**Інструкція до програми**

Програма реалізує клас **CustomSet**, який є користувацькою реалізацією множини (*set*). Вона гарантує, що елементи залишаються унікальними, і підтримує динамічне збільшення ємності.

Клас **CustomSet** успадковує інтерфейс Set і реалізує всі його методи, дотримуючись основних принципів роботи множин.

**Основні можливості**

1. **Гарантія унікальності**:
   * Елементи не дублюються.
   * Перевірка унікальності виконується через метод contains().
2. **Динамічна структура**:
   * Місткість масиву збільшується автоматично при досягненні ліміту.
   * Збільшення ємності виконується за допомогою коефіцієнта росту (*growth factor*).
3. **Підтримка основних операцій множини**:
   * Додавання, видалення, перевірка наявності елементів.
   * Методи addAll, retainAll, removeAll.
4. **Конструктори**:
   * Порожній.
   * З одним елементом.
   * Із колекції.
5. **Сумісність із Collection**:
   * Метод toArray() для перетворення множини в масив.

**Опис методів класу CustomSet**

**Основні методи**

1. **add(E e)**:
   * Додає елемент до множини.
   * Перевіряє, чи вже існує елемент, і додає лише новий.
2. **remove(Object o)**:
   * Видаляє елемент, якщо він присутній.
   * Переміщує елементи після видаленого для уникнення порожніх місць.
3. **contains(Object o)**:
   * Перевіряє наявність елемента в множині.
4. **toString()**:
   * Повертє рядкове представлення множини.
5. **ensureCapacity()**:
   * Збільшує розмір внутрішнього масиву, якщо місця недостатньо.

**Інші методи**

* **addAll(Collection<? extends E> c)**: Додає всі елементи з колекції.
* **retainAll(Collection<?> c)**: Зберігає лише ті елементи, які є в зазначеній колекції.
* **removeAll(Collection<?> c)**: Видаляє всі елементи, присутні в зазначеній колекції.

**Приклад використання**

**Код**

public class Main {

public static void main(String[] args) {

// Приклад 1: Порожній конструктор

CustomSet<Integer> set1 = new CustomSet<>();

set1.add(10);

set1.add(20);

set1.add(30);

System.out.println("Set 1: " + set1);

// Приклад 2: Конструктор із одним елементом

CustomSet<String> set2 = new CustomSet<>("Hello");

set2.add("World");

set2.add("Hello"); // Дублікат не буде додано

System.out.println("Set 2: " + set2);

// Приклад 3: Конструктор із колекцією

CustomSet<Double> set3 = new CustomSet<>(Arrays.asList(1.1, 2.2, 3.3, 1.1));

System.out.println("Set 3: " + set3);

// Видалення і перевірка елементів

set3.remove(2.2);

System.out.println("Set 3 після видалення: " + set3);

System.out.println("Set 3 містить 3.3: " + set3.contains(3.3));

}

}

**Вихід**

Set 1: [10, 20, 30]

Set 2: [Hello, World]

Set 3: [1.1, 2.2, 3.3]

Set 3 після видалення: [1.1, 3.3]

Set 3 містить 3.3: true

**Ключові моменти**

1. **Використання внутрішнього масиву**:
   * Динамічне розширення масиву для зберігання елементів.
2. **Унікальність елементів**:
   * Перевіряється через метод contains().
3. **Повна реалізація інтерфейсу Set**:
   * Сумісність із іншими класами Java, такими як Collection.
4. **Гнучкість конструкції**:
   * Можливість створювати множину з нуля, з одним елементом або з колекції.