Collections and Swing Framework

- Pratiksha Dhavale

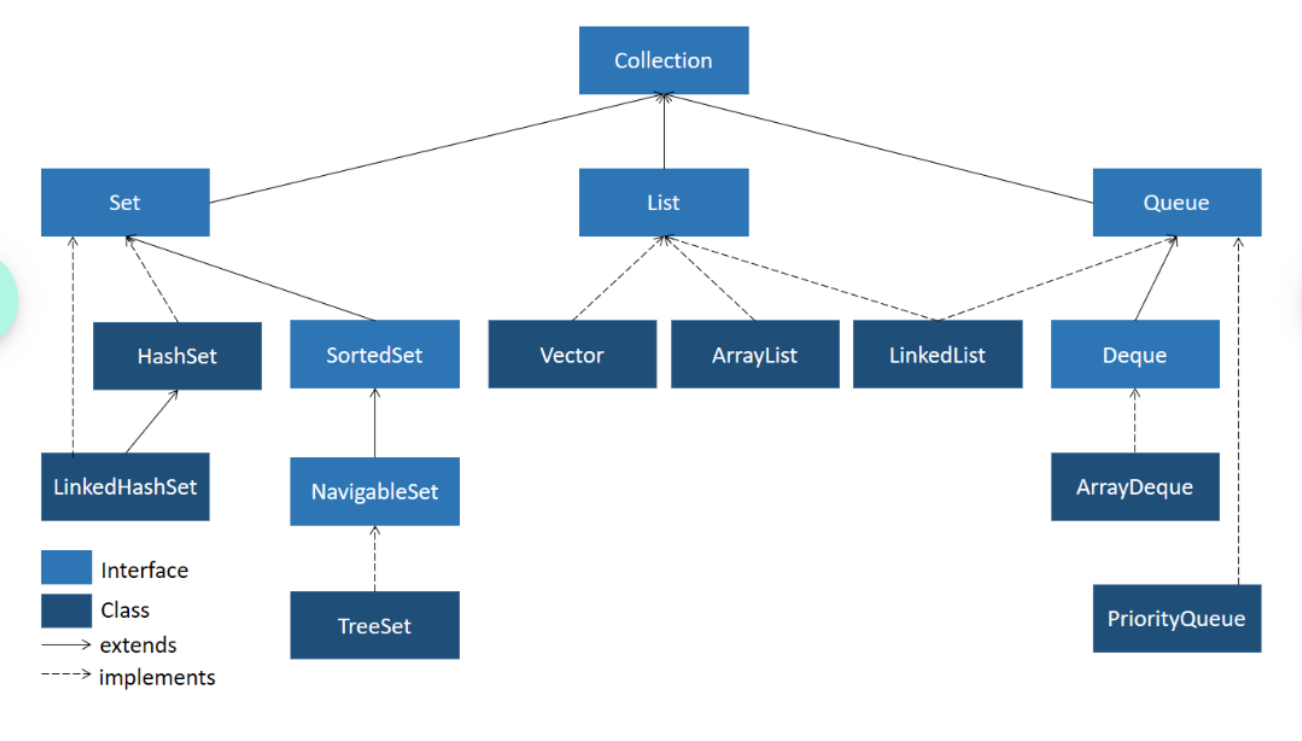
Collection

The **Collection in Java** is a framework that provides an architecture to store and manipulate the group of objects.

Java Collections can achieve all the operations that you perform on a data such as searching, sorting, insertion, manipulation, and deletion.

Java Collection means a single unit of objects. Java Collection framework provides many interfaces (Set, List, Queue, Deque) and classes (ArrayList, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet).

List


Collection
Set
List
Queue
HashSet
SortedSet
Vector
ArrayList
LinkedList
Deque
LinkedHashSet
NavigableSet
ArrayDeque
Interface
Class
extends
implements
TreeSet
PriorityQueue

Set

Let us assume that the following food items are ordered. Observe that there are repeated entries of few food items.

Now, the task is to find out the unique entries in the list. Instead of manually finding out the duplicate entries, it would be better if there is a collection which can automatically remove duplicate elements.

This can be done by using a collection called Set.

Duplicates will be removed only unique will be displayed

HashSet

HashSet uses hash tables for storing the unique elements. There is no guarantee on the iteration order of the set. HashSet depends on the equals() and hashCode() methods for detecting duplicates and null values are allowed.

Example

```java
import java.util.HashSet;
import java.util.Set;

public class HashSetExample {
    public static void main(String[] args) {
        //Creating HashSet object
        Set<String> food = new HashSet<String>();
        food.add("Pasta"); // Adding elements to the HashSet
        food.add("Noodles");
        food.add("Sandwich");
        food.add("Pasta");
        food.add("Burger");
        food.add("Noodles");
        System.out.print("Set output without the duplicates: ");
        System.out.println(food);
    }
}
```

LinkedHashSet

LinkedHashSet uses a combination of hash tables and linked lists for storing elements. Elements are returned in the order of their insertion. LinkedHashSet depends on the equals() and hashCode() methods for detecting duplicates and null values are allowed.

Example

```
import java.util.LinkedHashSet;// Importing the LinkedHashSet class
import java.util.Set;

public class Tester {
    public static void main(String[] args) {
        Set<String> food = new LinkedHashSet<String>(); // Creating LinkedHashSet object
        food.add("Pasta"); // Adding elements to the LinkedHashSet
        food.add("Noodles");
        food.add("Sandwich");
        food.add("Pasta");
        food.add("Burger");
        food.add("Noodles");
        System.out.print("The duplicates are removed and displayed in the order in which
        System.out.println(food);

    }
}
```

Map

Take an example of a bookstore. The number of books available in the store is huge and it becomes very difficult to locate the books.

Map

The book store wants a way in which the name of a book can be mapped to the shelf number on which the book is placed for easier access.

Here, there is a need to store two values for every element - book name and shelf number.

The collections that you have seen till now allow us to store a single value for a given element but not two values.

In such cases, maps can be used which can be used to store key-value pairs.

HashMap

A map represents a mapping between a key and a value. It is used to store data similar to a dictionary where lookup is performed based on a key and the respective value is returned.

HashMap is known as a HashMap because it uses a technique called hashing which converts a large value to a smaller value which can be used for indexing and easier retrieval of data.

HashMap Methods

Method	Description
V put(Object key, Object value)	inserts the specified key-value pair in the map
V remove(Object key)	deletes the element for the specified key
V get(Object key)	returns the value for the specified key
boolean replace(K key, V oldValue, V newValue)	replaces the old value with the new value for the specified key
void clear()	removes all key-value pairs from the map
int size()	returns the number of key-value pairs present in the map
boolean isEmpty()	returns true if map is empty

Swing Framework

Introduction

Java Swing is a GUI Framework that contains a set of classes to provide more powerful and flexible GUI components than AWT. Swing provides the look and feel of modern Java GUI. Swing library is an official Java GUI tool kit released by Sun Microsystems. It is used to create graphical user interface with Java.

Features of Swing

1. Platform Independent
2. Customizable
3. Extensible
4. Configurable
5. Lightweight
6. Rich Controls
7. Pluggable Look and Feel

AWT vs SWING

AWT(Abstract Window Toolkit) was platform dependent

It was heavy weight

Limited set of components

SWING is platform independent

Lightweight

More components and powerful

SWING

Swing is java’a extended library

Import javax.swing.*

The javax.swing package provides classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etc.

Prerequisites:

Basic knowledge of java

IDE

JDK

Set-up

Install any IDE of your choice

Install JDK

Set path variable

JFrame

Methods	Description
setVisible()	Sets visibility of the frame
setSize()	Sets size of the Frame
setDefaultCloseOperation()	Terminates the program once frame is closed
setLocation()	Sets x and y coordinate for the frame
setTitle()	Sets title of the frame
setIconImage()	Change icon
getContent()	
setBackground()	

JLabel

setText(text)	Sets the text for the label
setFont(font)	Sets font of the label

JTextField

Method	Description
setText()	Sets default text
setFont()	Sets font
setBackground()	Background color of the field
setForeground()	Font color of the text
setEditable(false)	Disables editing

JpasswordField

Method	Description
setEchoChar(“*”)	To change character(default is dot)
setEchoChar((char)0)	To show password

JButton

Method	Description
setCursor()	Sets cursor for button
setEnabled(false)	Disables the button
setVisible()	hide/show the button

JButton Events

ActionListener	Interface
Void ActionPerformed()	Abstract method
addActionListener()	Binds method to button