Collections Framework

A set of Collection classes and interfaces is called a collection framework.

A collection is an object that represents a group of objects.

Arranging data in different formats is called data structures.

All collection classes and interfaces are part of java.util package.

The collection framework helps to increase programming speed and quality.

# Core Interfaces

Here is the list of the core collection interfaces:

* Collection: It is a root interface in a one-dimensional collection hierarchy.
* List: it extends the Collection interface and it allows duplicate elements.
* Set: it extends the collection interface and it does not allow duplicate elements.
* Map: It is a root interface in a two-dimensional collection hierarchy, it contains key and value parts. It does not allow duplicate keys but values may be duplicated.

**A screenshot of a computer

Description automatically generated with medium confidence**

* SortedSet: A SortedSet is a set in which elements are sorted.
* SortedMap: A sortedmap is a map in which key and value pairs are sorted based on the keys.
* NavigableSet: it is used to navigate elements in a set.
* NavigableMap: it is used to navigate key and value pairs in a map.
* Queue: it is called as first in first out list, it allows insertion at rear end and deletion at front end.
* Deque: it stands for double-ended queue. It allows insertion and deletion on both sides.

These interfaces are the foundation of the collection framework.

**Timeline

Description automatically generated**

# General Purpose Implementation

Core collection interfaces implementation classes are called general-purpose implementations.

* ArrayList
* LinkedList
* HashSet
* LinkedHashSet
* TreeSet
* HashMap
* LinkedHashMap
* TreeMap
* PriorityQueue
* ArrayDeQue

## ArrayList

It is an array representation of the List implementation class.

It allows duplicate elements because it implements list interface.

It allows null values.

It supports both heterogeneous and homogeneous elements.

The default initial capacity of ArrayList is 10, the Load factor of ArrayList is 100%.

**Graphical user interface, text, application, table

Description automatically generated**

Refer Code Snippet: ArrayListExample1\_Class

As demonstrated in ArrayListExample1\_Class, ArrayList class is very unsafe as it allows different types of data.

We have to generalize it using generics to have safe or same type of data.

Refer Code Snippet: ArrayListExample2\_Class

As demonstrated in ArrayListExample2\_Class, ArrayList<Integer> class is generalized to accept only integer, this is safer as the user is not allowed to provide other than integers.

Refer Code Snippet: ArrayListExample3\_Class

As demonstrated in ArrayListExample3\_Class, ArrayList<employee> class is generalized to accept only employee type.