Name: Gautam K K

Department: Impact Trainee

Task: Java

Date: 22/11/2024

JAVA

Throw & Throws

- Throw is used to manually creating and throwing the exception using this keyword
- Used in block of code or inside the method that throw an exception
- We can either use try-catch block or using throws it can be transferred to the place where this method is called
- Or directly use try-catch block inside the specific method
- throw instantiates and throws an exception.
- throws declares the exception(s) that a method can throw
- It is advisable to handle with try catch block instead of using throws keyword.

  Syntax: { throw new ExceptionType("Error message"); }

  Syntax: public returnType methodName() throws ExceptionType1, ExceptionType2 {

  // Statements

  }

Private Class

- Private class can be created if it is a inner class – that enables that inner class can be accessed from outer class
- If it is private we can't able to initialize the object for it.
- Reason is private class can't able to communicate with other part of the code
- Same is applicable for protected class

Collection & Collections

- **Collection** – is the interface in java Collections Framework that is used to manage the groups of objects and which is the set of class & interfaces to store and manipulate that data of object
- The Collection interface allows you to work with different types of collections – such as
- List – Order of objects
- Set – uniqueness of object
- Queue – like line (FIFO,FILO) Collection interface
- Collection interface because it provides a standard way to handle groups of objects,
- Collection is like a container that store multiple values (objects) - Employee – had – aceid, name, email, phone no,

Collections

- Collections class in java is the utility class which means that class should be have only static methods so that no need of creating instance for that class
- And is meant for general purpose use to help to handle the data
- Some of collections method are sort(), reverse(), shuffle(), max(), min()

Heirachy

- Java.lang.object ----> java.lang.iterable -----> java.util.collection ----> list of class & interfaces
- Iterable interface --> has the single method --> iterator() --> which is used to return the Iterator objecct and moves to the next elemnt in the collection.
- The package itself used to specify that it is general purpose (utility)

Map in Java Collection Framework

- The reason map is not collection is that all the other class that implements the collection interface can store only single objects (individual elements) like list, set
- But in the Map it can hold multiple objects like Key-value Pair, like also be object and
- value also be object
- Map is the subclass of the object class
- Map is the interface that provides methods for mapping keys to values
- -put, get, contains key, remove

HashMap:

• The most used implementation of the Map interface.

• It allows null values and one null key.

• It does not guarantee the order of elements (i.e., no predictable order).

TreeMap:

• A Map implementation that maintains its elements in a sorted order according to the natural ordering of its keys or a specified comparator.

• Does not allow null keys.

LinkedHashMap:

• An implementation of Map that maintains the insertion order of elements.

• It preserves the order in which elements were added to the map

Hashtable:

• An older implementation of the Map interface.

• It is synchronized, which makes it thread-safe but slower than HashMap.

• Does not allow null keys or values.

Thread Safety

In java thread safety is to function correctly when a piece of code is accessed by multiple thread simultaneosly leading to errors like race conditions & data corruption

- Synchornization – when applied to method it doesn't allow multiple threads to access
- Concurrent collections
- Volatile – changes to the variables is visible to all threads

Interfaces in Java

- Comparable
- Runnable
- Callable
- Iterable
- Collection
- List
- Set
- Map
- Queue
- Deque
- Observable
- Stream
- Flyable
- Lockable
- Writable
- Readable
- EventListener
- ActionListener

User defined interfaces should be start with prefix 'I' and end with suffix 'able' or 'ible' & name should be verb

Interface – base for collection

- In Java, interfaces form the foundation of the Collection Framework because they define common behaviors that all collections (like List, Set, and Queue) basic methods like add(), remove(), and size().
- Interfaces provide more reusable, maintainable, and extensible code.

Difference b/w Interface and class Feature

Features	class	Interface
Methods	Can have both abstract and concrete methods.	Can only have abstract methods (until Java 8) or default/static methods (Java 8+)
Feilds	Can have instance fields (state)	Can only have constants (static final fields)
Constructors	Can have constructors	Cannot have constructors
Inheritance	Can extend only one class	Can extend multiple interfaces
Access Modifiers	Can have any access modifier (public, private, protected).	All methods and fields are implicitly public
Usage	Used to define the structure and behavior of objects	Used to define a contract or behavior that classes must implement

Vector:

- In Java, Vector is a class that implements the List interface and represents a growable array of objects.
- It is part of the java.util package and provides a way to store and manipulate a collection of objects.
- Although it was part of the original version of Java, Vector is now considered somewhat obsolete, and newer classes like ArrayList are preferred for most use cases.
- However, Vector is still used in legacy codebases or situations that require thread-safety