Java

wprowadzenie

instrukcja iteracyjna - WHILE

Copyright © Szymon Leyk

Czego się dowiesz?

- Czym jest instrukcja WHILE?
- Jak ją stworzyć?
- Jakie jest jej zastosowanie?

Copyright © Szymon Leyk

Czym jest WHILE?

While jest podstawową instrukcją iteracyjną. Umożliwia powtarzanie wykonywania kodu zdefiniowanego wewnątrz tej instrukcji.

Copyright © Szymon Leyk

Definicja

```
while(warunek){
    instrukcje;
}
```

Instrukcje zawarte pomiędzy klamrami, są wykonywane, dopóki warunek jest poprawny (zwraca true).

Copyright © Szymon Leyk

Przykład 1

Program posiada informację o cenach wszystkich produktów w sklepie. Policzmy, jaka jest wartość wszystkich produktów. Ceny: 2.51, 55.19, 9.21, 88.00.

W jaki sposób przechować więcej niż jedną cenę?
double[] prices = {2.51, 55.19, 9.21, 88.00};

Copyright © Szymon Leyk

Przykład 1

W jaki sposób zsumować elementy?

```
double sum = 0;                 // ustawiamy wartość początkową
sum += prices[0];               // dodajemy pierwszy element
sum += prices[1];               // dodajemy drugi element
sum += prices[2];               // dodajemy trzeci element
sum += prices[3];               // dodajemy czwarty element
System.out.println(sum);        // wyświetlenie sumy
```

Copyright © Szymon Leyk

Przykład 1

Wykonaliśmy sumę dla 4 elementów, a co w sytuacji gdyby było ich 4000?
Robiąc ręcznie tego typu operacje, musielibyśmy zapisać 4tys. lini kodu.

Czy znajdzie tutaj zastosowanie pętla? Pętli używamy, gdy coś się powtarza.

Co powtarza się w sumowaniu N liczb?
Powtarza się operacja dodania jednego elementu do aktualnej sumy.

Spróbujmy wrzucić ten kawałek kodu do pętli.

Copyright © Szymon Leyk

Przykład 1

```
double sum = 0;                 // ustawiamy wartość początkową
while(true) {

}
System.out.println(sum);    // wyświetlenie sumy
```

Pętla musi mieć jakiś warunek, na początek ustawmy go jako "nieskończony" (zawsze prawdziwy).

Copyright © Szymon Leyk

Przykład 1

```
double sum = 0;                 // ustawiamy wartość początkową
while(true) {
    sum += prices[0];           // dodajemy pierwszy element
}
System.out.println(sum);    // wyświetlenie sumy
```

Wewnątrz pętli ustawiłem jedną operację dodawania.

Copyright © Szymon Leyk

Przykład 1

```
double sum = 0;                 // ustawiamy wartość początkową
while(true) {
      sum += prices[0];         // dodajemy pierwszy element
}
System.out.println(sum);    // wyświetlenie sumy
```

Co z indeksem 0?
Przecież nie chcemy w nieskończoność dodawać tego samego elementu tablicy.

Jakie indeksy chcemy zsumować? od 0... do 3.

Copyright © Szymon Leyk

Przykład 1

```
double sum = 0;                 // ustawiamy wartość początkową
while(true) {
      int f = 0;
      sum += prices[f];         // dodajemy pierwszy element
}
System.out.println(sum);    // wyświetlenie sumy
```

Skoro indeks się zmienia, to użyjmy zmiennej np. o nazwie **f**.
Jeśli chcę użyć zmiennej, to muszę ją zadeklarować oraz ustawić wartość początkową. Od jakiej wartości chcemy rozpocząć? od **0**... do 3

Copyright © Szymon Leyk

Przykład 1

```
double sum = 0;                 // ustawiamy wartość początkową
while(true) {
        int f = 0;
        sum += prices[f];       // dodajemy pierwszy element
        f++;
}
System.out.println(sum);   // wyświetlenie sumy
```

Jak zwiększać naszą zmienną o 1? Mamy kilka opcji: f = f+1 lub f+=1 lub f++;
Wybieramy którąkolwiek.

Copyright © Szymon Leyk

Przykład 1

Czy nasz program działa poprawnie? Nie... Z dwóch powodów.

1. nie określiliśmy warunku końcowego
2. ciągle zerujemy nasze f, co w konsekwencji nadal dodaje pierwszy element

Ustawienie wartości początkowej powinno nastąpić tylko jeden raz, nie może być zatem w pęli, lecz przed nią.

Copyright © Szymon Leyk

Przykład 1

```
double sum = 0;                 // ustawiamy wartość początkową
int f = 0;
while(true) {
      sum += prices[f];         // dodajemy kolejny element
      f++;
}
System.out.println(sum);    // wyświetlenie sumy
```

Teraz jest już lepiej, bo program zacznie sumować, ale... wyjdzie poza zakres tablicy w momencie gdy f osiągnie wartość 4.

Copyright © Szymon Leyk

Przykład 1

Pozostało zatem ustawienie warunku. Skoro chcemy, aby program sumował wartości tablicy, które są na indeksach od 0 do 3, to należy ustawić warunek f <=3 lub f<4.

Copyright © Szymon Leyk

Przykład 1

```
double sum = 0;                 // ustawiamy wartość początkową
int f = 0;
while(f < 4) {
        sum += prices[f];       // dodajemy kolejny element
        f++;
}
System.out.println(sum);   // wyświetlenie sumy
```

Super! Działa! Ale... można lepiej... :)
Jak zmodyfikować kod, aby dostosował się do rozmiaru tablicy?

Copyright © Szymon Leyk

Przykład 1

```
double sum = 0;                 // ustawiamy wartość początkową
int f = 0;
while(f < prices.length) {
      sum += prices[f];         // dodajemy kolejny element
      f++;
}
System.out.println(sum);    // wyświetlenie sumy
```

Wystarczy uzależnić warunek od długości tablicy. W takim przypadku obojętnie jak długą tablicę podamy, algorytm obliczy sumę jej wartości.

Copyright © Szymon Leyk

Przykład 2

Wyświetlenie wszystkich elementów tablicy.
np. char[] name = {'M','a','r','e','k'};

Gdybyśmy chcieli wyświetlić elementy tej tablicy, to należałoby napisać kod:

```
System.out.println(name[0]);
System.out.println(name[1]);
System.out.println(name[2]);
System.out.println(name[3]);
System.out.println(name[4]);
```

Czy coś się powtarza?

Copyright © Szymon Leyk

Przykład 2

Oczywiście zauważamy, że powtarza się operacja wyświetlenia pojedynczego elementu tablicy, a więc:
System.out.println(names[0]);

Podobnie jak w poprzednim przykładzie jest też jeden element, który się zmienia, jest nim indeks.

Copyright © Szymon Leyk

Przykład 2

Użyjmy pętli.

```
int index = 0;
while(index < names.length){
	System.out.println(names[index]);
	index++;
}
```

Teraz jest ok.

Copyright © Szymon Leyk

Przykład 3

Jak wyświetlić elementy przechowywane w dwuwymiarze?

np. int[][] numbers = {{1,2,3}, {4,5}, {6}};
1 2 3
4 5
6

Copyright © Szymon Leyk

Przykład 3

Najpierw spróbujmy wyświetlić pierwszy wiersz. Wiemy, że tablica dwuwymiarowa to jednowymiarowa tablica, tablic jednowymiarowych.

Skorzystajmy z kodu wyświetlającego tablicę jednowymiarową.

```
int index = 0;
while(index < numbers[0].length){    //numbers[0].length, to liczba kolumn wiersza 0
      System.out.println(numbers[0][index]);          // pierwszy wiersz to index 0
      index++;
}
```

Copyright © Szymon Leyk

Przykład 3

Jak iterować, przechodzić, po kolejnych wierszach? Musimy zastanowić się, ile wierszy mamy do dyspozycji.

```
int rowIndex = 0;
while(rowIndex < numbers.length){          // numbers.length zwraca liczbę wierszy
    int colIndex = 0;                      // kolumna w każdym wierszu rozpoczyna się od 0
    while(rowIndex < numbers[rowIndex].length){
        System.out.println(numbers[rowIndex][colIndex]);
        colIndex++;
    }
    rowIndex++;
}
```

Copyright © Szymon Leyk

Zadanie 1

Napisz program, który dla dowolnej, naturalnej liczby n wyświetli tabliczkę mnożenia n x n.

Copyright © Szymon Leyk

Zadanie 2*

Napisz program, który spośród liczb z zakresu od 427 do 529, wypisze te, które są podzielne przez 3, przez 5 oraz podzielne zarówno przez 3 jak i przez 5

Założenia:

- przedział liczbowy 427 – 529.
- gdy program odnajdzie liczbę podzielną przez 3 wypisz “Tik“,
- gdy program odnajdzie liczbę podzielną przez 5 wypisz “Tak“,
- gdy program odnajdzie liczbę podzielną zarówno przez 3 i 5 wypisz “TikTak”.

* wykonaj samodzielnie

Copyright © Szymon Leyk

Zadanie 3*

Dla podanej liczby n, przy pomocy pętli wyświetl trójkąt złożony z gwiazdek, jak niżej:

dla n = 4

```
*
**
***
****
```

dla n = 6

```
*
**
***
****
*****
******
```

* wykonaj samodzielnie

Copyright © Szymon Leyk

Zadanie 4*

Dla podanej nieparzystej liczby n, przy pomocy pętli wyświetl:

dla n = 5

```
  *
 ***
*****
```

dla n = 7

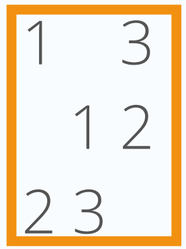
```
   *
  ***
 *****
*******
```

* wykonaj samodzielnie

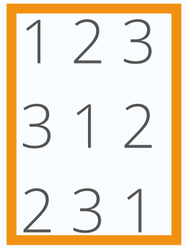
Copyright © Szymon Leyk

Zadanie 5*

Napisz program, który uzupełni brakujące liczby w sudoku.



- wprowadź sudoku do programu (na stałe, oznaczenia wedle uznania)
- program powinien wywnioskować których liczb brakuje, uzupełnić je i wyświetlić uzupełnione sudoku na ekranie

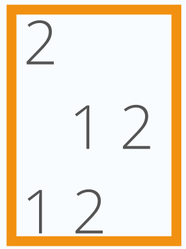


* wykonaj samodzielnie

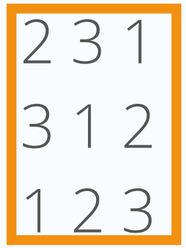
Copyright © Szymon Leyk

Zadanie 6*

Napisz program, który uzupełni brakujące liczby w sudoku.



- wprowadź sudoku do programu (na stałe, oznaczenia wedle uznania)
- program powinien wywnioskować których liczb brakuje, uzupełnić je i wyświetlić uzupełnione sudoku na ekranie



* wykonaj samodzielnie

Copyright © Szymon Leyk