# Lekcja – zbiory

HashSet to struktura danych, która przechowuje unikalne elementy w nieuporządkowany sposób. Wykorzystuje funkcję haszującą do mapowania elementów na indeksy w wewnętrznej tablicy, co umożliwia szybkie wyszukiwanie, wstawianie i usuwanie elementów. Kluczową cechą HashSetu jest to, że nie dopuszcza on powtarzających się wartości, co oznacza, że każdy element może pojawić się w zbiorze tylko raz.

W momencie dodawania nowego elementu do HashSetu, struktura ta używa funkcji haszującej do obliczenia indeksu tablicy, pod którym element powinien zostać zapisany. Jeśli w obliczonej lokalizacji nie ma jeszcze żadnego elementu, nowy element jest tam dodawany. W przypadku, gdy funkcja haszująca przypisze dwa różne elementy do tego samego indeksu (co jest znane jako kolizja haszy), HashSet stosuje pewne metody rozwiązania kolizji, takie jak łańcuchowanie lub adresowanie otwarte, aby znaleźć alternatywne miejsce dla nowego elementu.

Wyszukiwanie elementu w HashSet’cie również wykorzystuje funkcję haszującą. Dzięki obliczeniu indeksu na podstawie wartości hasza elementu, możliwe jest szybkie określenie, czy element znajduje się w zbiorze czy nie, bez konieczności przeglądania każdego elementu indywidualnie.

Usunięcie elementu z HashSetu polega na znalezieniu jego lokalizacji za pomocą funkcji haszującej i usunięciu go z tablicy. Dzięki temu operacja usunięcia jest również wydajna.

Podsumowując, HashSet jest wydajną strukturą danych do przechowywania unikalnych elementów. Dzięki wykorzystaniu funkcji haszującej do bezpośredniego adresowania w wewnętrznej tablicy, operacje takie jak wyszukiwanie, wstawianie i usuwanie mogą być realizowane bardzo szybko. Jednakże, HashSet nie zachowuje kolejności dodawania elementów, co oznacza, że nie jest odpowiedni w przypadkach, gdy kolejność elementów ma znaczenie.

## Funkcja hashująca

Funkcja hashująca to algorytm, który przekształca dowolną ilość danych wejściowych (nazywanych "wiadomością") na skróconą, stałodługościową sekwencję bitów, zwaną "hashem" lub "wartością skrótu". Wartości wygenerowane przez funkcje hashujące są unikalne dla unikalnych danych wejściowych; jednak ze względu na ograniczoną długość wyniku, różne dane wejściowe mogą teoretycznie generować tę samą wartość hash (zjawisko to nazywane jest "kolizją").

# Przykłady użycia

## Usuwanie duplikatów z listy

**import java.util.Arrays;**

**import java.util.HashSet;**

**import java.util.List;**

**import java.util.Set;**

**List<Integer> liczby = Arrays.asList(1, 2, 2, 3, 4, 4, 5);**

**Set<Integer> unikalneLiczby = new HashSet<>(liczby);**

**System.out.println(unikalneLiczby); // Wyświetli [1, 2, 3, 4, 5]**

## Sprawdzanie obecności elementu

**Set<String> nazwiska = new HashSet<>();**

**nazwiska.add("Kowalski");**

**nazwiska.add("Nowak");**

**boolean czyZawiera = nazwiska.contains("Kowalski"); // true**

**System.out.println("Czy zawiera Kowalski? " + czyZawiera);**

## Operacje zbiorów

**Set<Integer> zbiorA = new HashSet<>(Arrays.asList(1, 2, 3));**

**Set<Integer> zbiorB = new HashSet<>(Arrays.asList(3, 4, 5));**

**// Unia**

**Set<Integer> unia = new HashSet<>(zbiorA);**

**unia.addAll(zbiorB);**

**System.out.println("Unia: " + unia);**

**// Przecięcie**

**Set<Integer> przeciecie = new HashSet<>(zbiorA);**

**przeciecie.retainAll(zbiorB);**

**System.out.println("Przecięcie: " + przeciecie);**

**// Różnica**

**Set<Integer> roznica = new HashSet<>(zbiorA);**

**roznica.removeAll(zbiorB);**

**System.out.println("Różnica: " + roznica);**

## Zapobieganie duplikatom podczas dodawania elementów

**Set<Integer> identyfikatory = new HashSet<>();**

**identyfikatory.add(100);**

**identyfikatory.add(101);**

**identyfikatory.add(100); // Ta linia nie ma efektu, ponieważ 100 jest już w zbiorze**

**System.out.println(identyfikatory); // Wyświetli [100, 101]**

## Zadanie 1

Zaimplementuj strukturę danych Zbior, która będzie posiadac metody add, remove i contains.

## Zadanie 2

Problem: Dany jest ciąg liczb całkowitych. Twoim zadaniem jest znalezienie liczby unikalnych elementów w tym ciągu.

Przykład:

Dla ciągu 1, 2, 3, 4, 2, 3, odpowiedź powinna wynosić 4, ponieważ mamy cztery unikalne liczby: 1, 2, 3, i 4.

## Zadanie 3

Problem: Dany jest ciąg liczb, gdzie każda liczba pojawia się dwa razy, oprócz jednej, która pojawia się raz. Znajdź tę liczbę.

Przykład:

Dla ciągu 1, 2, 3, 4, 2, 3, 1, jedynka i dwójka pojawiają się dwa razy, a czwórka tylko raz. Odpowiedzią jest 4.

## Zadanie 4

Problem: Dane są dwa zbiory liczb całkowitych. Sprawdź, czy mają one przynajmniej jeden wspólny element.

Przykład:

Zbiór A: 1, 3, 5, 7

Zbiór B: 2, 4, 6, 7

Zbiory mają wspólny element 7, więc odpowiedź to true.