ЛАБОРАТОРНА РОБОТА 5

Тема: Використання Collection Framework

Парадигма ООП підтримує концепцію інкапсуляції даних всередині класів. Крім цього важливим є спосіб розміщення даних, тому що це може суттєво вплинути як на реалізацію методів класу, так і на ефективність рішення певної задачі. В Java підтримуються набори даних або колекції. Колекції представляють деяку групу об'єктів — елементи колекції. Різноманіття видів колекцій дозволяє підвищити ефективність роботи програм у ключових для неї частинах, що інколи зі зниженням ефективності у менш важливих. Каркас колекцій - Collection Framework, було розроблено для за для досягнення декількох цілей. Першою було забезпечення високої продуктивності. Наступною — забезпечення однаковості функціювання колекцій з високою ступеню взаємодії. Також каркас колекцій побудовано на єдиному наборі стандартних інтерфейсів, що забезпечує просте розширення та адаптацію колекцій. Останньою є впровадження механізмів інтеграції стандартних масивів.

Також невід'ємною частиною філософії каркасу колекцій є узагальнення алгоритмів. Алгоритми оперують колекціями і визначені у вигляді статичних методів у класі Collections. Вони доступні усім колекціям і не потребують реалізації їх власної версії у кожному класі колекції. Алгоритми надають стандартні засоби для маніпулювання колекціями.

Інтерфейси та класи каркасу колекцій знаходяться у пакеті java.util, тому для їх використання цей пакет слід імпортувати до проекту.

У каркасі колекцій визначено декілька інтерфейсів, на вершині ієрархії яких знаходиться інтерфейс Collection. Інтерфейс Collection є фундаментальним інтерфейсом для класів, які підтримують колекції і він імплементується у будь-якому класі каркасу колекцій. В інтерфейсі Collection є два основних методи:

```
public interface Collection<E>
{
   boolean add (E element);
   Iterator <E> iterator();
}
```

де <E> позначає тип об'єктів, які буде утримувати колекція.

Метод add() додає елемент до колекції та повертає або значення true, якщо колекція змінилась, або false, якщо не змінилась. Наприклад, при спробі додати до множини вже існуючий об'єкт метод add() буде проігноровано тому, що за визначенням множина не може мати об'єкти, що дублюються.

Інтерфейс Collection розширює інтерфейс Iterable. Це дозволяє перебирати будь-які колекції за допомогою циклу for у стилі for each. Метод iterator() повертає об'єкт-ітератор, який реалізує інтерфейс Iterator, що використовується для послідовного звернення до елементів колекції.

Крім названих методів, у класах, що будь імплементувати інтерфейс Collection необхідно буде реалізовувати і інші методи (рис.1) або залишити їх у вигляді типу:

```
@Override
public boolean ім'я_методу(Collection clctn) {
   throw new UnsupportedOperationException("Not supported yet!");
}
```

де ***ім'я_методу*** - певний метод інтерфейсу Collection.

Такий тип реалізації методів інколи називають “заглушкою” тому, що вона дозволяє створити клас з імплементацією певного інтерфейсу але фактично не реалізує функціональність усіх його методів. Використання виключення UnsupportedOperationException() дозволяє виконати його перехоплення у коді за допомогою try{}catch(){}, де у блоці try{} буде робитись спроба виконати метод, що залишається “заглушкою”.

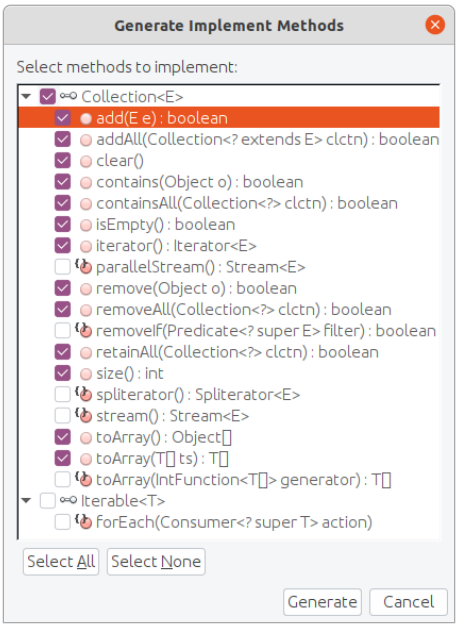

Рис.1. Абстрактні методи інтерфейсу Collection позначено значком ☑.

Серед інтерфейсів каркасу колекцій можна позначити наступні:
– Collection — базовий інтерфейс для усіх класів колекцій та інших інтерфейсів;
– Queue — наслідує інтерфейс Collection та надає функціональність для структур даних виду “черга”;
– Deque — наслідує інтерфейс Queue і надає функціональність для структур типу “двонаправла черга”;
– List — наслідує інтерфейс Collection і надає функціональність для простих списків;
– Set — розширює інтерфейс Collection та використовується для зберігання множини унікальних об’єктів;
– SortedSet — розширює інтерфейс Set для створення сортованих колекцій;
– NavigableSet — розширює інтерфейс SortedSet для створення колекцій, в яких можна здійснити пошук за відповідністю;
– Map — призначено для створення структур даних у вигляді словника, де кожний елемент має певний ключ та значення. На відміну від інших інтерфейсів каркасу колекцій він не розширює інтерфейс Collection.

Наведені інтерфейси частково реалізуються такими абстрактними класами:

– AbstractCollection - базовий абстрактний клас для інших класів колекцій, який використовує інтерфейс Collection;
– AbstractLis — розширює клас AbstractCollection та застосовує інтерфейс List, що призначено для створення колекцій у вигляді списків;

− AbstractSet — розширює клас AbstractCollection та застосовує інтерфейс Set для створення колекцій у вигляді множини;
− AbstractQueue — розширює клас AbstractCollection та використовує інтерфейс Queue, призначається для створення колекцій у вигляді черг та стеків;
− AbstractSequentialList — розширює клас AbstractList та реалізує інтерфейс List, використовується для створення зв’язаних списків;
− AbstractMap — використовує інтерфейс Map, призначається для створення колекцій за типом словника з об’єктами у вигляді пари “ключ — значення”.

За допомогою наведених інтерфейсів та абстрактних класів в Java реалізується широка палітра класів колекцій: списки, множини, черги, відображення, тощо. Можна позначити наступні:
− ArrayList це клас, що реалізує простий список об’єктів;
− LinkedList це клас, що реалізує зв’язаний список;
− ArrayDeque це клас, що реалізує двонаправлену чергу;
− HashSet це клас, що реалізує набір об’єктів або хеш-множину, де кожний елемент в якості ключа має унікальний хеш-код;
− TreeSet це клас, що реалізує набір відсортованих об’єктів у вигляді дерева;
− LinkedHashSet це клас, що реалізує зв’язану хеш-множину;
− PriorityQueue це клас, що реалізує чергу з пріоритетами;
− HashMap це клас, що реалізує структуру даних у вигляді словника, в якому кожний об’єкт має унікальний ключ і деяке значення;
− TreeMap це клас, що реалізує структуру даних у вигляді дерева, де кожний елемент має унікальний ключ та деяке значення.

ПРАКТИЧНЕ ЗАВДАННЯ

1. Вивчить теоретичну частину лабораторної роботи.
2. Для наданого файлу розробіть клас відповідно до структури даних в ньому.
3. Використовуючи клас ArrayList напишіть програму, в якій:
   - зчитуються дані з файлу та заносяться у вигляді об’єктів створеного класу до ArrayList;
   - послідовно виводяться данні з ArrayList за допомогою циклу for each;
   - дозволяється визначити індекс заданого об’єкту у списку;
   - дозволяється повернути об’єкт зі списку за певним індексом;
   - дозволяється отримати набір елементів у вигляді окремого списку, що знаходяться в ArrayList між індексами start та end;
   - дозволяється виконати сортування за допомогою певного компаратора та виведенням результату сортування;
   - дозволяється додати новий об’єкт до списку;
   - дозволяється видалити об’єкт зі списку за його індексом або за його значенням.

   Наведіть програмний код вказаних реалізацій та скриншоти з поясненням отриманих результатів.

4. Використовуючи клас LinkedList напишіть програму, в якій:
   - зчитуються дані з файлу та заносяться у вигляді об’єктів створеного класу до LinkedList;
   - послідовно виводяться данні зі списку у прямому та зворотньому порядку;
   - дозволяється визначити перший та останній елементи у списку;

- дозволяється визначити індекс заданого об’єкту у списку;
- дозволяється повернути об’єкт зі списку за певним індексом;
- дозволяється отримати набір елементів у вигляді окремого списку, що знаходяться у списку між індексами start та end;
- дозволяється виконати сортування за допомогою певного компаратора та виведенням результату сортування;
- дозволяється додати новий об’єкт у початок або у кінець списку;
- дозволяється видалити об’єкт зі списку за його індексом або за його значенням.

Наведіть програмний код вказаних реалізацій та скриншоти з поясненням отриманих результатів.

5. Використовуючи клас ArrayDeque напишіть програму, в якій:
   - зчитуються дані з файлу та заносяться у вигляді об’єктів створеного класу до ArrayDeque;
   - дозволяється визначити перший та останній елементи у черзі;
   - дозволяється додати новий об’єкт у початок або у кінець списку;
   - послідовно виводяться данні із черги у прямому та зворотньому порядку;
   - використовується виключення NoSuchElementException для завершення дії вибирання елементів із черги;
   - дозволяється видалити перший знайдений об’єкт зі списку за його значенням;
   - дозволяється видалити останній знайдений об’єкт зі списку за його значенням.

   Наведіть програмний код вказаних реалізацій та скриншоти з поясненням отриманих результатів. Порівняйте ефективність структур ArrayDeque та LinkedList для певних операцій.

6. Використовуючи клас HashSet напишіть програму, в якій:
   - зчитуються дані з файлу та заносяться у вигляді об’єктів створеного класу до HashSet;
   - дозволяється визначити наявність елементу у множині;
   - дозволяється додати новий об’єкт у множину;
   - дозволяється видалити об’єкт з множини;
   - послідовно виводяться данні із множини;
   - дозволяється визначити хешкод певного елементу;
   - дозволяється повернути множину у форматі масиву;
   - виконати сортування елементів множини за заданим критерієм.

   Наведіть програмний код вказаних реалізацій та скриншоти з поясненням отриманих результатів. Порівняйте ефективність реалізації сортування елементів у множині з реалізацією на LinkedList.

7. Розширте можливості наведеного у попередньому пункті коду за допомогою імплементації інтерфейсу SortedSet. Реалізуйте сортування множини та порівняйте її ефективність з попереднім пунктом. Наведіть програмний код вказаних реалізацій та скриншоти з поясненням отриманих результатів.
8. Підготуйте звіт.