Przerywanie pętli w języku Java

Źródło: Mitchell Luo, domena publiczna.

W e-materiale Przerywanie pętli poznaliśmy optymalizujące algorytm instrukcje `break` oraz `continue` . Są to dodatkowe instrukcje sterujące zachowaniem pętli, które pozwalają wymusić zakończenie iteracji i ominąć pewne instrukcje lub całkowicie zatrzymać działanie pętli.

W tym e-materiale zaimplementujemy omawiane instrukcje w języku Java.

Ciekawi cię, jak wyglądają implementacje w innych językach programowania? Możesz się z nimi zapoznać w dwóch pozostałych e-materiałach z tej serii:

- Przerywanie pętli w języku C++,
- Przerywanie pętli w języku Python.

Więcej zadań? Sięgnij do Przerywanie pętli – zadania maturalne.

> **Twoje cele**
>
> - Scharakteryzujesz, czym jest `break` i `continue`.
> - Przeanalizujesz, jak i kiedy stosować te instrukcje w pętlach.
> - Stworzysz kilka prostych programów, w których zastosujesz instrukcje `break` i `continue`.

Film samouczek

Polecenie 1

Wypisz i zsumuj kolejne liczby nieparzyste od 1 do 100. Operację przerwij, gdy suma przekroczy liczbę 50. Porównaj swoje rozwiązanie z filmem poniżej.

Dane:

Liczba jest parzysta, jeżeli dzieli się przez 2 bez reszty.

Wynik:

Na konsoli wyświetlą się liczby nieparzyste, a pętla zostanie przerwana gdy suma będzie większa niż 50.

Lista kroków:

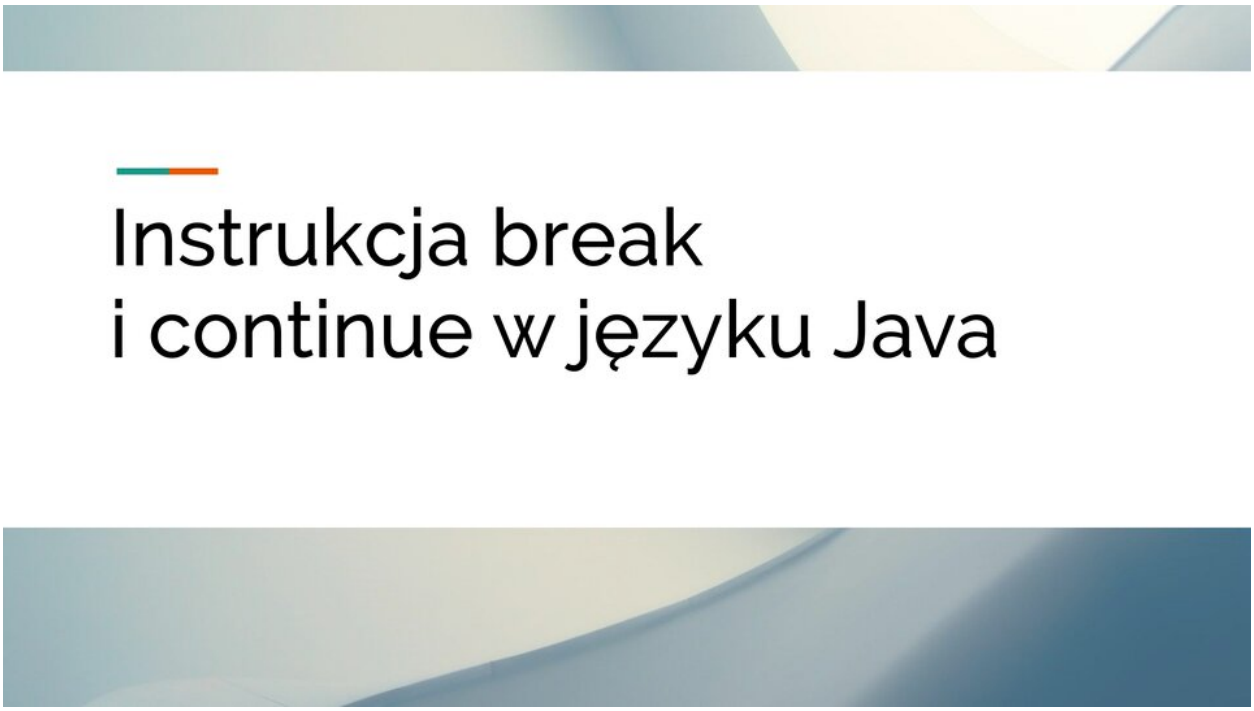
```
```

```
public class Zadanie {
    public static void main(String[] args) {

        // tutaj dopisz kod

    }
}
```

```
```



Film dostępny pod adresem /preview/resource/RhSzk8utarp1h

Nagranie filmowe przedstawiające instrukcje break i continue w języku Java.

Polecenie 2

Obejrzyj film, a następnie opisz algorytm, w którym przerwanie pętli byłoby niewskazane.

Film dostępny pod adresem /preview/resource/R9jokJFHZzHCK

Nagranie filmowe dotyczące przykładów przerywania pętli.

Przeczytaj

W języku Java instrukcje `break` i `continue` stosujemy według schematu:

```
for (wyrażenie) {
    if (warunek) {
        break;
    }
}
```

oraz:

```
for (wyrażenie) {
    if (warunek) {
        continue;
    }
}
```

Instrukcja `break`

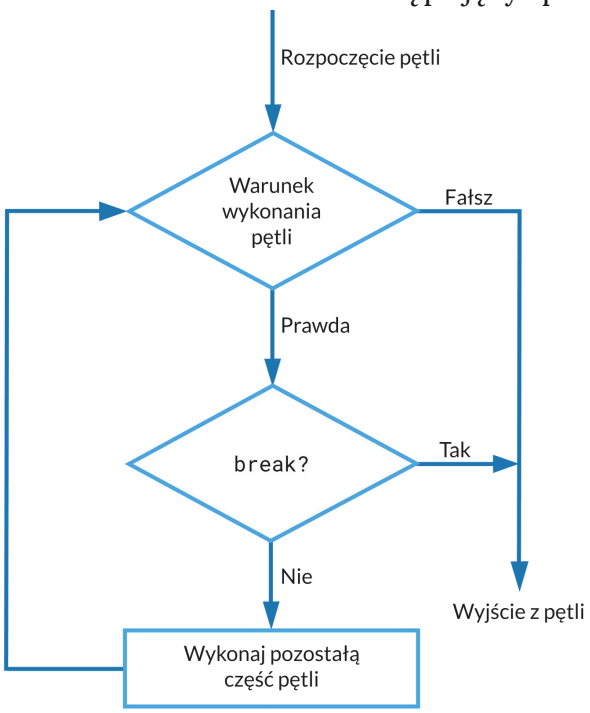
Instrukcja `break` przerywa działanie pętli przy określonym warunku `if`.

```
int i = 0;
while (true) {
    if (i == 9) {
        break;
    }

    System.out.println(i);
    i++;
}
```

System wydrukuje kolejno liczby od 0 do 8. Przy dziewiątej iteracji następuje przerwanie pętli. Gdybyśmy nie zastosowali instrukcji `break` w powyższym kodzie, pętla ta byłaby nieskończona i nasz program by się zawiesił. Dzieje się tak, ponieważ warunek tej pętli zawsze jest spełniony.

Działanie instrukcji `break` można zobrazować w następujący sposób:



Źródło: GroMar Sp. z o.o., licencja: CC BY-SA 3.0.

Jeżeli pętla natrafi na przypadek zadeklarowany w warunku `if`, to zatrzyma działanie pętli (`break`).

`Break` jest również wygodnym i czytelnym sposobem na kończenie działania pętli zagnieżdżonych, np.

```
for (int i = 0; i <= 4; i++) {
    for (int j = 0; j <= 4; j++) {
        if (i == 3) {
            break;
        }

        System.out.print("*");
    }

    System.out.println();
}
```

Taki program wydrukuje:

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

```
4
5 *****
```

To właśnie dzięki temu fragmentowi kodu:

```
if (i == 3) {
    break;
}
```

czwarta wydrukowana linia jest pusta. Dzieje się tak, ponieważ zatrzymaliśmy wewnętrzną pętle, gdy zewnętrzna jest przy iteracji `i == 3`.

Instrukcja `continue`

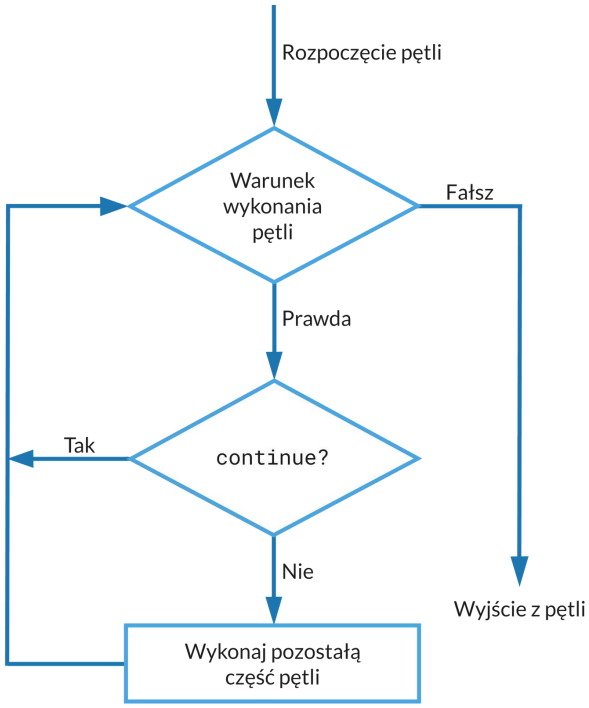
Instrukcja `continue` wymusza zakończenie iteracji przy określonym warunku `if`.

```
for (int i = 0; i < 7; i++) {
    if (i % 2 == 0) {
        continue;
    }

    System.out.println(i);
}
```

Taki program wypisze kolejno liczby 1, 3 i 5 w nowych wierszach. Dlaczego? Ponieważ pętla iteruje po liczbach mniejszych od 7, a gdy liczba jest parzysta, wymusza przejście do następnej iteracji i nie wykonuje instrukcji.

Dla `continue` schemat blokowy będzie wyglądał następująco:



Źródło: GroMar Sp. z o.o., licencja: CC BY-SA 3.0.

Jeżeli pętla natrafi na przypadek zadeklarowany w warunku `if`, to wymusi obrót pętli – nie wykona instrukcji w pętli i przejdzie do kolejnej iteracji.

Różnicę między `break` i `continue` można przeanalizować na poniższym przykładzie:

```
for (int i = 0; i < 100; i += 10) {
    if (i == 30) {
        break;
    }

    System.out.print(i + " ");
}
```

Ten kod wydrukuje w konsoli: `0 10 20`.

```
for (int i = 0; i < 100; i += 10) {
    if (i == 30) {
        continue;
    }

    System.out.print(i + " ");
}
```

Otrzymamy następujący wynik: `0 10 20 40 50 60 70 80 90`.

> **Ważne!**
>
> Instrukcje `break` i `continue` są również używane do optymalizacji działania programów, szczególnie w długich lub mocno zagnieżdżonych pętlach. Oszczędzają zasoby, bo dzięki temu program nie sprawdza niepotrzebnie wszystkich przypadków.

Pętle nazwane

Obie instrukcje są wykorzystywane w pętlach zagnieżdżonych. Można je też stosować do tworzenia pętli nazwanych (etykietowanych). W tym celu używamy etykiet, które bardzo się przydają, kiedy np. potrzebujemy zakończyć zewnętrzną pętlę warunkiem w pętli wewnętrznej.

Przykład zagnieżdżonej pętli etykietowanej:

```
labeledLoop:
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 4; j++) {
            if (i == 2) {
                break labeledLoop;
            }

            System.out.print("*");
        }

        System.out.println(" ");
    }
```

Wynik takiego programu to tym razem:

```
****
****

```

Dlaczego tak się dzieje? Ponieważ otagowany `break` zakończy zewnętrzną zaetykietowaną pętlę. Bez etykiety labeledLoop zostałaby zakończona wewnętrzna pętla, w której `break` się znajduje. Gdybyśmy użyli pętli nieetykietowanej, czyli:

```
for (int i = 0; i < 4; i++) {
    for (int j = 0; j < 4; j++) {
```

```
3           if (i == 2) {
4               break;
5           }
6
7           System.out.print("*");
8       }
9
10      System.out.println(" ");
11  }
```

przy `i` równym 2 instrukcja `break` zakończyłaby działanie jedynie pętli wewnętrznej, a zewnętrzna przeszłaby do kolejnej iteracji. Otrzymamy wtedy taki wynik:

```
1  ****
2  ****
3
4  ****
```

> **Ważne!**
>
> Mimo że instrukcje `break` i `continue` są przydatne w pewnych określonych sytuacjach, to zdecydowanie odradza się ich nadużywania. Rozbudowane, wielokrotnie zagnieżdżone pętle z wieloma instrukcjami `break` i `continue` są trudne do analizy, szczególnie gdy wracamy do swojego kodu po dłuższym czasie lub ktoś musi czytać nasz kod.

Słownik

> **etykieta**
>
> to znacznik do oznaczania konkretnej pętli, której działanie chcemy zmodyfikować za pomocą instrukcji `break` lub `continue`

Sprawdź się

Pokaż ćwiczenia: 

Ćwiczenie 1

Używając instrukcji `break` lub `continue`, napisz program, który wypisze liczby podzielne przez 3 z przedziału od 0 do 100 (włącznie).

Twoje zadania

1. Wypisz wszystkie liczby podzielne przez 3 z określonego przedziału.

Ćwiczenie 2

Napisz program, który wypisze w konsoli cztery linie. Pierwsza i ostatnia powinny zawierać po cztery znaki * rozdzielone spacjami, a dwie pozostałe powinny być puste.

Twoje zadania

1. Wypisz wskazaną figurę.

Ćwiczenie 3

Zmodyfikuj program z ćwiczenia nr 2. Dodaj dwie pionowe linie, tak aby utworzyć ze znaków * kwadrat.

Twoje zadania

1. Wypisz wskazaną figurę.

Ćwiczenie 4

Ponownie zmodyfikuj program z ćwiczenia nr 2. Powiększ wysokość i szerokość figury o jeden znak *. Użyj znaku # zamiast * w środku obu pionowych linii figury.

Twoje zadania

1. Wypisz wskazaną figurę.

Ćwiczenie 5

Napisz program, który wypisze liczby z przedziału od 100 do 900, których suma cyfr setek i jedności wynosi 7. Użyj etykiety i zatrzymaj działanie pętli, gdy cyfra setek lub jedności jest większa od 7.

Twoje zadania

1. Wypisz liczby, które spełniają warunki zadania.

Ćwiczenie 6

Rozbuduj pętlę `do-while`, wykorzystując instrukcje `break` oraz `continue`. Napisz program, który znajdzie pierwsze trzy liczby jednocześnie podzielne przez 7 i 12.

Twoje zadania

1. Wypisz pierwsze trzy liczby podzielne jednocześnie przez 7 i 12

Dla nauczyciela

Autor: Natalia Bujnowska
Przedmiot: Informatyka

Temat: Przerywanie pętli w języku Java

Grupa docelowa:

Szkoła ponadpodstawowa, liceum ogólnokształcące, technikum

Podstawa programowa:

Cele kształcenia – wymagania ogólne

> 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

> II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.
>
> > Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:
> >
> > > 2) stosuje zasady programowania strukturalnego i obiektowego w rozwiązywaniu problemów;

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):

- Scharakteryzujesz, czym jest `break` i `continue`.
- Przeanalizujesz, jak i kiedy stosować te instrukcje w pętlach.
- Stworzysz kilka prostych programów, w których zastosujesz instrukcje `break` i `continue`.

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: „Przerywanie pętli w języku Java". Uczniowie zapoznają się z treściami w sekcji „Film samouczek" w kontekście programowania.

Faza wstępna:

1. Nauczyciel wyświetla temat i cele zajęć. Prosi uczniów, by na podstawie wiadomości zdobytych przed lekcją zaproponowali kryteria sukcesu.
2. Rozpoznawanie wiedzy potocznej uczniów.

Faza realizacyjna:

1. **Praca z tekstem.** Jeżeli przygotowanie uczniów do lekcji jest niewystarczające, nauczyciel prosi o indywidualne zapoznanie się z treścią zawartą w sekcji „Film samouczek". Każdy uczestnik zajęć podczas cichego czytania wynotowuje najważniejsze kwestie poruszane w tekście.
2. **Praca z multimedium.** Nauczyciel wyświetla zawartość sekcji „Film samouczek". Uczniowie wspólnie zapoznają się z filmem, a następnie opisują algorytm, w którym

przerwanie pętli byłoby niewskazane.
3. **Ćwiczenie umiejętności.** Nauczyciel przechodzi do sekcji „Sprawdź się". Zapowiada uczniom, że w kolejnym kroku będą rozwiązywać ćwiczenia nr 1-6 – od najprostszych do najtrudniejszych – i będą to robić wspólnie. Wybrany uczestnik zajęć czyta po kolei polecenia. Po każdym przeczytanym poleceniu ochotnik udziela odpowiedzi. Reszta uczniów ustosunkowuje się do niej, proponując swoje pomysły. Nauczyciel w razie potrzeby koryguje odpowiedzi, dopowiada istotne informacje, udziela uczniom informacji zwrotnej.

Faza podsumowująca:

1. Wybrany uczeń podsumowuje zajęcia, zwracając uwagę na nabyte umiejętności, omawia ewentualne problemy podczas rozwiązania ćwiczeń.

Praca domowa:

1. Uczniowie wykonują Wykonaj program przedstawiony na filmie w sekcji „Film samouczek".

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:

- Multimedia w sekcjach: „Film samouczek", „Przeczytaj", „Sprawdź się" można wykorzystać jako materiał służący powtórzeniu materiału.