Введение в программирование на Java
Домашнее задание 1

18 февраля 2025

Описание задания

В рамках данного домашнего задания требуется реализовать интерактивный калькулятор для вычисления математических выражений в постфиксной записи (a.k.a. обратной польской нотации).

Постфиксная запись

Математические выражения традиционно записываются в **инфиксной записи**. При такой записи оператор ставится между его операндами.

Например:

$$3 \; - \; 2$$
$$2 \; + \; 3 \; * \; 4$$
$$2 \; * \; 3 \; + \; 4$$
$$2 \; * \; (3 \; + \; 4)$$

В **постфиксной записи** операторы ставятся **после** операндов.

Например (соответствуют примерам выше):

$$3 \; 2 \; -$$
$$2 \; 3 \; 4 \; * \; +$$
$$2 \; 3 \; * \; 4 \; +$$
$$2 \; 3 \; 4 \; + \; *$$

1. В первом примере $-$ применяется к 3 и 2 (именно в таком порядке).
2. Во втором примере $*$ относится к 3 и 4, а $+$ относится к 2 и результату умножения.
3. В третьем примере $*$ относится к 2 и 3, а $+$ относится к результату умножения и 4.
4. В четвёртом примере $+$ относится к 3 и 4, а $*$ относится к 2 и результату сложения.

Особенность данной записи в том, что:

- В такой записи не нужны скобки — приоритет операций определяется порядком записи операций (как можно видеть в четвёртом примере выше).
- **Что важно для данного задания**: Для вычисления выражения существует простой алгоритм, использующий абстрактный тип данных **”стек”** (см. Википедия).

Требования к реализации

Программа должна представлять собой интерактивную оболочку (так называемый REPL — read-eval-print-loop).

Выполнение программы должно состоять из следующих шагов:

1. Запрос выражения от пользователя.
2. Чтение введённого выражения в постфиксной записи.
3. Проверка выражения на ошибки. Если выражение содержит ошибку — вывод соответствующего сообщения и возврат к шагу №1.
4. Вычисление выражения и вывод результата.
5. Переход к шагу №1.

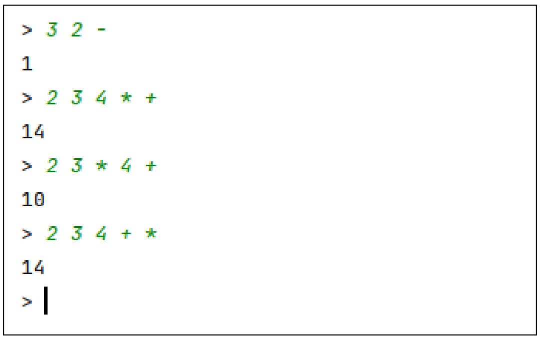
Таким образом, программа представляет собой бесконечный цикл из вышеописанных действий.

Рекомендуется предусмотреть дополнительную команду, позволяющую завершить программу (например, ввод строки `quit` или пустой строки).

Выражения

Программа должна поддерживать вычисления выражений в постфиксной записи над неотрицательными целыми числами с операциями "сложение", "вычитание", "умножение", "целочисленное деление" (`+`, `-`, `*` и `/` соответственно).

Пример работы программы (зелёное — пользовательский ввод, чёрное — вывод программы):



```
> 3 2 -
1
> 2 3 4 * +
14
> 2 3 * 4 +
10
> 2 3 4 + *
14
>
```

Переменные

Помимо этого, в программе должна быть возможность использовать переменные — присваивать им значения и использовать в последующих выражениях.

Должны поддерживаться всего 5 следующих переменных: `x1`, `x2`, `x3`, `x4`, `x5`.

Изначально значения всех переменных равны нулю.

Чтобы присвоить значение переменной, пользователь должен ввести команду следующего вида:

```
<имя_переменной> = <выражение>
```

К примеру:

```
x1 = 2 3 +
```

Операция присваивания тоже является выражением, поэтому при его обработке должно быть напечатано значение, которое было присвоено в переменную[1].

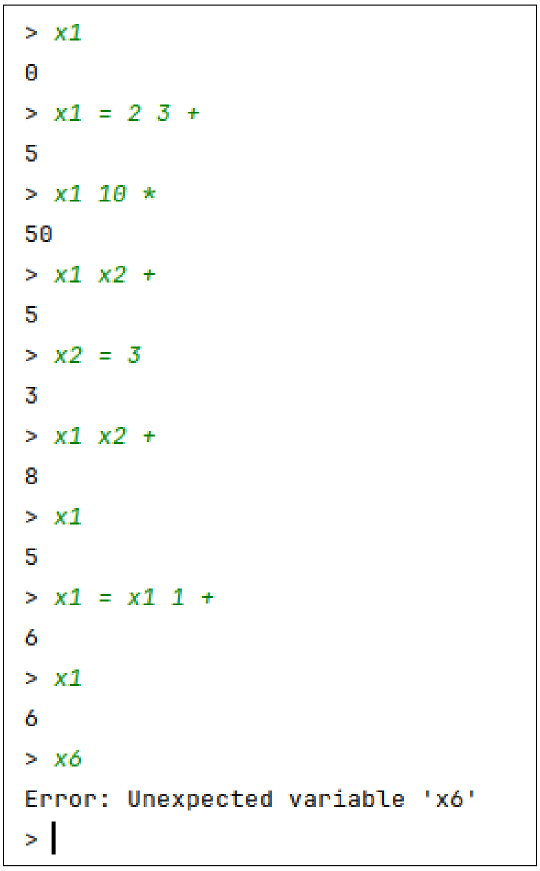
Для использования переменной в выражении необходимо написать её имя в качестве операнда. Например:

```
x1 10 *
```

В случае, если переменная одновременно используется и слева, и справа от присваивания, то в выражении используется прежнее значение переменной, а только затем в переменную присваивается новое значение. Например, увеличение переменной на единицу можно написать следующим образом:

```
x1 = x1 1 +
```

Пример работы программы (зелёное — пользовательский ввод, чёрное — вывод программы):



```
> x1
0
> x1 = 2 3 +
5
> x1 10 *
50
> x1 x2 +
5
> x2 = 3
3
> x1 x2 +
8
> x1
5
> x1 = x1 1 +
6
> x1
6
> x6
Error: Unexpected variable 'x6'
> 
```

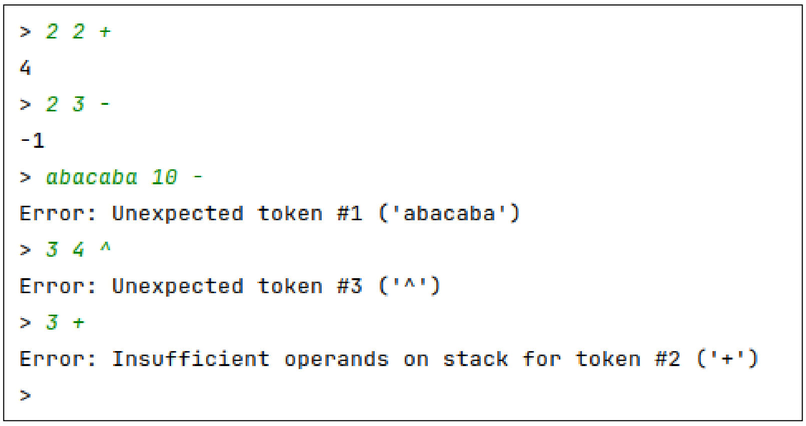
---

[1] Цепочку присваиваний типа `x1 = x2 = x3 = 2 3 +` можно обрабатывать на ваше усмотрение. Но это не является обязательным, поэтому можно считать такое выражение ошибкой.

Обработка некорректных входных данных

- Разрешается не обрабатывать: переполнение значения выражения при выполнении операций (можно ограничиться типом `int` при вычислениях).
- Разрешается не обрабатывать: ввод слишком большого числа для типа `int`.
    - При вводе строки, состоящей из цифр, но имеющей слишком большое значение для типа `int` (например, 1 000 000 000 000) программе разрешается завершаться с ошибкой без корректной её обработки.
    - При вводе чисел до $2^{31} - 1$ (т.е. помещающихся в тип `int`) программа должна работать корректно.
    - При вводе в качестве числа нечисловой строки (например, `Abacaba`) программа должна сообщать об ошибке, но продолжать корректно работать.
- Все прочие ошибки, связанные с некорректными входными данными (например, ввод несуществующего оператора), должны быть корректно обработаны. При вводе некорректных данных программа должна сообщать об ошибке, но продолжать работать.

Пример работы программы (зелёное — пользовательский ввод, чёрное — вывод программы):



```
> 2 2 +
4
> 2 3 -
-1
> abacaba 10 -
Error: Unexpected token #1 ('abacaba')
> 3 4 ^
Error: Unexpected token #3 ('^')
> 3 +
Error: Insufficient operands on stack for token #2 ('+')
>
```

Примечание 1: на скриншоте приведён один из возможных вариантов вывода. Сообщать об ошибках можно любым другим текстом на ваше усмотрение (главное — **понятным образом**).

Примечание 2: на скриншоте приведены не все возможные ситуации ошибочных данных. Найти и правильно обработать такие входные данные: одна из задач данного домашнего задания.

Правила кодирования

Программа должна быть реализована в одном классе `Main` в процедурном стиле.

Исходный код программы должен быть совместим с Java 21 (при разработке рекомендуется использовать OpenJDK 21).

Код **не должен** быть написан в одном единственном методе `main`. Программа должна быть декомпозирована на небольшие методы с понятными именами, каждый из которых решает свою небольшую задачу.

Разбиение кода на несколько классов допускается, но только если это сделано **грамотно**.

Прочие правила кодирования, в соответствии с которыми должна быть оформлена программа, можно найти в соответствующем документе.

Критерии оценивания

Программа, реализующая весь описанный функционал (все операции, поддержку переменных и т.д.) и корректно обрабатывающая ошибочные ситуации получает оценку 8.

Снижение оценки:

- При неполной реализации требуемого функционала, оценка может быть снижена вплоть до 0 баллов.
  - В частности, при отсутствии поддержки переменных оценка снижается на 2 балла (при неполной поддержке — оценка снижается от 0.1 до 2 баллов).
- При отсутствии обработки ошибочных ситуации оценка снижается на не более чем 3 балла (в зависимости от критичности и количества таких ситуаций).
- За нарушение правил кодирования оценка снижается на не более чем 3 балла (в зависимости от регулярности и серьёзности).
- Если код программы не компилируется, выставляется оценка 0 (допускаются исключения на усмотрение проверяющего, если ошибка легко исправляется, а программа в целом корректно реализована).
- При обнаружении несамостоятельного выполнения задания все участники получают оценку 0 (в том числе и настоящий автор программы).

Дополнительные баллы:

- При **грамотном** использовании знаний, которые не проходили на лекциях и семинарах, оценка может быть повышена
  - Как пример: использование перечислений (enum).
- **Корректная** реализация дополнительного функционала:
  - Поддержка чисел неограниченного размера (до +2 баллов).
  - Поддержка переменных с любым именем (до +2 баллов).
  - Поддержка выражений в инфиксной записи со скобками (до +4 баллов). **Важно**: должна быть возможность запустить программу в исходном режиме (т.е. с постфиксными выражениями). Отсутствие такой возможности равносильно невыполнению задания. Другими словами, выражения в инфиксной записи должны транслироваться в постфиксную запись и выполнять исходную программу.
  - Прочий дополнительный функционал (при условии одобрения со стороны лектора).

При реализации дополнительного функционала требуется создать текстовый файл README, в котором описать реализованный функционал. Файл необходимо приложить в архив при сдаче задания.

Формат сдачи задания

Задание должно быть загружено в **Smart LMS**.

Дедлайн: 5 марта 2025, 23:59.

В качестве решения должен быть приложен zip-архив, содержащий проект в IntelliJ IDEA с решением, в том числе:

- Директорию .idea и файл *.iml, находящиеся в корне проекта.
- Директорию src с исходными файлами *.java.

Прочих "мусорных" файлов в архиве быть **не должно**.

Архив должен иметь имя `HW1_<ГРУППА>_<ФИО>.zip`, например `HW1_ББИ2401_ИвановИванИванович.zip`.

Нарушение правил именования архива, наличие в нём лишних файлов или отсутствие необходимых влечёт за собой **снижение оценки**.

Пример корректного архива:

- `HW1_ББИ2401_ИвановИванИванович.zip`
  - `.idea`
    - `workspace.xml`
    - `misc.xml`
    - `modules.xml`
    - `...`
  - `Calculator.iml`
  - `src`
    - `Main.java`