JAVA

Interface in Java

· The interface in Java is a mechanism to achieve abstraction. There can be only abstract methods in the Java interface, not method body. It is used to achieve abstraction and multiple inheritance in Java.

Interface Names

· Comparable

· Cloneable

· Serializable

· Runnable

· Iterable

· Collection

· List

· Set

· Map

· Queue

· Deque

· Comparator

· Iterator

· AutoCloseable

· Observable

· Callable

Collection

· Collection is a group of objects, typically referred to as elements, stored together and managed as a single entity. The Collection interface is part of the Java Collections Framework and serves as the root interface for most data structures in Java.

Collection Interface

· The Collection interface defines basic methods to manipulate collections of objects, such as adding, removing, querying, and iterating over elements. Collections can store objects in various ways, depending on the specific type of collection (e.g., list, set, queue).

Collection Class

· Collection class does not directly exist. The Collection Framework provides various interfaces and classes that help you store and manipulate data, but there is no specific collection class; instead, it is an interface that defines a group of objects and basic operations on those objects.

· However, there are concrete collection classes that implement the Collection interface or other related interfaces, such as List, Set, Queue, and so on. These are part of the Java Collections Framework and provide more specialized functionality for storing and managing groups of objects.

Map

· A Map allows you to store and retrieve values based on a unique key. Unlike other collections like lists or sets, which store individual elements, a map associates each key with a value.

Key Concepts of Map

· Key-Value Pair: A Map stores elements in pairs, where each key is mapped to a value.

· Uniqueness of Keys: Each key in a map is unique, but multiple keys can have the same value.

· Common Implementations: Java provides several implementations of the Map interface, such as HashMap, TreeMap, LinkedHashMap, etc.

Key Differences Between Map and Collection

- **No Duplicate Keys in `Map`**: While collections like `Set` prevent duplicate elements, maps prevent duplicate keys but allow duplicate values. This behavior differs from standard collections, which are focused on individual elements.
- **Mapping**: A `Map` does not have a concept of ordering elements in a simple sequence or group. Instead, it defines relationships between keys and their associated values, and it

supports operations like `put(key, value)`, `get(key)`, `remove(key)`, and `containsKey(key)`, which are specific to key-value mapping.

- **Iterators**: Collections like `Set` and `List` support iteration over their elements directly, but iterating over a `Map` involves iterating over either the **keys**, **values**, or **key-value pairs** (entries). This is handled with specialized iterators: `keySet()`, `values()`, and `entrySet()`.

Queue

· Queue is a data structure that follows the First-In-First-Out (FIFO) principle.

· A queue is typically used to model real-world scenarios such as a line at a ticket counter, job scheduling, or managing requests in a web server.

· The Queue interface extends the Collection interface and provides methods for adding, removing, and inspecting elements in the queue.

Priority Queue:

· Priority Queue is an implementation that orders the elements based on their natural ordering or by a comparator provided at queue creation.

· It is not a typical FIFO queue. The elements are removed in priority order, not insertion order.

Array Deque:

· Array Deque is a resizable array implementation of the Deque interface, which can be used as both a stack and a queue.

· It provides a more efficient implementation than LinkedList for a queue and supports FIFO operations.

Double Ended Queue

· Double-Ended Queue (Deque) is a type of queue that allows elements to be added or removed from both ends—both the front and the back.

· This flexibility allows the Deque to function as both a FIFO (First-In-First-Out) queue and a LIFO (Last-In-First-Out) stack, depending on how elements are inserted and removed.

Vector

· Vector is a growable array of objects that implements the List interface. It is part of the java.util package and is like an Array List but with some key differences in terms of synchronization and growth policy.

Throw and Throws

Aspect throw throws

Purpose Used to explicitly throw an exception Used to declare that a method can throw exceptions

Location Used inside a method or constructor to throw an exception Used in the method signature to declare the possible exceptions

Action Creates and sends an exception up the call stack Specifies which exceptions a method might throw

Syntax throw new ExceptionType("message"); methodName() throws ExceptionType1, ExceptionType2

Responsibility The method or block where the exception is thrown is responsible for handling it The method that calls a method with throws must handle or declare the exception

Dictionary Alternative in Java - HashMap

In programming, a **dictionary** is a data structure that stores key-value pairs, similar to how a real-world dictionary stores words (keys) and their definitions (values). In this context:

- **Key**: A unique identifier used to look up a corresponding value in the dictionary.
- **Value**: The data associated with the key, which can be of any type (e.g., string, integer, object).

Dictionaries are also known by different names in different programming languages, such as HashMap in Java