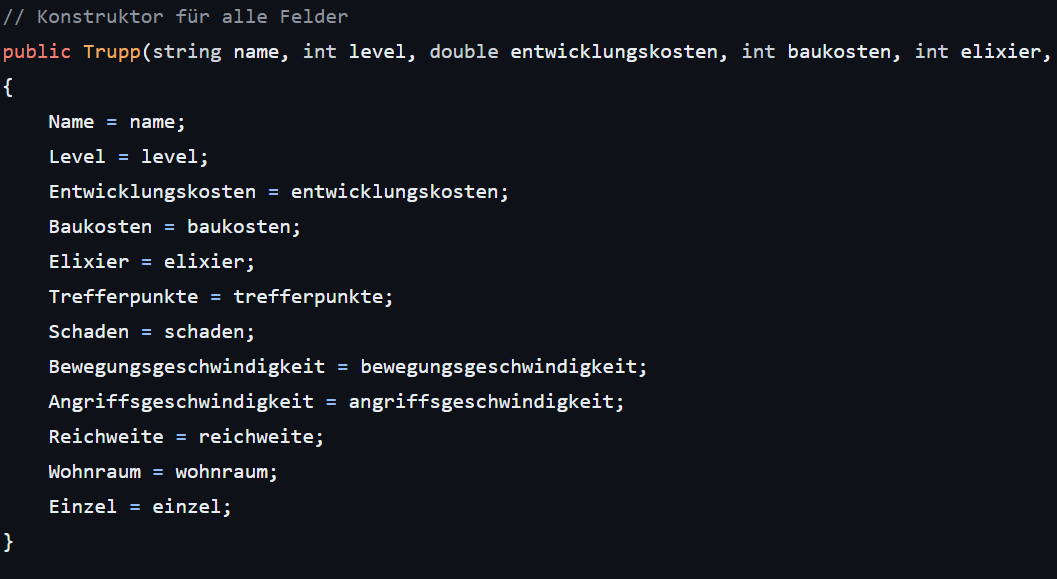
* **Debuggen:** Beim Entwickeln der Filmklasse haben wir ähnliche Logik und Struktur wie bei der Buchklasse verwendet, außer für die 2 neuen Techniken, was den Debugging-Prozess erleichtert hat.
* **Im** **Gegensatz** zu den vorherigen bei der Implementierung von Methoden haben wir neue Wege wie die Umwandlung der Laufzeit von Minuten in Stunden und die verarbeitete Liste von Schauspielern hinzugefügt

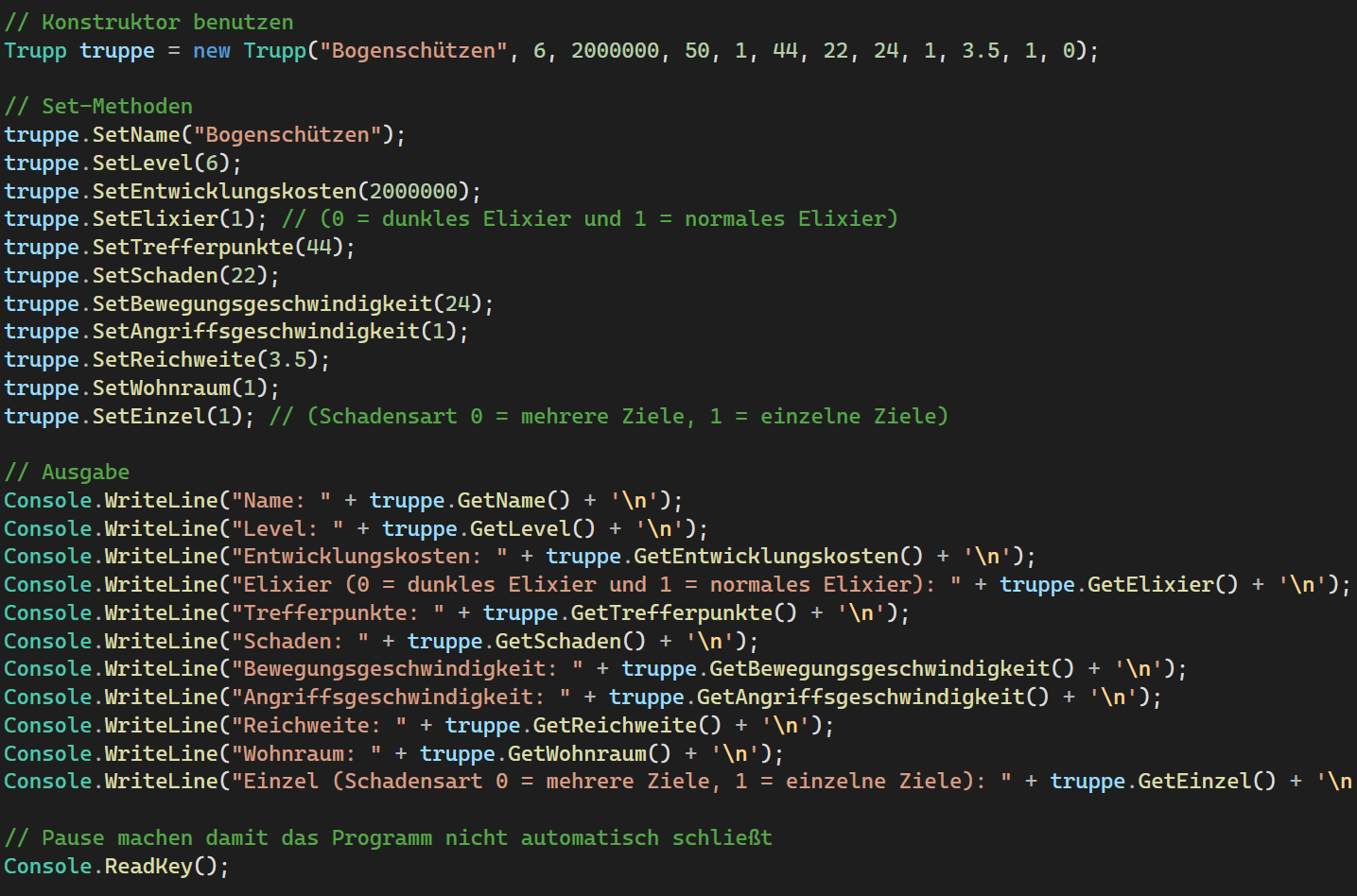


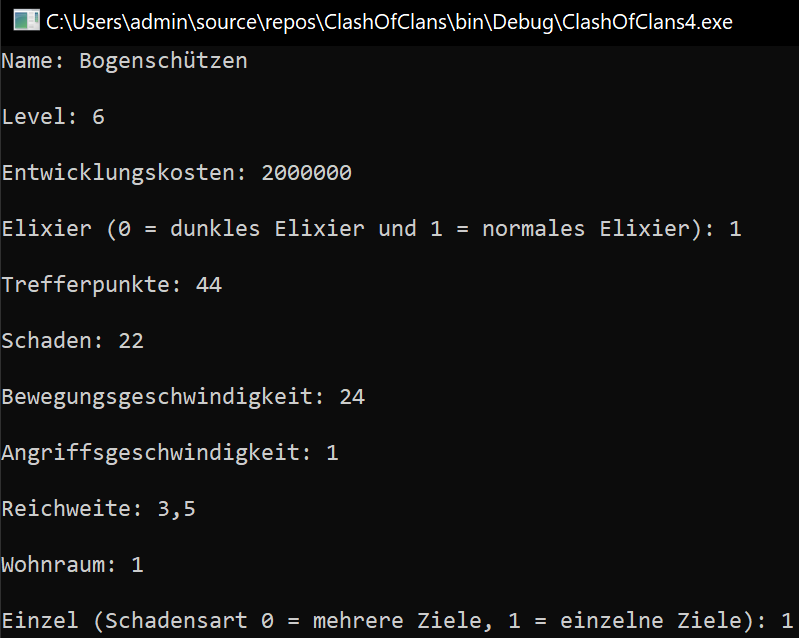
4. Clash of Clans

**Konstruktoren für verschiedene Szenarien:** Es wurden Konstruktoren implementiert, um Truppen/Helden mit unterschiedlichen Kombinationen von Informationen zu erstellen, darunter Name, Level, Entwicklungskosten, Baukosten, Elixier, Trefferpunkte, Schaden,

**Objekterstellung und Ausgabe:** Ein Trupp-/Heldobjekt wurde erstellt und seine Eigenschaften wurden festgelegt und anschließend ausgegeben.

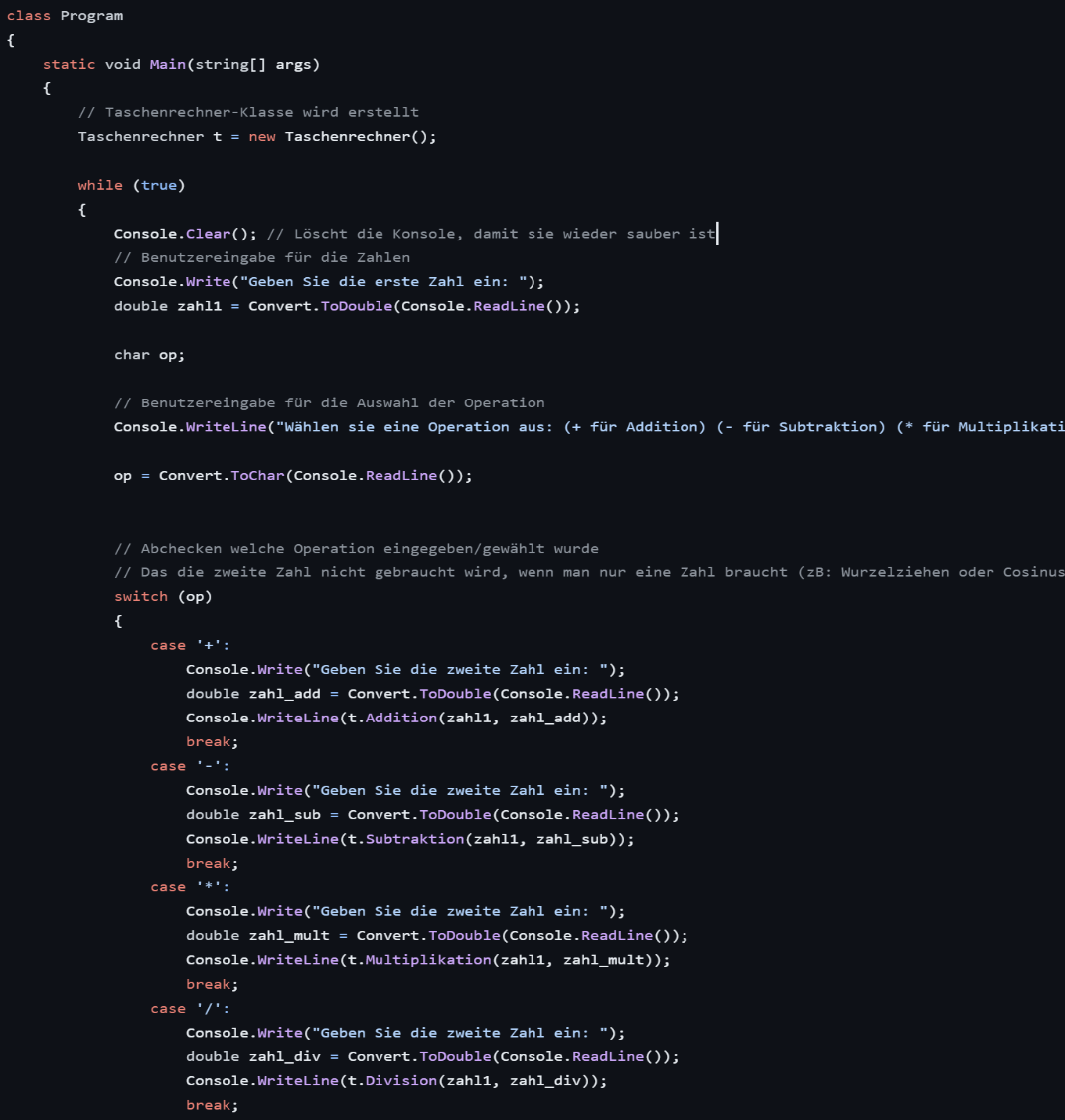






5. Taschenrechner



****

****

**Logik / Gedankengang:**

* Die Taschenrechner-Klasse wurde entwickelt, um grundlegende mathematische Operationen sowie fortgeschrittene Funktionen wie Sinus, Cosinus, Tangens, Potenzieren, Wurzelziehen und Logarithmus durchzuführen. Ziel war es, eine flexible und leicht zu bedienende Klasse zu erstellen, die verschiedene mathematische Operationen unterstützt.

**Vorgehensweise:**

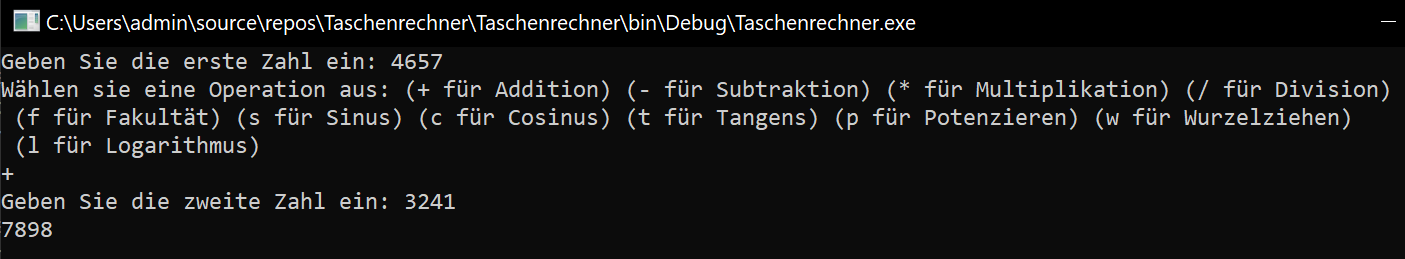
In der Hauptanwendung haben wir eine Schleife erstellt, die es dem Benutzer ermöglicht, beliebige Operationen auszuwählen und die entsprechenden Eingaben für die Berechnungen einzugeben.

Dazu haben wir eine Switch-Case-Struktur implementiert, um je nach Benutzereingabe die entsprechende mathematische Operation auszuführen und das Ergebnis auszugeben.

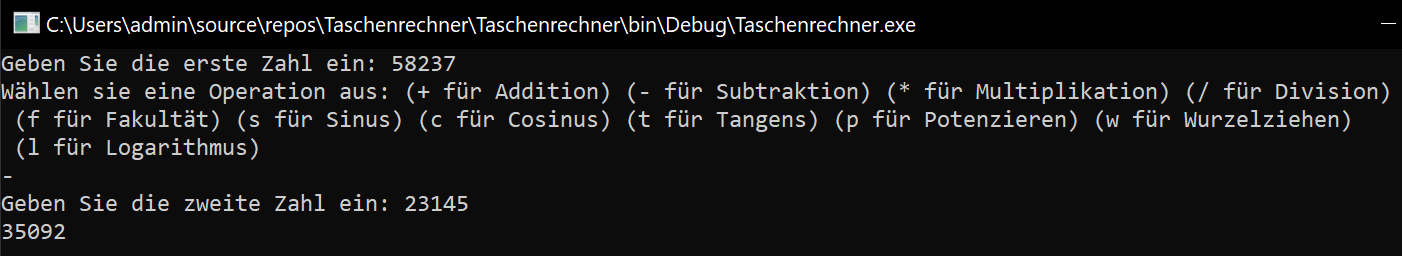
Zusätzlich haben wir das in eine „**While**“ Schleife gepackt damit es unendlich ausgeführt, quasi können wir den Taschenrechner so oft verwenden wie wir es wollen.

**Testfälle:**

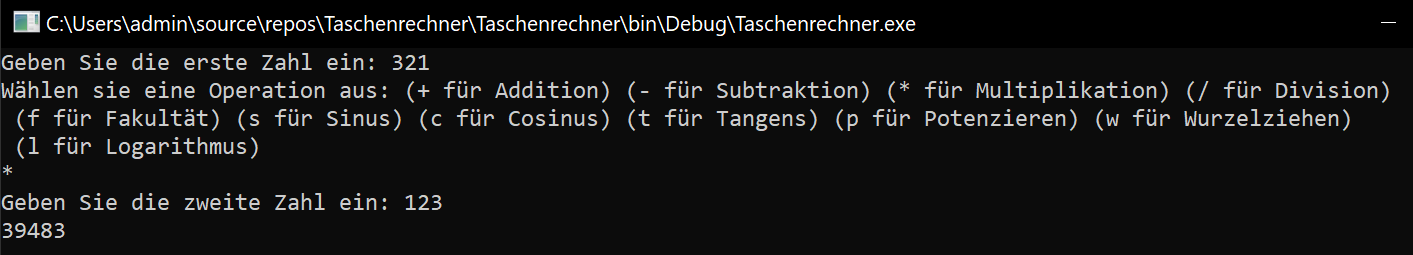
1. Addition



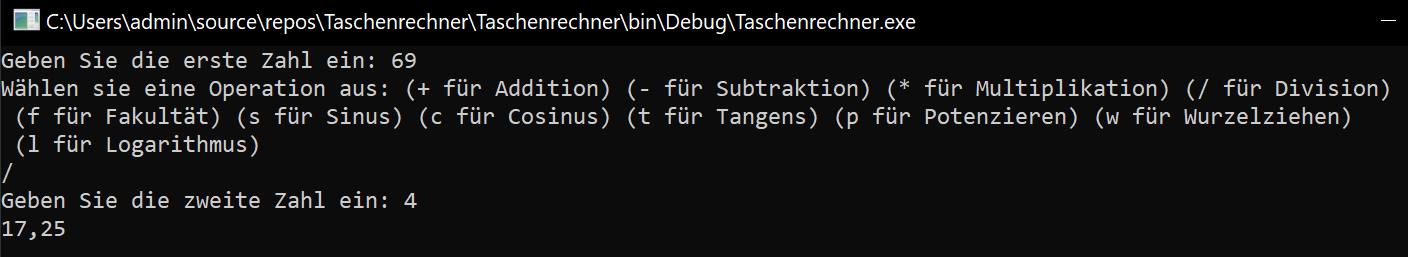
2. Subtraktion



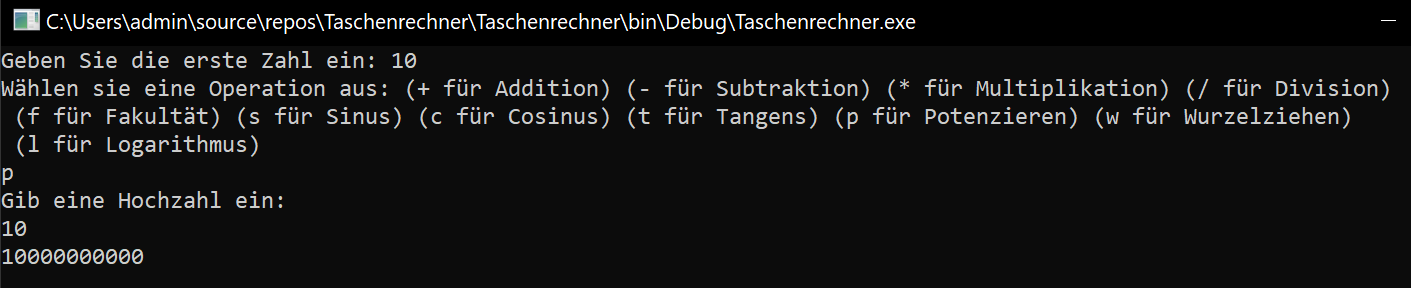
3. Multiplikation



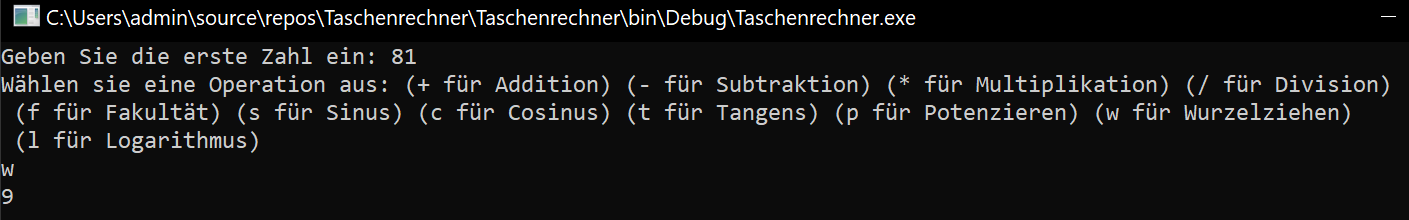
4. Division



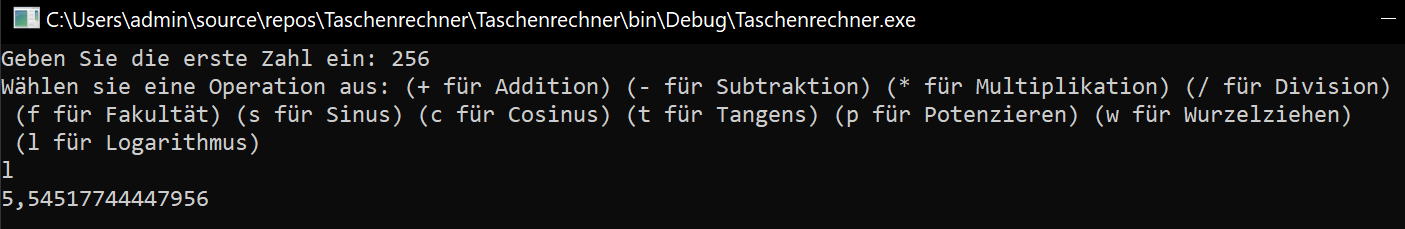
5. Potenzen



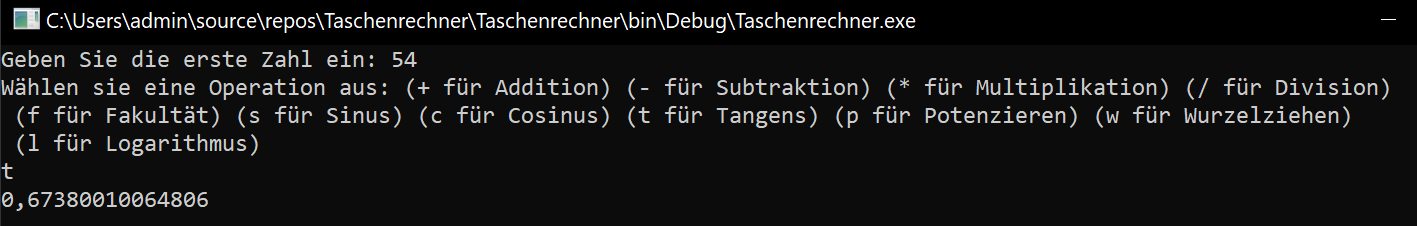
6. Wurzel ziehen



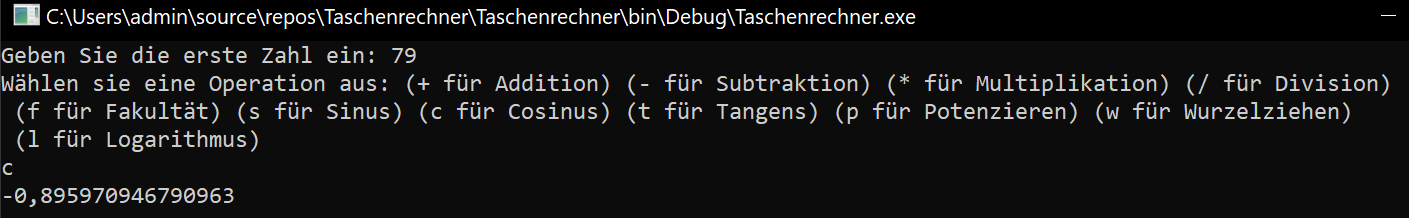
7. Logarithmus



8. Tangens

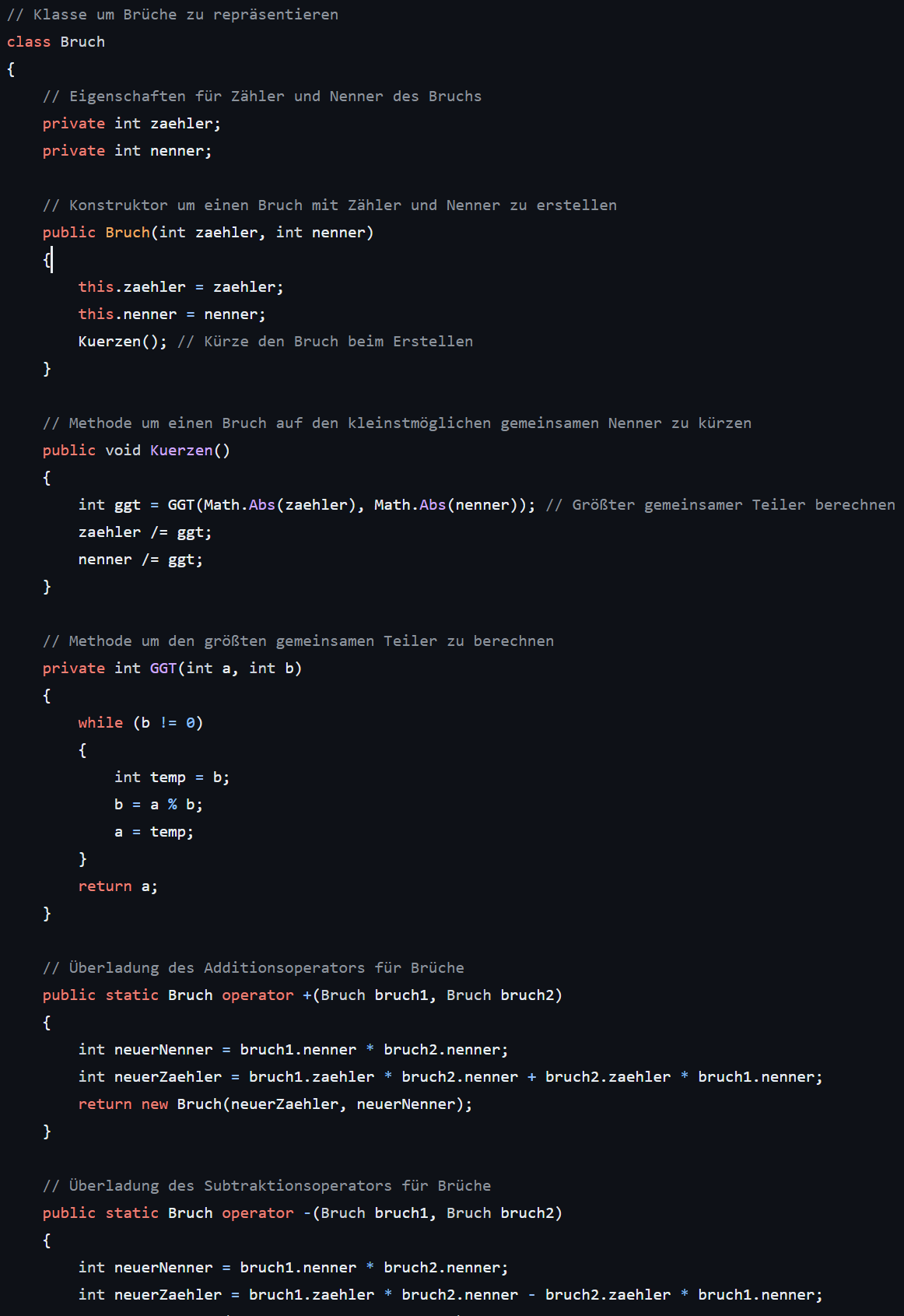


9. Cosinus



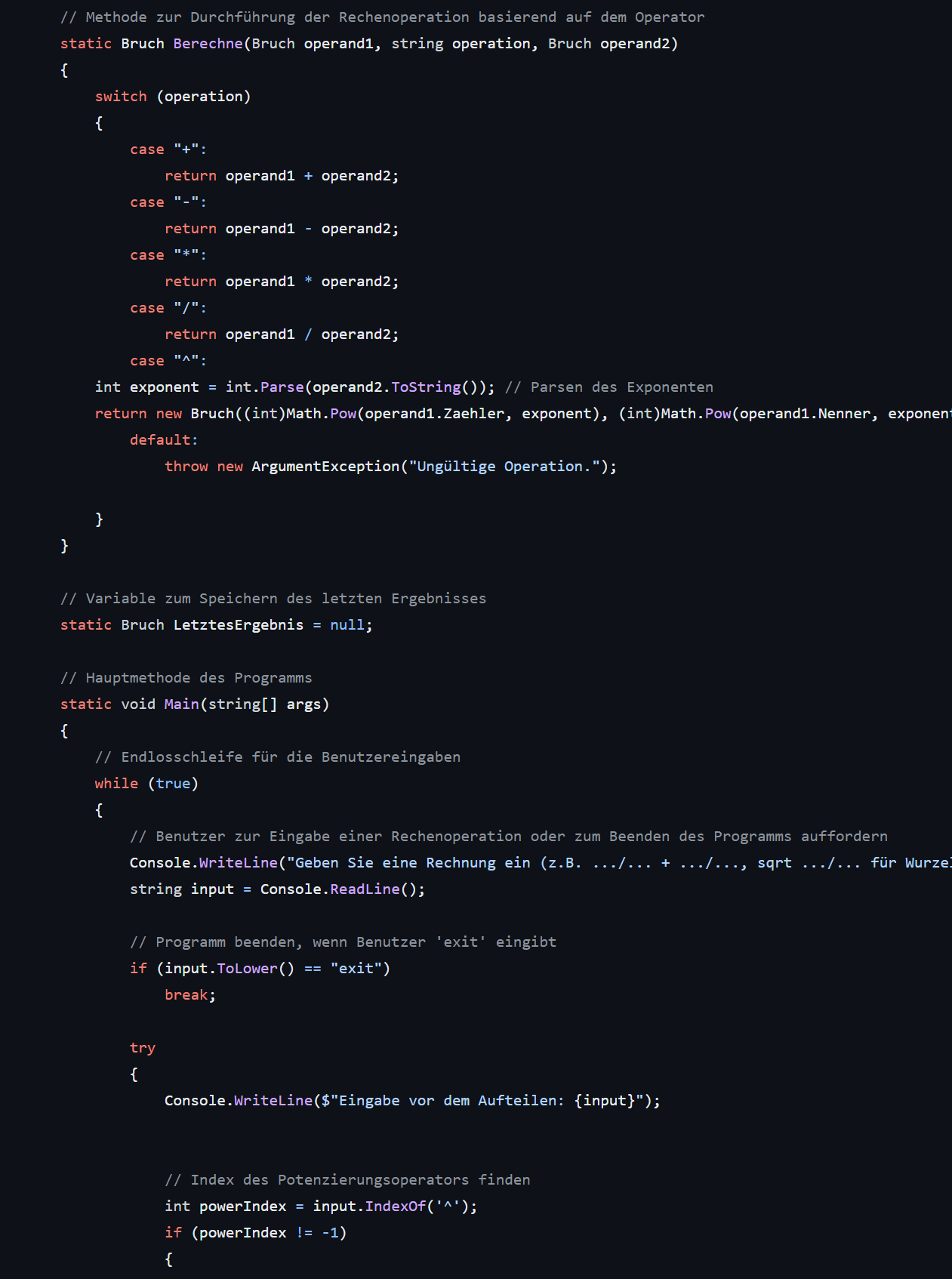
Unten Scrollen!

6. Bruchrechner

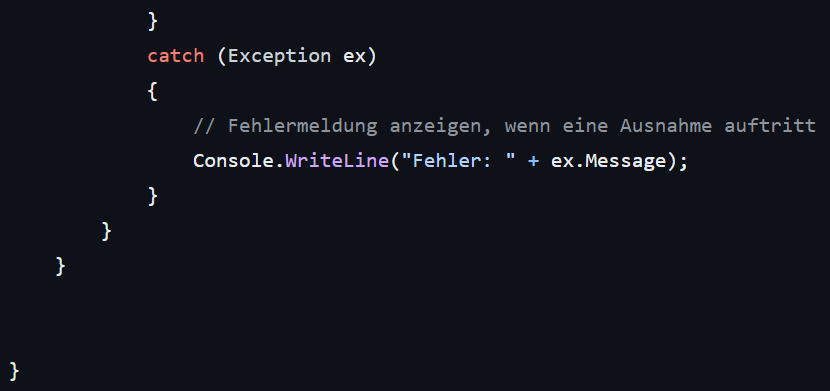


****

****

****

****

****

**1. Logik / Gedankengang:**

* Der Bruchrechner ist ein Programm, das erstellt wurde, um Operationen mit Brüchen durchzuführen. Es ermöglicht Benutzern, Brüche zu erstellen, zu kürzen und grundlegende mathematische Operationen wie Addition, Subtraktion, Multiplikation und Division durchzuführen. Außerdem unterstützt es fortgeschrittene Operationen wie das Ziehen von Wurzeln und Potenzieren von Brüchen.

**2. Vorgehensweise:**

* Klassenstruktur: Die Hauptklasse Bruch repräsentiert Brüche und enthält Methoden zur Durchführung von Operationen und zur Vereinfachung von Brüchen.
* Größter gemeinsamer Teiler (GGT): Die Methode Kuerzen() berechnet den größten gemeinsamen Teiler, um Brüche zu kürzen und sie in ihrer einfachsten Form darzustellen.
* Operatorüberladung: Die Klasse überlädt die mathematischen Operatoren (+, -, \*, /) für die Verwendung mit Brüchen, um die Rechenoperationen zu erleichtern.
* Fortgeschrittene Funktionen: Zusätzlich zu den grundlegenden Operationen unterstützt die Klasse auch das Ziehen von Wurzeln und Potenzieren von Brüchen.
* Benutzereingabe und -ausgabe: Die Start-Klasse verarbeitet die Benutzereingaben, führt die entsprechenden Operationen mit Brüchen aus und gibt die Ergebnisse aus.

**Benutzereingabe und -verarbeitung:** Die **Main**-Methode der **Start**-Klasse verarbeitet die Benutzereingaben, liest was eingetippt wurde und führt Operationen mit Brüchen aus

**Verwendung von Bruchobjekten:** Die **ParseBruch**-Methode wird verwendet, um Benutzereingaben in Bruchobjekte umzuwandeln, die dann für Berechnungen verwendet werden

**Fehlermeldungen:** Das Programm erkennt Fehler, quasi, dass ungültige Eingaben oder Operatoren eingegeben wurden und dem Benutzer gemeldet werden

**Testfälle:**

7. Zahlensystemumrechner