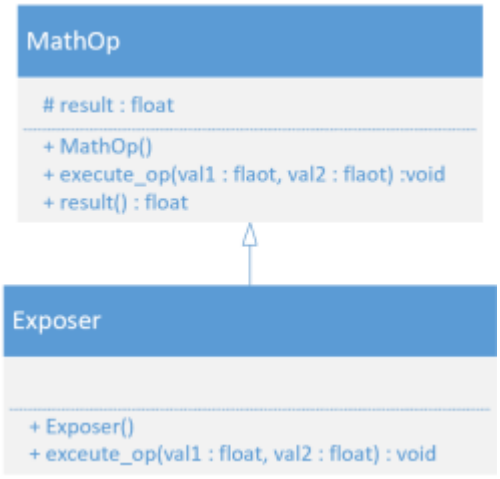
Aufgabe 4 - Polymorphie am Beispiel Taschenrechner

Ziel

Sie können eine bestehende OO-Anwendung erweitern.

Auftrag

Der von Ihnen erstellt Rechner soll neben den 4 Grundoperationen auch noch die Exponentialfunktion anbieten.



Als Operationszeichen dient das Zeichen ^.

Vorgehen

Verwenden Sie als Basis das Projekt aus Aufgabe LU07-3. Erstellen Sie einen neuen Branch mit Namen exposer. Erstellen Sie nun in diesem Branch Ihre Lösung.

1. Studieren Sie das Klassendiagramm der Klasse `Tokenizer`. Wie können Sie - ohne deren Code zu verändern - weitere Operationszeichen zufügen? Passen Sie entsprechend **Ihren Code** an.
2. Erstellen Sie die neue Klasse `Exposer`. Suchen Sie im Internet nach einer Lösung, wie Sie in Python eine Potenz berechnen können.
3. Ergänzen Sie den Code, so dass die Operation ^ auch erkannt und die entsprechende Klasse instanziiert wird.
4. Testen Sie Ihre Anwendung. Die Eingabe 2^6 müsste 64 ergeben.

Abgabe

Keine.

Zu Polymorphie

Sie haben hier und jetzt gesehen, wie kräftig das Konstrukt der Polymorphie bei OO sein kann. Es bedarf aber einer guten Planung der Klassen und deren Methoden, damit Polymorphie auch wirklich genutzt werden kann. Diesen Teil - das OOD (object oriented design) - werden Sie im Rahmen des Schulunterrichts leider nicht erlernen.
Darum ist es wichitg, dass Sie die gewonnenen Erkenntisse mit auf Ihren beruflichen Weg nehmen und bei Gelegenheit sich weiteres Wissen rundum OO - so z.B. auch den Umgang mit Design Pattern - im Selbststudium aneignen.

© René Probst

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

Permanent link:
https://wiki.bzz.ch/modul/archiv/m320/learningunits/lu99/aufgaben/lu08-calculatorext

Last update: **2026/08/30 14:10**