Uitrekenen van postfix-expressies

In deze opgave kijken we naar eenvoudige expressies, waar alleen de volgende elementen in voorkomen:
– positieve gehele getallen
– de operatoren +, -, * en /, waarbij / de gehele deling is.

In de "normale" notatie van expressies staan de operatoren tussen de operanden (de getallen). We noemen dat *infix-notatie*. In deze notatie zijn er soms haakjes nodig om duidelijk te maken hoe de expressie gelezen moet worden, bijvoorbeeld (15 + 5) * 2 is iets anders dan 15 + 5 * 2. Om een expressie goed te lezen is bovendien kennis nodig van prioriteiten (vermenigvuldigen gaat voor optellen).

Er zijn ook notaties waarin het gebruik van haakjes en kennis van prioriteiten niet nodig is. In deze opdracht kijken we naar één zo'n notatie, en wel de *postfixnotatie*. In die notatie staat de operator altijd achter de twee operanden: in plaats van
5 + 7, schrijven we 5 7 +, in plaats van (15 - 5) * 2 schrijven we 15 5 - 2 *, en in plaats van 15 - 5 * 2 schrijven we 15 5 2 * -. De volgorde waarin de getallen en de operatoren staan, bepaalt de 'uitrekenvolgorde', dus kennis over prioriteiten en haakjes zijn niet nodig.

De volgende tabel toont nog meer voorbeelden van deze notatie.

Expressie	*Postfixexpressie*	*Waarde*
12	12	12
12 - 5	12 5 -	7
(12 + 8) * 4	12 8 + 4 *	80
12 + 8 * 4	12 8 4 * +	44
12 + 8 * 4 / 2	12 8 4 * 2 / +	28
8 * 2 + 8 * 4 / 2	8 2 * 8 4 * 2 / +	32
((5 + 7) * (10 - 7)) / 12	5 7 + 10 7 - * 12 /	3
10 - 4 -2	10 4 - 2 -	4
10 - (4 - 2)	10 4 2 - -	8
100 / 10 / 2	100 10 / 2 /	5
100 / (10 / 2)	100 10 2 / /	20

Merk op, dat we 10 - 4 -2 moeten lezen als (10 - 4) - 2, en niet als 10 - (4-2). In postfixnotatie wordt 10 - 4 -2 dus 10 4 - 2 -. Voor / geldt iets dergelijks.

De waarde van een postfixexpressie kan gemakkelijk worden uitgerekend met behulp van een stack.

OPGAVE
Ga na hoe je een stack kunt gebruiken om de waarde van een expressie in postfixnotatie uit te rekenen. Gebruik de voorbeelden uit de tabel om dit uit te zoeken.

In deze opdracht ontwerpt en implementeert u een applicatie waarmee postfix expressies gelezen en uitgerekend worden. De volledige applicatie is getoond in figuur 1.

FIGUUR 1 Applicatie die postfixexpressies inleest en uitrekent

In het expressieveld wordt een expressie ingevoerd in postfixnotatie, waarbij getallen gescheiden moeten worden door spaties. Verder is het gebruik van spaties vrij (je mag ook rond een operator spaties zetten, maar het hoeft niet). Bij klikken op de knop Bereken waarde moet de waarde van de expressie getoond worden in het veld Waarde. Eventuele foutmeldingen verschijnen in een foutLabel onderin het frame.

Bouwstenen

Bij het uitwerken van de opdracht gaat het alleen om het uitrekenen van de waarde van de expressie; het splitsen van de expressie in getallen en operatoren en het maken van de gebruikersinterface is al gedaan. Daartoe zijn de volgende bouwstenen beschikbaar.

– Een klasse Token. Een token is een lexicale eenheid (een 'woord') uit een programma. Voorbeelden van tokens uit Java zijn "public", "if", "9827", "+", "==", "{" en ")". Lay-out elementen (spaties, tabs, nieuwe regels) zijn geen tokens.
In een postfixexpressie komen vijf soorten tokens voor: vier verschillende operatoren, plus getallen. Het enige toegestane layout-element is de spatie.
Een instantie van Token heeft een type en een inhoud. Het type geeft aan wat voor soort token het is, en inhoud geeft de bijbehorende string. Er is een extra type ONBEKEND voor symbolen die niet in de expressie thuishoren.
Zie voor de details van de interface bijlage 1.

– Een klasse TokenLezer. Een instantie van TokenLezer krijgt bij creatie een string mee die bestaat uit spaties, getallen en operatoren (+, -, *, /). Een aanroep naar volgendeToken() levert het volgende token, of null als er geen volgende token meer is.
TokenLezer maakt gebruik van de Java-klasse StreamTokenizer, maar is wat eenvoudiger in het gebruik.
Zie voor de details van de interface bijlage 2.

– Een onvolledige klasse ExpressieFrame. De event-handler voor de berekenKnop moet nog worden ingevuld.

Aanwijzingen:
– U mag gebruik maken van de klasse java.util.stack. Denk er wel aan dat op een dergelijke stack alleen objecten gelegd kunnen worden, en dus geen ints. U mag ook een eigen stack gebruiken, maar dan moet u ook echt een klasse Stack schrijven met de vereiste interface (push, pop, top, isEmpty en size).

– Schrijf eerst een versie die legale postfixexpressie kan uitrekenen, en test deze met behulp van de expressies uit de tabel.
– Voeg foutafhandeling toe. Denk er over na wat er allemaal fout kan zijn!

Inleveren

Alle java-klassen die tot uw uitwerking behoren.

BIJLAGE 1: De klasse Token

public class **Token**
extends java.lang.Object

Een token is een 'woord' uit een computerprogramma. Voorbeelden van dergelijke woorden uit de taal Java zijn "public", "if", "9827", "+", "==", "{", ")". Een instantie van Token heeft een type en een inhoud. Deze klasse kent zes typen, één voor Getal, één voor elk van de operetoren, en één voor alle andere symbolen (type is onbekend, voor het geval er een onbekend symbool in de expressie staat).

Field Summary

static int	**DELING** Constante voor operator /
static int	**GETAL** Constante voor een getal
static int	**MAAL** Constante voor operator *
static int	**MIN** Constante voor operator -
static int	**ONBEKEND** Constante voor een gelezen symbool dat geen getal, operator of spatie is.
static int	**PLUS** Constante voor operator +

Constructor Summary

Token(int type, java.lang.String inhoud) Maak een nieuw Token

Method Summary

java.lang.String	**getInhoud**() Geeft de inhoud van het token als String, bv. “87” als type = GETAL; “+” als type is PLUS.
int	**getType**() Geeft het type van het token

BIJLAGE 2: De klasse TokenLezer

---

public class **TokenLezer**
extends java.lang.Object

Deze klasse splitst een string bestaande uit de operatoren +, -, *, /, positieve gehele getallen en spaties in tokens.

Constructor Summary

TokenLezer(java.lang.String expressie) Maak een nieuwe tokenlezer

Method Summary

Token	**volgendeToken**() Leest het volgende token uit de expressie en geeft dit terug.