Introduction to Collections in Java

Marcin Kurzyna

November 26, 2024

Contents

1 What Are Collections in Java?

Collections in Java are part of the `java.util` package. They provide a framework for storing and manipulating groups of objects efficiently. Collections include interfaces, implementations (classes), and algorithms to work with groups of data.

1.1 Why Use Collections?

- Dynamically manage the size of data structures.
- Easily perform operations like searching, sorting, and iterating.
- Reduce the need for manual data structure implementations.

2 Key Interfaces in the Collection Framework

The Java Collection Framework defines several interfaces. Here are the primary ones:

- `Collection<E>`
- `List<E>`
- `Set<E>`
- `Queue<E>`
- `Map<K, V>`

Each interface represents a different type of data structure.

2.1 `List<E>`

An ordered collection allowing duplicate elements.

- Implementations: `ArrayList`, `LinkedList`, `Vector`.
- Common Methods: `add()`, `get()`, `remove()`, `size()`.

2.2 Set<E>

A collection that does not allow duplicate elements.

- Implementations: `HashSet`, `LinkedHashSet`, `TreeSet`.
- Common Methods: `add()`, `contains()`, `remove()`.

2.3 Queue<E>

A collection designed for holding elements before processing (FIFO order by default).

- Implementations: `PriorityQueue`, `LinkedList`.
- Common Methods: `offer()`, `poll()`, `peek()`.

2.4 Map<K, V>

A collection of key-value pairs, where keys are unique.

- Implementations: `HashMap`, `LinkedHashMap`, `TreeMap`.
- Common Methods: `put()`, `get()`, `remove()`, `keySet()`.

3 Common Collection Classes and Usage

3.1 ArrayList

`ArrayList` is a resizable array implementation of the `List` interface.

```
// Example: Using an ArrayList
import java.util.ArrayList;

public class Main {
    public static void main(String[] args) {
        ArrayList<String> list = new ArrayList<>();
        list.add("Apple");
        list.add("Banana");
        list.add("Cherry");

        for (String item : list) {
            System.out.println(item);
        }
```

```
    }
}
```

3.2 HashSet

HashSet is a fast implementation of the Set interface.

```
// Example: Using a HashSet
import java.util.HashSet;

public class Main {
    public static void main(String[] args) {
        HashSet<Integer> set = new HashSet<>();
        set.add(1);
        set.add(2);
        set.add(3);
        set.add(2); // Duplicate element, ignored

        for (int number : set) {
            System.out.println(number);
        }
    }
}
```

3.3 HashMap

HashMap is a popular implementation of the Map interface.

```
// Example: Using a HashMap
import java.util.HashMap;

public class Main {
    public static void main(String[] args) {
        HashMap<String, Integer> map = new HashMap<>();
        map.put("Alice", 25);
        map.put("Bob", 30);

        for (String key : map.keySet()) {
            System.out.println(key + ": " +
               map.get(key));
        }
    }
}
```

4 Iterating Over Collections

Java offers multiple ways to iterate over collections:

- `for-each` loop
- `Iterator` interface
- Streams API

4.1 Using an Iterator

```java
// Example: Using an Iterator
import java.util.ArrayList;
import java.util.Iterator;

public class Main {
    public static void main(String[] args) {
        ArrayList<String> list = new ArrayList<>();
        list.add("Dog");
        list.add("Cat");
        list.add("Horse");

        Iterator<String> iterator = list.iterator();
        while (iterator.hasNext()) {
            System.out.println(iterator.next());
        }
    }
}
```

5 Comparison of Key Implementations

Interface	Implementation	Use Case
`List`	`ArrayList`	Fast random access, frequent reads
	`LinkedList`	Frequent inserts/removals
`Set`	`HashSet`	Unique elements, fast lookup
	`TreeSet`	Sorted unique elements
`Map`	`HashMap`	Key-value pairs, fast lookup
	`TreeMap`	Sorted key-value pairs

6 Exercises

1. **Working with `ArrayList`:** Create a program to store a list of student names. Allow the user to add, remove, and display the names. Use an `ArrayList`.

2. **Unique Elements with `HashSet`:** Write a program to read a list of numbers from the user and store only unique numbers in a `HashSet`. Print the resulting set.

3. **Key-Value Pairs with `HashMap`:** Create a phonebook application where the user can store names and phone numbers using a `HashMap`. Implement options to add, delete, and search for a contact.

4. **Sorting with `TreeSet`:** Write a program to store a collection of strings in a `TreeSet`. Add several strings and print them in sorted order.

5. **Iterating with `Iterator`:** Create a program to store a collection of tasks (as strings) in a `List`. Use an `Iterator` to iterate over the tasks and print each one.