Wyższa Szkoła Bankowa
we Wrocławiu

Programowanie aplikacji w Java

Maciej Gowin

Zjazd 2 - dzień 1

Linki

Opis

https://maciejgowin.github.io/wsb-java/

Kod źródłowy przykładów oraz zadań

https://github.com/MaciejGowin/wsb-programowanie-aplikacji-java

Język Java: pętla for

Przy okazji tablic wspomnieliśmy o pętlach, które pozwalają na iterowanie po elementach tablicy. U swoich podstaw pętle służą do wykonywania danej operacji wielokrotnie.

Wyobraźmy sobie, że chcielibyśmy wypisać na ekran tekst `I like programming` dwadzieścia razy. Zamiast kopiować instrukcję:

```
System.out.println("I like programming");
```

Użylibyśmy czegoś na obraz:

```
EXECUTE 20 times:
    System.out.println("I like programming");
```

Język Java: pętla for

Z pomocą przychodzi pętla `for`, która pozwala na wykonywanie operacji wielokrotnie:

```
for (int i = 0; i < 20; i++) {
    System.out.println("I like programming");
}
```

W naszym przykładzie:

- `int i = 0` jest warunkiem początkowym, od którego rozpoczynamy odliczanie
- `i < 20` jest warunkiem testowym zwracającym `true` lub `false`
  - w przypadku `true` blok kodu jest wykonywany
  - w przypadku `false` wykonywanie pętli jest przerywane
- `i++` jest instrukcją zmieniającą wartość warunku początkowego

Język Java: pętla for

Możemy to uogólnić do:

```
FOR (initial expression; condition; update expression)
    Statement to be executed if condition met
```

Programowanie: przykład 12

Pętla `for` z warunkami oraz różnymi instrukcjami aktualizacyjnymi.

```
public class Main {

    public static void main(String[] args) {
        breakLine();

        int loop1Invocations = 0;
        for (int i = 0; i < 10; i++) {
            loop1Invocations++;
            printIteration("loop1", i);
        }
        printTotal("loop1", loop1Invocations);

        breakLine();

        int loop2Invocations = 0;
        for (int i = 0; i < 10; i = i + 3) {
            loop2Invocations++;
            printIteration("loop2", i);
        }
        printTotal("loop2", loop2Invocations);

        breakLine();
    }

    public static void printIteration(String loopName, int index) {
        System.out.printf("Loop: %s: current index: %d%n", loopName, index);
    }

    public static void printTotal(String loopName, int index) {
        System.out.printf("Loop: %s: invoked: %d times%n", loopName, index);
    }

    public static void breakLine() {
        System.out.println("----------");
    }
}
```

Programowanie: zadanie 10

Zdefiniuj tablicę 5 cen produktów o nazwie `prices` wraz z przypisanymi wartościami. Oblicz sumę cen przed obniżką (`totalPrice`) oraz po obniżce (`totalPriceDiscounted`) zakładając, że każda z cen podlega obniżce równej `10% * indeks ceny w tabeli` . Do obliczeń ceny po obniżce zdefiniuj funkcję `priceAfterDiscount(double price, int discountInPercent)` . Do obliczeń sumy użyj instrukcji `for` . Wynik wypisz na ekran.

Język Java: pętla for

Możliwe jest zdefiniowanie warunku, który będzie zawsze prawdą. Wtedy kod będzie wykonywał się w nieskończoność aż do wyczerpania się pamięci lub przerwania działania przez użytkownika. Podobnie dzieje się w przypadku pominięcia warunku.

```
        for (int i = 0; i >= 0; i++) {
            System.out.println("Run 1");
        }

        for (int i = 0;; i++) {
            System.out.println("Run 2");
        }

        for (;;) {
            System.out.println("Run 3");
        }
```

Język Java: pętla for...each

O pętli `for...each` mówiliśmy przy okazji tablic. Pozwalają one na przegląd wartości w kolekcjach i mają uproszczoną formę w porównaniu do klasycznej pętli `for`.

```
FOR EACH item IN collection
    Statement to be executed
```

Dla przypomnienia:

```
String[] names = {"John", "Peter", "Andrew"};
for (String name: names) {
    System.out.println("Hello " + name);
}
```

Programowanie: przykład 13

Porównanie pętli `for` oraz `for...each`.

```
public class Main {

    public static void main(String[] args) {
        String[] names = {"John", "Peter", "Andrew"};

        for (String name: names) {
            System.out.println("Hello " + name);
        }

        for (int i = 0; i < names.length; i++) {
            System.out.println("Hello " + names[i]);
        }
    }
}
```

Język Java: pętla while

Pętla `while` w założeniu jest podobna do pętli `for`. Pozwala na wielokrotne wykonanie bloku kodu. W tym przypadku jednak kod jest wykonywany, dopóki dany warunek jest spełniony:

```
int = 0;
while (i < 20) {
    System.out.println("I like programming");
    i++;
}
```

W naszym przykładzie:

- `i < 20` jest warunkiem testowym zwracającym `true` lub `false`
  - w przypadku `true` blok kodu jest wykonywany
  - w przypadku `false` wykonywanie pętli jest przerywane

Język Java: pętla while

Możemy to uogólnić do:

```
WHILE (condition)
    Statement to be executed if condition met
```

Język Java: pętla while

Możemy zdefiniować pętlę nieskończoną, podobnie jak w przypadku pętli `for`. Oba poniższe przykłady mają taki sam efekt:

```
for (;;) {
    System.out.println("I like programming");
}

while (true) {
    System.out.println("I like programming");
}
```

Język Java: pętla do...while

Pętla `do...while` działa podobnie do pętli `while` z jedną małą różnicą. W tym przypadku blok kodu jest wykonany po raz pierwszy przed sprawdzeniem warunku:

```
int i = 0;
do {
    System.out.println("I like programming");
    i++;
} while (i < 1);
```

Język Java: pętla do...while

Możemy to uogólnić do:

```
DO
    Statement to be executed if condition met
WHILE (condition)
```

Programowanie: przykład 14

Pobieranie numerów na standardowym wejściu do momentu wprowadzenia wartości oczekiwanej.

```
import java.util.Scanner;

public class Main {

    private static final int MAGIC_NUMBER = 7;

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        int inputInt = 0;
        do {
            System.out.print("Please specify an integer: ");
            inputInt = input.nextInt();
        } while (inputInt != MAGIC_NUMBER);

        System.out.print("Finally!");
    }
}
```

Język Java: break w pętlach

Język Java uzbraja nas w kolejne wyrażenia, które pozwalają na lepsze sterowanie pętlami. Wyrażenie `break` pozwala na natychmiastowe wykonywanie pętli, nawet jeżeli kolejne warunki będą spełnione. Dla przykładu:

```
        for (int i = 0; i < 10; i++) {
            if (i % 9 == 7) {
                break;
            }
            System.out.println("In " + i);
        }
```

Język Java: break w pętlach

Wyrażenie to często jest używane z nieskończoną pętlą `while` pozwalając na jej przerwanie, gdy dany warunek jest spełniony:

```
    while (true) {
        // Some statements

        if (state == DONE) {
            break;
        }

        // Some statements
    }
```

Przy zagnieżdżonych pętlach wyrażenie to przerywa wykonywanie najbardziej wewnętrznej pętli.

Język Java: break w pętlach

Java pozwala na definiowanie `labeled break`, które służą do przerywania wykonywania pętli i przekazania kontroli do wskazanego miejsca. Choć jest to możliwe, odradza się używania tej konstrukcji.

```
topBreak:
while (i < 1000) {
    while (j < 1000) {
        if (i * j == 4004) {
            break topBreak;
        }
    }
}
```

Język Java: continue w pętlach

Podczas przepływu rozkazów w pętlach może wystąpić konieczność pominięcia wykonania bloku dla pewnych iteracji oraz przejście do kolejnego warunku. Używamy do tego wyrażenia `continue` jak w przykładzie:

```
        for (int i = 0; i < 10; i++) {
            if (i % 9 == 7) {
                continue;
            }
            System.out.println("In " + i);
        }
```

Przy zagnieżdżonych pętlach wyrażenie to przerywa wykonywanie iteracji najbardziej wewnętrznej pętli.

Język Java: continue w pętlach

Java pozwala na definiowanie `labeled break`, które służą do przerywania wykonywania pętli i przekazania kontroli do wskazanego miejsca. Choć jest to możliwe, odradza się używania tej konstrukcji.

```
topContinue:
while (i < 1000) {
    while (j < 1000) {
        if (i * j == 4004) {
            continue topContinue;
        }
    }
}
```

Programowanie: zadanie 11

Napisz program, który sczytuje liczny wpisane przez użytkownika do momentu wpisania liczby 0. Zsumuj liczny parzyste i nieparzyste oraz wypisz wynik na ekran.

Język Java: instrukcja warunkowa switch

Znajdź przykład i porównaj z `if...elseif...else`.

```
    public static String getState(int i) {
        String state;
        switch (i) {
            case 0:
                state = "stop";
            case 1:
                state = "low-seed";
            case 2:
                state = "top-speed";
            default:
                state = "unknown";
        }
        return state;
    }
```

Programowanie: przykład 15

Porównanie instrukcji warunkowej `switch` z instrukcją `if...elseif...else`.