Odwrotna notacja polska w języku Java

Źródło: Guillaume Techer, domena publiczna.

Część języków programowania (a także kalkulatorów) używa do swoich obliczeń odwrotnej notacji polskiej. Wiesz już, na czym ona polega i czym różni się od tradycyjnego zapisu wyrażeń arytmetycznych.

W tym e-materiale zaimplementujesz algorytm konwersji z notacji infiksowej do ONP w języku Java.

Jeśli ciekawi cię, jak wyglądają implementacje w innych językach programowania, możesz się z nimi zapoznać w dwóch pozostałych e-materiałach z tej serii:

- Odwrotna notacja polska w języku C++,
- Odwrotna notacja polska w języku Python.

Więcej zadań? Sięgnij do: Odwrotna notacja polska – zadania maturalne.

> **Twoje cele**
>
> - Wykorzystasz w praktyce sposób zapisywania wyrażeń arytmetycznych z zastosowaniem odwrotnej notacji polskiej.
> - Zaimplementujesz w języku Java algorytm przekształcania wyrażenia arytmetycznego zapisanego z wykorzystaniem notacji infiksowej do postaci w ONP.
> - Rozwiążesz przykładowe zadania wymagające wykazania się znajomością ONP.

Przeczytaj

Polecenie 1

Przeanalizuj prezentację przedstawiającą kolejne kroki implementacji rekurencyjnego algorytmu konwertującego wyrażenie arytmetyczne zapisane za pomocą notacji infiksowej na postfiksową. Następnie zastanów się, dlaczego program działa jedynie w sytuacji, gdy wyrażenie arytmetyczne zawiera liczby naturalne.

Specyfikacja problemu:

Dane:

- `klasyczna` – łańcuch znaków przechowujący wyrażenie arytmetyczne zapisane w notacji infiksowej, pozbawione spacji, gdzie każde z działań wydzielone jest parą nawiasów

Wynik:

- wyrażenie arytmetyczne zamienione na notację ONP

1

Zaimplementujemy w języku Java algorytm zamiany wyrażenia arytmetycznego zapisanego w postaci infiksowej na ONP. Pamiętaj, że sprawdza się on jedynie w przypadku wyrażeń arytmetycznych zawierających liczby naturalne.

2

Zaczniemy od deklaracji metody, która będzie zmieniać zapis wyrażenia arytmetycznego. Przyjmie ona dwa parametry: ciąg znaków zawierający analizowane wyrażenie arytmetyczne (`klasyczna`) oraz indeks (`i`)

odpowiadający pozycji aktualnie badanego elementu w podanym ciągu znaków.

```
1 public static int
  ONP(String klasyczna, int
  i) {
2
3 }
```

3

Następnie zapiszemy instrukcję warunkową `if`. Będzie ona odpowiadać za wykonanie tej części kodu, która powinna zostać wywołana w sytuacji, gdy badany znak (ten o indeksie `i`) będzie różny od nawiasu otwierającego „(". Jest to przypadek bazowy. Po natrafieniu na niego przerywamy rekurencję.

```
1 public static int
  ONP(String klasyczna, int
  i) {
2     if klasyczna[i] !=
  '(':
3
4     } else {
5
6     }
7
8 }
```

4

Jeżeli znak o indeksie `i` nie jest znakiem „(", oznacza to, że jest on liczbą i należy ją wypisać, a następnie zwrócić wartość indeksu. Ponieważ liczba może składać się z więcej niż jednej cyfry, do wypisania każdej z nich użyjemy pętli `while`.

```

```

```
public static int
ONP(String klasyczna, int
i) {
    if klasyczna[i] !=
'(':
        while
(klasyczna.charAt(i) >=
'0' && klasyczna.charAt(i)
<= '9') {

    } else {

    }

}
```

5

Dopóki kolejny znak jest cyfrą, wypisujemy go w konsoli i zwiększamy wartość zmiennej `i` o 1.

```
public static int
ONP(String klasyczna, int
i) {
    if klasyczna[i] !=
'(':
        while
(klasyczna.charAt(i) >=
'0' && klasyczna.charAt(i)
<= '9') {

System.out.print(klasyczna
.charAt(i));
            i += 1;
    } else {

    }

}
```

6

Po zakończeniu działania zmienna `i` będzie wskazywać na znak za wypisaną liczbą. Dlatego odejmujemy od niej 1, aby wskazywała ostatni element operandu.

```
1  public static int
   ONP(String klasyczna, int
   i) {
2      if klasyczna[i] !=
   '(':
3          while
   (klasyczna.charAt(i) >=
   '0' && klasyczna.charAt(i)
   <= '9') {
4
   System.out.print(klasyczna
   .charAt(i));
5              i += 1;
6          System.out.print("
   ");
7          return i - 1;
8      } else {
9
10     }
11
12 }
```

7

Przejdźmy do uzupełnienia kodu, który powinien się wykonać, gdy wskazywany przez zmienną `i` znak w napisie `klasyczna` jest nawiasem otwierającym. Wywołujemy metodę ONP z indeksem powiększonym o 1. Będzie wykonywana tak długo, aż lewy operand tej operacji zostanie przeanalizowany.

```
1  public static int
   ONP(String klasyczna, int
   i) {
```

```
    if klasyczna[i] !=
'(':
        while
(klasyczna.charAt(i) >=
'0' && klasyczna.charAt(i)
<= '9') {
System.out.print(klasyczna
.charAt(i));
            i += 1;
        System.out.print("
");
        return i - 1;
    } else {
        i = ONP(klasyczna,
i + 1);

    }

}
```

8

Wiedząc, że indeks `i` wskazuje pozycję ostatniego elementu pierwszego operandu wyrażenia arytmetycznego, możemy wywnioskować, że kolejny znak ciągu to operator arytmetyczny, który w odwrotnej notacji polskiej zapisywany jest na końcu wyrażenia arytmetycznego. Zamierzamy zapisać jego wartość do zmiennej `operator`. Zwiększymy więc wartość indeksu `i` o 1, a następnie pobierzemy znak operatora do zmiennej.

```
public static int
ONP(String klasyczna, int
i) {
    if klasyczna[i] !=
'(':
        while
(klasyczna.charAt(i) >=
```

```
'0' && klasyczna.charAt(i)
<= '9') {
4
System.out.print(klasyczna
.charAt(i));
5               i += 1;
6           System.out.print("
");
7           return i - 1;
8       } else {
9           i = ONP(klasyczna,
i + 1);
10          i += 1;
11          char operator =
klasyczna.charAt(i);
12
13      }
14
15  }
```


Konwersja wyrażenia arytmetycznego obliczana jest dla dwóch operandów umieszczonych między nawiasami. W kodzie zawarliśmy już analizę pierwszego z nich – lewego. Teraz czas zająć się drugim, czyli prawym.

```
1 public static int
ONP(String klasyczna, int
i) {
2     if klasyczna[i] !=
'(':
3         while
(klasyczna.charAt(i) >=
'0' && klasyczna.charAt(i)
<= '9') {
4
System.out.print(klasyczna
.charAt(i));
5               i += 1;
6           System.out.print("
");
```

```
7           return i - 1;
8       } else {
9           i = ONP(klasyczna,
i + 1);
10          i += 1;
11          char operator =
klasyczna.charAt(i);
12                  i =
ONP(klasyczna, i + 1);
13
14      }
15
16  }
```

10

Teraz, gdy przeanalizowaliśmy już oba operandy, zwiększymy indeks `i` o 1, ponieważ następny znak będzie prawym nawiasem „)", który nie wnosi nic do rozwiązania. Możemy go pominąć. Dodatkowo, ponieważ przeanalizowaliśmy już oba operandy, możemy wypisać na standardowe wyjście wartość zmiennej `operator`, a następnie zwrócić wartość zmiennej `i`. Posłuży ona kolejnemu wywołaniu do kontynuowania analizy dalszych części wyrażenia arytmetycznego.

```
1  public static int
   ONP(String klasyczna, int
   i) {
2      if klasyczna[i] !=
   '(':
3          while
   (klasyczna.charAt(i) >=
   '0' && klasyczna.charAt(i)
   <= '9') {
4
   System.out.print(klasyczna
   .charAt(i));
5              i += 1;
```

```
        System.out.print("
");
        return i - 1;
    } else {
        i = ONP(klasyczna,
i + 1);
        i += 1;
        char operator =
klasyczna.charAt(i);
                i =
ONP(klasyczna, i + 1);
        i += 1;

System.out.print(operator
+ " ");
                return i;

    }

}
```

Dodajmy do programu przykładowe wywołanie.

11

```
public class Main {
    public static void
main(String[] args) {
        String klasyczna =
"((8*9)*(3+6))";
        int i = 0;
        i = ONP(klasyczna,
i);
    }

    public static int
ONP(String klasyczna, int
i) {
        if
(klasyczna.charAt(i) !=
'(') {
            while
(klasyczna.charAt(i) >=
```

```
'0' && klasyczna.charAt(i)
<= '9') {
11
 System.out.print(klasyczn
a.charAt(i));
12                  i += 1;
13              }
14
 System.out.print(" ");
15              return i - 1;
16          } else {
17              i =
ONP(klasyczna, i + 1);
18              i += 1;
19              char operator
= klasyczna.charAt(i);
20              i =
ONP(klasyczna, i + 1);
21              i += 1;
22
 System.out.print(operator
+ " ");
23              return i;
24          }
25      }
26 }
```

Tym samym udało nam się zaimplementować algorytm zamiany wyrażenia w notacji infiksowej na wyrażenie w ONP.

W przykładzie dane jest wyrażenie zapisane w notacji infiksowej: `((8*9)*(3+6))`

Zamieniliśmy je na wyrażenie w notacji postfiksowej, które ma następującą postać: `8 9 * 3 6 + *`

Problem 1

Napisz program, który przekształci wyrażenie arytmetyczne zapisane w notacji infiksowej (zakładamy, że wyrażenie zawiera pełne nawiasowanie, występują w nim wyłącznie liczby naturalne oraz operatory jednoznakowe) do postaci ONP. Przetestuj działanie programu dla wyrażenia arytmetycznego `(((27/5)-(5*23))+((26-4)-(23^24)))`.

Specyfikacja problemu:

Dane:

- `notacjaKlasyczna` – wyrażenie arytmetyczne w notacji infiksowej z pełnym nawiasowaniem; łańcuch znaków

Wynik:

- wyrażenie arytmetyczne w ONP

> **Ważne!**
>
> W wyrażeniu zastosowaliśmy operator ^. Wykorzystujemy go jako operator potęgowania.

Twoje zadania

1. Program zamienia podane wyrażenie arytmetyczne z notacji klasycznej na ONP.

```
1 public class Main {
2     public static void main(String[] args){
3         String notacjaKlasyczna = "(((27/5)-(5*23))+((26-4)-
  (23^24)))";
4 
5         // Miejsce na wpisanie kodu
6     }
7 }
```

```

```

```
1
```

Słownik

operand

liczba, na której wykonywane jest działanie

> **Przykład 1**
>
> Dla wyrażenia arytmetycznego:
>
> ```
> 27 * 5
> ```
>
> liczby 27 oraz 5 to **operandy**, a znak mnożenia jest **operatorem**

pełne nawiasowanie

szczególne ustawienie nawiasów w wyrażeniu arytmetycznym w notacji infiksowej; dzięki niemu każde działanie ma jednoznacznie określone dwa operandy za pomocą ujęcia w nawiasy, a także całe działanie jest objęte nawiasem, np. `(a+(b·c))`

stos

jedna ze struktur danych w informatyce, której działanie opiera się na metodzie LIFO (ang. *Last In First Out*), co znaczy, że ten element, który został dodany do stosu jako ostatni, jako pierwszy zostanie z niego pobrany; dane są dokładane na wierzch stosu i z wierzchołka stosu są pobi erane

zapis infiksowy

inaczej: notacja infiksowa, notacja klasyczna; sposób zapisu wyrażeń, w którym operator jest stawiany między operandami, na których chcemy wykonać daną operację; w odróżnieniu do ONP stosuje się nawiasy

zapis postfiksowy

inaczej: odwrotna notacja polska (ONP), notacja postfiksowa; sposób zapisu wyrażeń, w którym operator jest stawiany po operandach, na których chcemy wykonać daną operację; w tej notacji nie stosuje się nawiasów

zapis prefiksowy

inaczej: notacja prefiksowa; sposób zapisu wyrażeń, w którym operator jest stawiany przed operandami, na których chcemy wykonać daną operację; w tej notacji nie stosuje się nawiasów

Symulacja interaktywna

Polecenie 1

Uruchom symulację i przekształć wyrażenie arytmetyczne zapisane w notacji **infiksowej** na **postfiksową**. Informacje teoretyczne dotyczące tego zagadnienia znajdziesz w e-materiale Odwrotna notacja polska.

Uwaga, w symulacji nie stosujemy pełnego nawiasowania.

Symulacja 1

W poniższej symulacji interaktywnej twoim zadaniem jest samodzielne przekształcenie wyrażenia arytmetycznego zapisanego w notacji **infiksowej** na notację **postfiksową.**

Czasem by przejść dalej, konieczne będzie dokonanie wyboru z podanych opcji. W przypadku slajdów wyjaśniających zawsze możesz przejść do kolejnego slajdu (opcja `Przejdź do kolejnego slajdu`) lub wrócić do poprzedniego slajdu (opcja `Wróć do poprzedniego slajdu`), natomiast w slajdach, gdzie konieczny jest wybór kolejnego kroku rozwiązania problemu, powrót do wcześniejszego slajdu jest możliwy, natomiast przejście do kolejnego już nie.

Od trzeciego slajdu możesz również zacząć od nowa i wrócić do pierwszego slajdu (opcja `Wróć do pierwszego slajdu i zacznij od nowa`).

> **Ważne!**
>
> W wyrażeniu zastosowaliśmy operator ^. Wykorzystujemy go jako operator potęgowania.

> **Przykład 1**
>
> **W jaki sposób czytać informacje w symulacji?**
>
> 1. Wyrażenie i pobierany element: `27 * 5 + 23 - (26 / (24 ^ 4))`
> 2. Stos: `+`
> 3. Przekształcone wyrażenie: `27 5 * 23`
> 4. Co należy zrobić z analizowanym elementem ?
> 5. Umieść element na stosie.
>   Stos: + -
>   Wyjście: 27 5 * 23
>
>   Dodaj element do wyniku.
>   Stos: +
>   Wyjście: 27 5 * 23 -
> 6. Wróć do poprzedniego slajdu
>
>   Wróć do pierwszego slajdu i zacznij od nowa

1

Zwróć uwagę na to, że element 23 jest wyróżniony. To on jest analizowany.

2

Ten element mówi, co w danym momencie znajduje się na stosie.

3

Tak wygląda dotychczas przekształcone wyrażenie.

4

Polecenia, które należy wykonać.

5

Pierwsze wiersze obu opcji wskazują kroki, które mają zostać wykonane, natomiast kolejne pokazują, jak powinny wyglądać stos oraz aktualny wynik po wykonaniu danego kroku.

6

Kolejne opcje pozwalają na nawigację po symulacji.

Symulacja

Polecenie 2

Na podstawie symulacji interaktywnej podsumuj w notatce najważniejsze informacje dotyczące konwersji wyrażeń arytmetycznych zapisanych za pomocą notacji infiksowej do postaci postfiksowej.

Polecenie 3

Zapisz przykład wyrażenia arytmetycznego w notacji infiksowej. Następnie przekaż go koleżance lub koledze, a od innej osoby weź zadanie dla siebie. Przeprowadź analizę otrzymanego wyrażenia, przekształcając je do postaci postfiksowej – w razie potrzeby skorzystaj z symulacji.

Polecenie 4

W sekcji „Przeczytaj” przedstawiono program, który przekształca wyrażenia arytmetyczne zapisane w notacji infiksowej na postfiksową.

Zmodyfikuj go tak, by przekształcał wyrażenie arytmetyczne zapisane w notacji postfiksowej na postać infiksową.

Uruchom swój program w środowisku lokalnym i przetestuj dla następującego wyrażenia arytmetycznego zapisanego w postaci postfiksowej:

```
8 9 * 3 6 + *
```

Zastosuj pełne nawiasowanie.

Poprawny wynik dla powyższych danych:

```
((8*9)*(3+6))
```

```
// Poniżej znajdziesz program z sekcji "Przeczytaj", który nale

public class Main {
    public static void main(String[] args) {
        String klasyczna = "((8*9)*(3+6))";
        int i = 0;
        i = ONP(klasyczna, i);
    }

    public static int ONP(String klasyczna, int i) {
        if (klasyczna.charAt(i) != '(') {
            while (klasyczna.charAt(i) >= '0' && klasyczna.char
                System.out.print(klasyczna.charAt(i));
                i += 1;
            }
            System.out.print(" ");
            return i - 1;
        } else {
            i = ONP(klasyczna, i + 1);
```

```
            i += 1;
            char operator = klasyczna.charAt(i);
            i = ONP(klasyczna, i + 1);
            i += 1;
            System.out.print(operator + " ");
            return i;
        }
    }
}
```

Twoje zadania

1. Program przekształca wyrażenie arytmetyczne zapisane w notacji postfiksowej na postać infiksową.

```
// Miejsce na wpisanie kodu
```

```
```

Sprawdź się

Pokaż ćwiczenia: 

Ćwiczenie 1

Napisz program, który wypisze wszystkie operatory arytmetyczne występujące w przekazanym wyrażeniu arytmetycznym zapisanym w notacji infiksowej w postaci łańcucha znaków, bez spacji między nimi. Przetestuj swój program dla następujących danych:

```
dzialanie = "(3/4*9)-4+8/9"
```

Program powinien działać dla operatorów odejmowania, dodawania, mnożenia oraz dzielenia.

Specyfikacja problemu:

Dane:

- `dzialanie` – wyrażenie arytmetyczne zapisane w postaci infiksowej; łańcuch znaków

Wynik:

- operatory w kolejności ich występowania w wyrażeniu arytmetycznym `dzialanie`; każdy kolejno w nowej linii

Twoje zadania

1. Program wypisuje operatory w kolejności występowania.

```
public class Main {
    public static void main(String [] args) {
        String działanie = "(3/4*9)-4+8/9";

        // Miejsce na wpisanie kodu

    }
}
```

```
1
```

Ćwiczenie 2

Napisz program, który wypisze cyfry z podanego ciągu znaków aż do napotkania znaku, który nie jest cyfrą. Rozwiązanie przetestuj dla ciągu znaków: $25476325b4823675472634$.

Specyfikacja problemu:

Dane:

- `ciagZnakow` – badany ciąg zawierający dowolne znaki; łańcuch znaków

Wynik:

- ciąg cyfr z ciągu `ciagZnakow`, które poprzedzają pierwszy znak, który nie jest cyfrą; każda cyfra w nowej linii

Twoje zadania

1. Program wypisuje kolejne cyfry z ciągu znaków aż do napotkania znaku, który nie jest cyfrą.

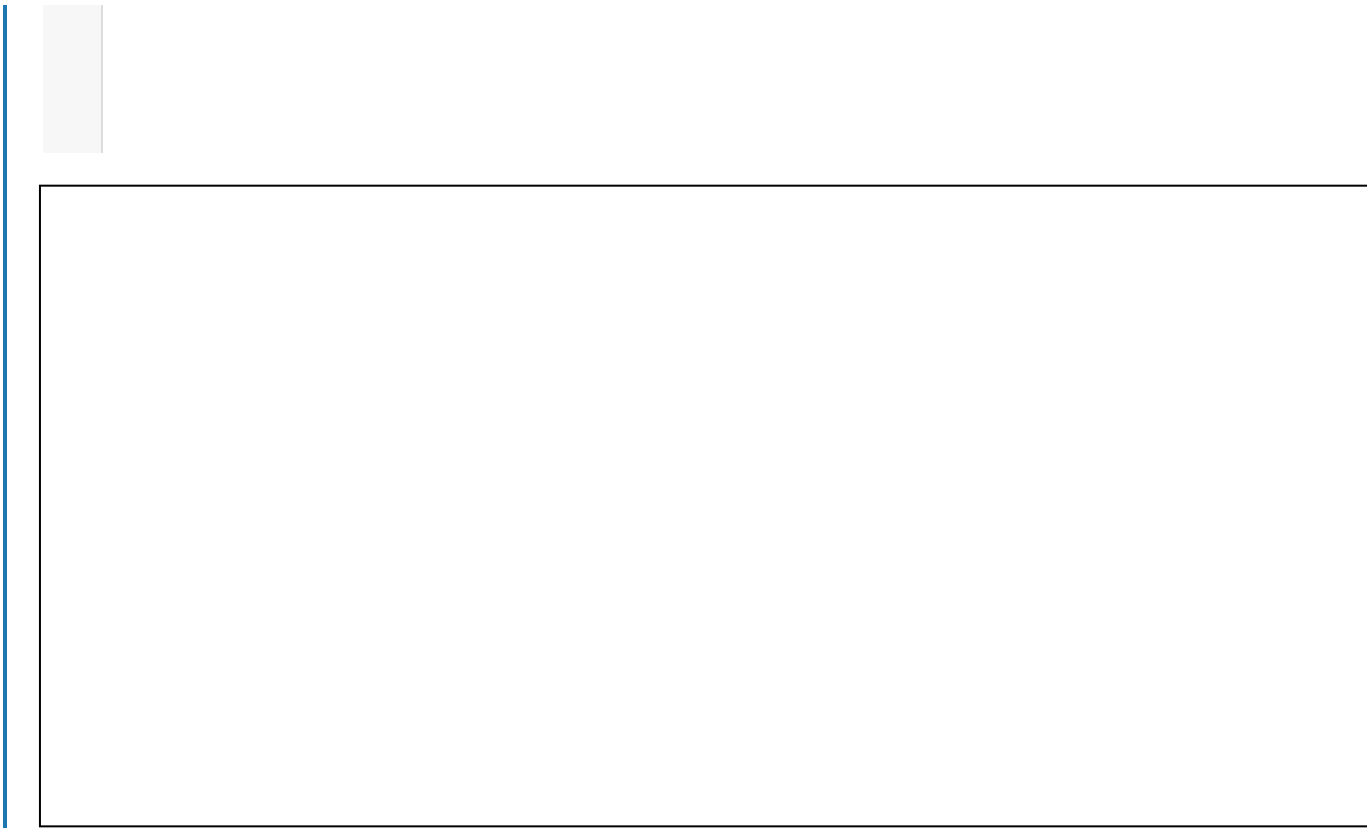
```
public class Main {
    public static void main(String [] args) {
        String ciagZnakow = "25476325b4823675472634";

        // Miejsce na wpisanie kodu

    }
}
```

```

```



Ćwiczenie 3

Napisz program, który zamieni wyrażenie arytmetyczne napisane w odwrotnej notacji polskiej na wyrażenie arytmetyczne zapisane w notacji klasycznej. Do zrealizowania tego algorytmu wykorzystaj stos; w języku Java tę strukturę implementuje klasa **`Stack()`**.

Przetestuj swój program dla następujących danych:

```
notacjaONP = "15 9 * 25 6 + *"
```

Specyfikacja problemu:

Dane:

- `notacjaONP` – wyrażenie arytmetyczne zapisane w notacji postfiksowej; łańcuch znaków

Wynik:

- wyrażenie arytmetyczne zamienione na notację klasyczną z pełnym nawiasowaniem; niepusty ciąg znaków

Twoje zadania

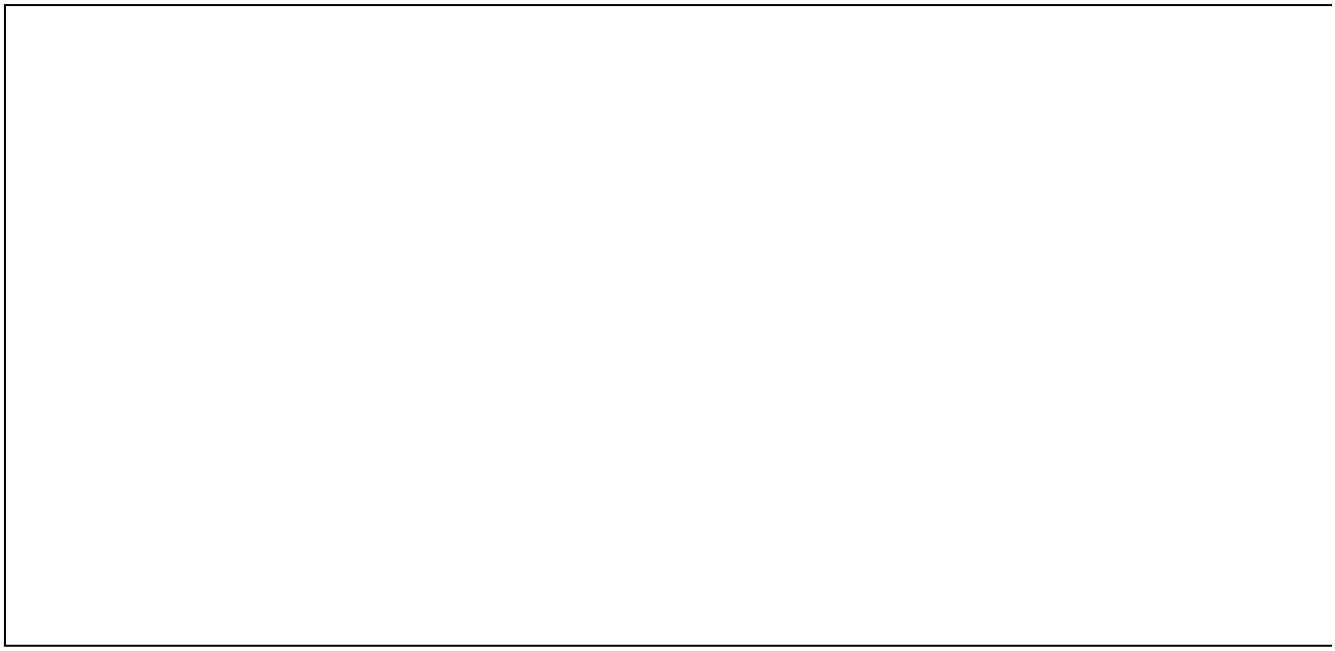
1. Program zamienia notację postfiksową na notację infiksową.

```
import java.util.Stack;
/*
    Użycie stosu:
        - stos.push(element) - włożenie elementu na stos
        - stos.pop() - wyciągnięcie górnej wartości ze stosu
        - stos.peek() - sprawdzenie górnej wartości na stosie bez jej wyciągania
*/

public class Main {
    public static void main(String [] args) {
        String notacjaONP = "15 9 * 25 6 + *";
```

```
12
13          zamiana(notacjaONP);
```

```
1
```



Dla nauczyciela

Autor: Maurycy Gast
Przedmiot: Informatyka

Temat: Odwrotna notacja polska w języku Java

Grupa docelowa:

Liceum ogólnokształcące i technikum, liceum ogólnokształcące, technikum, zakres rozszerzony

Podstawa programowa:

Cele kształcenia – wymagania ogólne

I. Rozumienie, analizowanie i rozwiązywanie problemów na bazie logicznego i abstrakcyjnego myślenia, myślenia algorytmicznego i sposobów reprezentowania informacji.

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera oraz innych urządzeń cyfrowych: układanie i programowanie algorytmów, organizowanie, wyszukiwanie i udostępnianie informacji, posługiwanie się aplikacjami komputerowymi.

Treści nauczania – wymagania szczegółowe

I + II. Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

2) wykorzystuje znane sobie algorytmy przy rozwiązywaniu i programowaniu rozwiązań następujących problemów:

d) zamiany wyrażenia na postać w odwrotnej notacji polskiej i obliczanie jego wartości na podstawie tej postaci,

3) objaśnia, a także porównuje podstawowe metody i techniki algorytmiczne oraz struktury danych, wykorzystując przy tym przykłady problemów i algorytmów, w szczególności:

i) struktury dynamiczne: stos, kolejka, lista (do realizacji algorytmu: ONP, symulacji problemu Flawiusza, sortowania leksykograficznego),

Kształtowane kompetencje kluczowe:

- kompetencje cyfrowe;

- kompetencje osobiste, społeczne i w zakresie umiejętności uczenia się;
- kompetencje matematyczne oraz kompetencje w zakresie nauk przyrodniczych, technologii i inżynierii.

Cele operacyjne (językiem ucznia):

- Wykorzystasz w praktyce sposób zapisywania wyrażeń arytmetycznych z zastosowaniem odwrotnej notacji polskiej.
- Zaimplementujesz w języku Java algorytm przekształcania wyrażenia arytmetycznego zapisanego z wykorzystaniem notacji infiksowej do postaci w ONP.
- Rozwiążesz przykładowe zadania wymagające wykazania się znajomością ONP.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

- dyskusja;
- rozmowa nauczająca z wykorzystaniem multimedium i ćwiczeń interaktywnych;
- ćwiczenia praktyczne.

Formy pracy:

- praca indywidualna;
- praca w parach;
- praca w grupach;
- praca całego zespołu klasowego.

Środki dydaktyczne:

- komputery z głośnikami, słuchawkami i dostępem do internetu;
- zasoby multimedialne zawarte w e-materiale;
- tablica interaktywna/tablica, pisak/kreda;
- oprogramowanie dla języka Java SE 8 (lub nowszej wersji), w tym Eclipse 4.4 (lub nowszej wersji).

Przebieg lekcji

Przed lekcją:

1. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Odwrotna notacja polska w języku Java". Nauczyciel prosi uczniów o zapoznanie się z treściami w sekcji „Przeczytaj".

Faza wstępna:

1. Nauczyciel wyświetla uczniom temat, wskazuje cele zajęć oraz ustala z uczestnikami zajęć kryteria sukcesu.
2. **Rozpoznanie wiedzy uczniów.** Nauczyciel zadaje uczniom pytania dotyczące ich aktualnego stanu wiedzy w obszarze poruszanego tematu i programowania, np.
   – na czym polega notacja infiksowa?
   – jak nazywamy notację, w której operator działania zostaje zapisany przed argumentami?
   – czy w notacji postfiksowej operator działania zostaje zapisany przed czy za argumentami?
   Chętni uczniowie udzielają na nie odpowiedzi.

Faza realizacyjna:

1. **Praca z tekstem.** Nauczyciel ocenia, na podstawie informacji na platformie, stan przygotowania uczniów do zajęć. Jeżeli jest ono niewystarczające prosi o ciche zapoznanie się z treścią w sekcji „Przeczytaj".
2. Nauczyciel czyta polecenie 1 w sekcji „Przeczytaj". Prosi uczniów, aby w parach napisali program, a następnie przeanalizowali rozwiązanie problemu przedstawione w prezentacji. Na forum klasy uczniowie dyskutują o swoich rozwiązaniach.
3. **Praca z multimedium.** Uczniowie indywidualnie zapoznają się z multimedium w sekcji „Symulacja interaktywna". Wykonują w parach polecenie 1.
4. Uczniowie indywidualnie wykonują polecenia 4 oraz 5.
5. **Ćwiczenie umiejętności.** Uczniowie indywidualnie wykonują ćwiczenia 1 oraz 2. Nauczyciel sprawdza poprawność wykonanych zadań, omawiając je wraz z uczniami.

Faza podsumowująca:

1. Nauczyciel wyświetla na tablicy temat lekcji i cele zawarte w sekcji „Wprowadzenie". W kontekście ich realizacji podsumowuje przebieg zajęć, a także wskazuje mocne i słabe strony pracy uczniów.
2. Na koniec zajęć z programowania w Javie nauczyciel prosi uczniów o rozwinięcie zdania: „Na dzisiejszych zajęciach nauczyłam/łem się jak…".

Praca domowa:

1. Uczniowie wykonują ćwiczenie 3 z sekcji „Sprawdź się".
2. Uczniowie rozwiązują problem 1 z sekcji „Przeczytaj".

Materiały pomocnicze:

- Oficjalna dokumentacja techniczna dla języka Java SE 8 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla oprogramowania Eclipse 4.4 (lub nowszej wersji).

Wskazówki metodyczne:

- Multimedium w sekcji „Symulacja interaktywna” można potraktować jako zadanie domowe dotyczące analizy problemu zawartego w temacie „Odwrotna notacja polska w języku Java”.