

LU11.A02 - Abstrakte Klasse für Taschenrechner

Ziel

Sie können eine umfassende Aufgabe mit diversen Techniken der OOP umsetzen.

Auftrag

Sie implementieren einen einfachen Rechner mit den Grundoperationen $+$, $-$, $*$, $/$, basierend auf dem gezeigten Klassendiagramm.



Hinweise

- Die beiden Klassen Reader und Tokenizer sind vorgegeben und sollen/dürfen durch Sie nicht verändert werden.
- Die Klasse ZeroDivisionError stammt aus der Python-Bibliothek und darf **nicht** selber implementiert werden. Sie ist der Vollständigkeit wegen im Diagramm skizziert.
- Die beiden Exception-Klassen (NumberFormatException und OperationException) erben von Exception. Das ist hier im Diagramm aus Platzgründen nicht mehr gezeigt.
- Pushen Sie jede Teilaufgabe mit Codegenerierung auf github.

Schritt 1

1. Akzeptieren Sie das Assignment im GitHub Classroom.
2. Klonen Sie das Repository in Ihre Entwicklungsumgebung.
3. Prüfen Sie, dass pytest und pylint funktionieren.

Schritt 2

Studieren Sie den Code der Klasse Reader.

- Was fällt Ihnen auf?
- Was bewirkt eine Klasse, die als **Singleton** deklariert ist? Studieren Sie dazu das WEB. Wir haben das Thema **Singleton** auch schon einmal kurz angesprochen.

Schritt 3

Implementieren Sie die beiden Exception-Klassen (in der Datei `exceptions.py`). Sie erben von der Klasse `Exception` aus der Python Bibliothek.

- `OperationException` gibt folgenden Hinweis aus: „ERROR: ungültiges Operationszeichen eingegeben!“. Dabei wird nicht mitgeteilt, was falsch eingegeben wurde.
- `NumberFormatException` gibt folgenden Hinweis aus: „ERROR: *VALUE* ist ein ungültiger Zahlenwert“. Hier steht *VALUE* als Platzhalter für den konkret falsch eingegebenen Text.

Schritt 4

Implementieren Sie die abstrakte Klasse `MathOp` (in der Datei `math_operations.py`) gemäss den Beispielen in der Theorie.

- Halten Sie sich an das Klassendiagramm. Es zeigt ihnen - durch *kursive* Nennung - die abstrakte Methode.
- Implementieren Sie die andere Methode als Property. Sie ist nicht abstrakt und muss daher einen Code enthalten.

Testen Sie diesen Schritt mit dem Testfall `test_math_op_instantiate` in der Datei `test_mathop_class_instantiation.py`.

Schritt 5

Implementieren Sie die 4 Klassen `Adder`, `Subtractor`, `Multiplier` und `Divider` für die konkrete Umsetzung der mathematischen Operationen. Dabei müssen Sie die abstrakte Methode der Oberklasse `MathOp` überschreiben.

- Erstellen Sie diese 4 Klassen jeweils in einer eigenen Datei.
- Die Funktion `execute_op` „weiss“ jeweils, welche Operation sie mit den beiden Zahlenwerten ausführen muss.
- Beachten Sie, dass die Klasse `Divider` bei einer Division mit 0 die entsprechende Exception (`ZeroDivisionError`) werfen muss.

Testen Sie die Klassen mit den entsprechenden Testfällen aus der Datei `test_math_operations.py`.

Schritt 6

Implementieren Sie nun die Klasse `Calculator` in der Datei `calculator.py`.

- Halten Sie sich an die Vorgaben des Klassendiagramms und an die Hinweise in der Datei.
- Wichtig ist, dass Sie hier keine Exceptions fangen und auswerten. Das geschieht dann alles in

der main-Routine.

- Die Methoden sind entsprechend der folgenden Sequenzdiagramme zu implementieren:

read_input



create_concrete_op

Hier am Beispiel der Addierfunktion (Klasse Adder) gezeigt:



calculate

Diese Methode schreibt das Ergebnis in den Stdout.



Testen Sie die Klasse mit den Testfällen aus der Datei `test_calculator.py`.

Schritt 7

Implementieren Sie in der Datei `main.py` die Hauptroutine (`main`). Hier werden die Exceptions gefangen und verarbeitet. Das heisst, dass jeweils eine entsprechende Meldung auf den Stdout (mit `print`-Befehl) ausgegeben wird. Halten Sie sich dabei an das folgende Sequenzdiagramm.



Führen Sie das Programm aus. Sie sollten eine vergleichbare Ausagbe erhalten.



Abgabe

Die Teilaufgaben werden laufend auf GitHub mittels push abgelegt.

[M320-LU11](#)



René Probst, bearbeitet durch Marcel Suter

From:
<https://wiki.bzz.ch/> - **BZZ - Modulwiki**

Permanent link:
https://wiki.bzz.ch/de/modul/m320_2024/learningunits/lu11/aufgaben/calculator?rev=1788091539

Last update: **2026/08/30 14:05**

