Datenstrukturen
Übung – SS2020

Abgabe: 24.05. bis 24Uhr.
Abgabeort: Bitte laden Sie Ihre Klasse im OPAL-Bereich „Abgabe_Pflichtaufgabe_3“ hoch. Ergänzen Sie bei Ihrer hochgeladenen Klasse die Namen (Vor- und Nachname) aller beteiligten Personen. Sie können Ihre Abgabe bis zur Abgabefrist beliebig oft löschen und erneut einreichen.

Wichtig: Da es sich um eine Prüfungsvorleistung handelt, ist es wichtig, dass jeder Teilnehmer Zugang zu allen notwendigen Informationen hat. Deswegen bitten wir Sie Fragen direkt ins Forum, in den Thread „Fragen zur Pflichtaufgabe 3“ zu stellen.

Sind Sie in einer Gruppe eingetragen wählen Sie den Gruppennamen, auch wenn sie einzeln bzw. als Teil der Gruppe abgeben. Schreiben Sie in jedem Fall Ihren Namen als Kommentar oben in die Klasse. Sie in keine Gruppe eingetragen sein, wählen sie statt "GroupX" bitte "Name", wobei Name die Form "$Nachname_$Vorname" besitzt.
Packen Sie Ihre Klasse in das Package "calculator_group_x" bzw. statt "group_x" Ihren Namen in folgender Form "calculator_nachname_vorname".

Für die Abgabe: Packen Sie all Ihre Klassen in ein .zip Archiv mit dem Namen "Calulator_GroupX.zip" [Anmerkung: .rar ist kein .zip]

Prüfungsvorleistung 3 – Der Taschenrechner

Schreiben Sie eine Klasse "Calulator_GroupX" die folgendes Verhalten durch das gegebene Interface Implementiert.

Ihre Aufgabe ist es einen kleinen Taschenrechner zu schreiben. Dieser Taschenrechner soll mit einem String in Infix-Notation arbeiten (https://de.wikipedia.org/wiki/Infixnotation). Dabei existieren die folgenden Zeichensätze:

Der Zeichensatz A soll dabei aus den Zeichen {'+', '*'} bestehen.
Der Zeichensatz B soll dabei aus ganzen Zahlen bestehen. [Der Datentyp „int“ reicht aus]
Der Zeichensatz C soll dabei aus den Zeichen {'(',')'} bestehen.
Jedes Zeichen aus einem Zeichensatz sei durch ein Leerzeichen zu seinem Nachfolger abgegrenzt.

Gegeben sei ein Ausdruck T.

T sei korrekt, wenn gilt:
-ein Zeichen aus A muss Links und Rechts ein Zeichen aus B oder C besitzen
  -ein Ausdruck '... 24 + (...' sei in Bezug auf das Snippet korrekt
  -ein Ausdruck '... 42 + + ...' sei in Bezug auf das Snippet nicht korrekt

-ein Zeichen aus B muss Links und Rechts entweder ein Zeichen aus A oder C oder kein Zeichen besitzen
- ein Ausdruck '12 + ...' sei in Bezug auf das Snipet korrekt
- ein Ausdruck '... + 12 ' sei in Bezug auf das Snipet korrekt
- ein Ausdruck '... 12 + 12 ...' sei in Bezug auf des Snipet korrekt
- ein Ausdruck '... 12 12 ...' sei in Bezug auf des Snipet nicht korrekt

-zu jeder '(' exestiert eine schließende ')'
-eine ')' darf nicht direkt nach einer '(' folgen
-es muss gelten '(T)' wobei T wieder ein korrekter Ausdruck ist
-auf ein Zeichen '(' muss links ein Zeichen aus A oder '(' oder kein Zeichen folgen, rechts ein Zeichen aus B oder '(' folgen
-auf ein Zeichen ')' muss links ein Zeichen aus B oder ')' folgen, rechts ein Zeichen aus A oder ')' oder kein Zeichen folgen
-ein Ausdruck '... ((T) + 12) ' sei in Bezug auf das Snippet korrekt
-ein Ausdruck '... ( ) ... (12 + (T) ...' sei in Bezug auf das Snippet nicht korrekt

Korrekte Ausdrücke:
(12 + (4 * 3) + 5 * 10)
(10)
10
10 + (2 * 12) + 8
5 + ((8 + 4) * 6)

Inkorrekte Ausdrücke:
(12 + 4
12 + + 4
12 + * 2
+
()()()
Der Leere Ausdruck sei kein Valider Ausdruck

Implementieren Sie das gegebene Interface 'Calulator' in Java oder Python

Java: boolean isValidExpression(String expression)
python: def isValidExpression(expression : str) -> bool
Diese Funktion bekommt einen String und soll prüfen, ob der Ausdruck nach obiger Definition korrekt ist. Gehen Sie davon aus das „expression" nicht null ist und nur aus den Zeichensätzen mit den dazugehörigen Trennzeichen besteht.

Java: int calculate(String expression)
python: def calculate(expression : str) -> int
Diese Funktion bekommt einen validen Ausdruck ungleich null, berechnen Sie dessen Wert.

Es ist davon auszugehen, dass im String nur valide Zeichen aus den Zeichensätzen A, B, C enthalten sind und diese stets mit Leerzeichen getrennt sind. Das Erstellen anderer Klassen ist erlaubt.