Java Backend Development Live-85

lecture-3

Class Agenda

- Exception handling in Java
- Java Collections Framework.
- Important Interfaces and their implementation.
- Practical use of these collections.

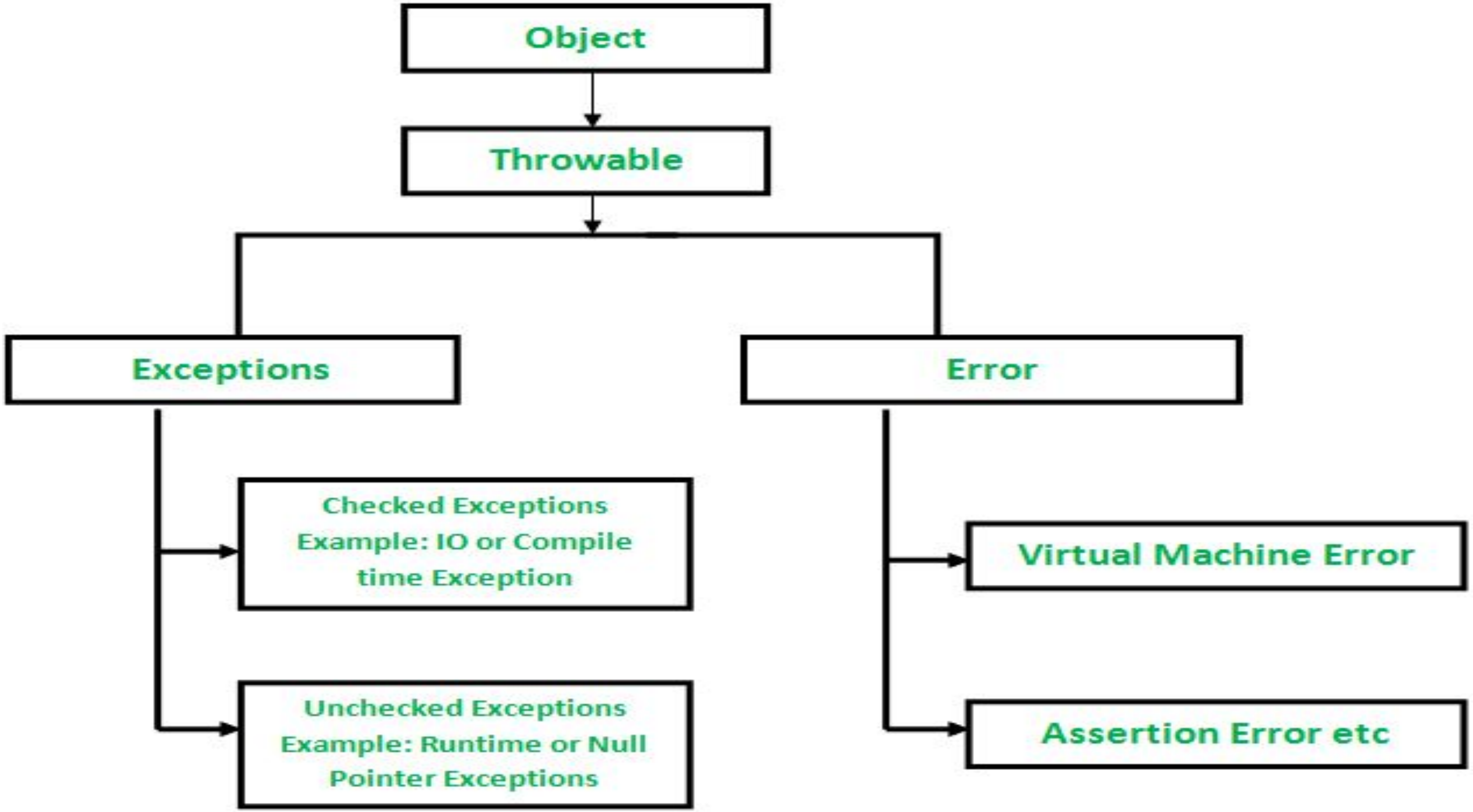
Exception

- unexpected/unwanted events.
- An exception is an event that disrupts the normal flow of the program.
- Leads to system failure.
- Exception handling is the mechanism to handle runtime errors such as Network, SQL, Invalid Data, FileNotFound etc.

Error vs. Exception

Error	**Exception**
Impossible to recover from error	Possible to recover from Exception
Errors are of Unchecked Type	It can be of checked or unchecked
Happen at Run-Time	Can be detected at Compile time & Runtime
Caused by the environment on which the java application is running	Caused by Application

Exception Hierarchy



Object

Throwable

Exceptions

Error

Checked Exceptions
Example: IO or Compile time Exception

Unchecked Exceptions
Example: Runtime or Null Pointer Exceptions

Virtual Machine Error

Assertion Error etc

Checked and Unchecked Exception

Checked Exception	**Unchecked Exception**
An exception that is checked by the compiler at compilation-time.	An exception that occurs at the time of execution.
These exceptions cannot simply be ignored, the programmer must handle these exceptions.	These are also called as Runtime Exceptions. Runtime exceptions are ignored at the time of compilation.
IOException, etc	NullPointerException, ArrayOurOfBoundException

Exception Handling Methods

Try: try block enclosed the code that can throw exception.

Catch: catch block provide exception handling code.

Multiple Catch: To handle multiple exceptions

Finally: finally block is always executed. It can be used to release resource acquired in try block.

Throw: keyword used to throw exceptions.

Throws: used to declare exceptions, used with method signature.

User defined exception

- Built in exception: already available in JAVA

- We can define our own exception like ProductNotFoundException, BadRequestException, etc.

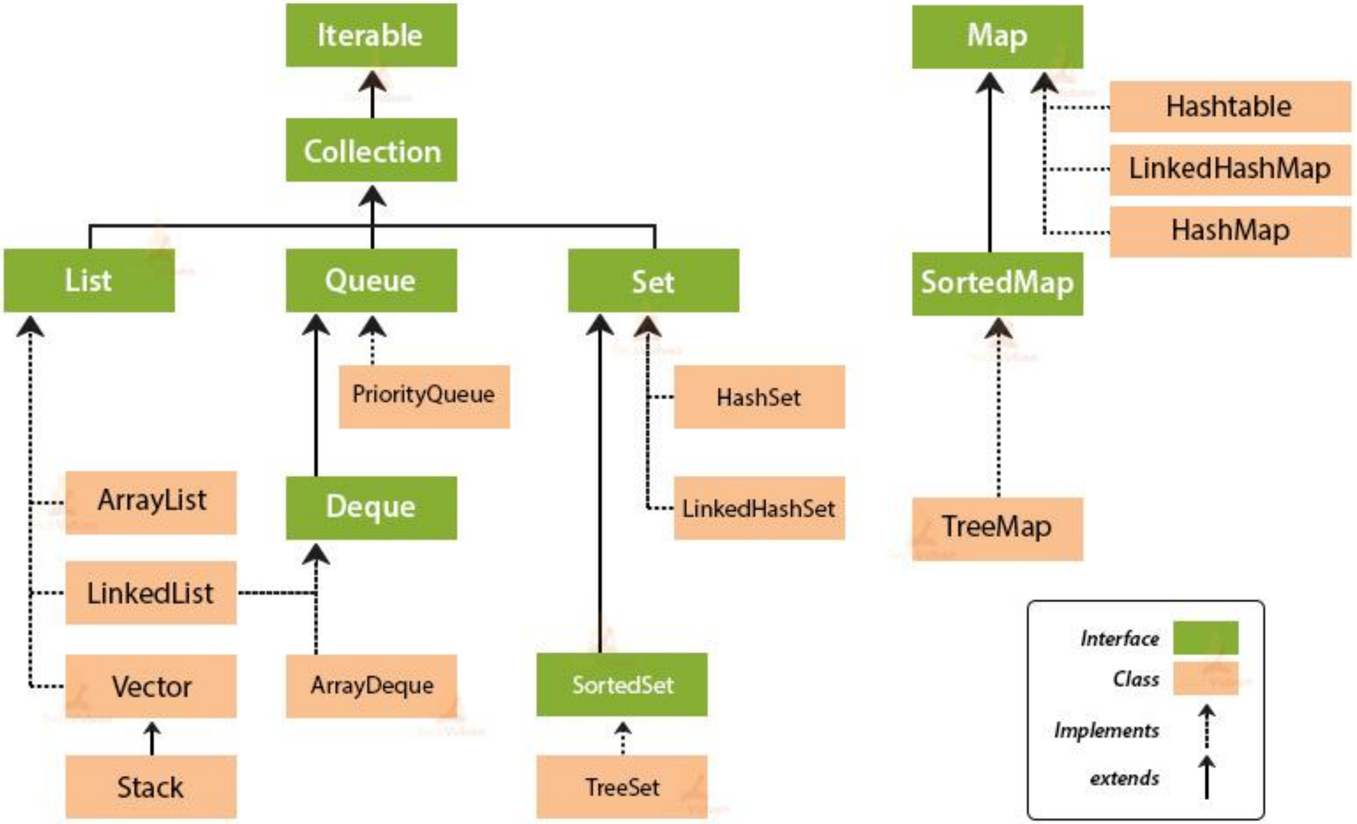
Java Collections Framework

- The Java language API provides many of the data structures from this framework for you.
- It defines a “collection” as “an object that represents a Group of elements (references to objects) It is not specified whether they are Ordered / not ordered Duplicated / not duplicated”.
- It defines a collections framework as “**a unified architecture for representing and manipulating collections,** allowing them to be manipulated independent of the details of their representation.”

Why Java Collection Framework ?

- Provides useful data structures and algorithms.
- Decreases extra effort required to learn, use, and design new API’s
- Supports reusability of standard data structures and algorithms

Collection Framework Hierarchy in Java



Keyword Analyzer Code

Develop keyword analyzer code with following basic features. This code can be used in any App providing search functionality based on keyword.

- Record keywords.
- Return list of all the recorded keywords.

List

A List is an ordered Collection of elements which may contain duplicates. It is an interface that extends the Collection interface. Lists are further classified into the following:

- ArrayList
- LinkedList
- Vectors

List implementations

ArrayList	**LinkedList**
Random Access: get(n) Constant time O(1)	Random Access: get(n) Linear time O(n)
Insert (beginning) and delete while iterating Linear time O(n)	Insert (beginning) and delete while iterating Constant time O(1)

Vectors: Similar to ArrayList but these are synchronized.

Practical uses of List

- Listing of product on amazon/flipkart.
- Listing of jobs on naukri.com
- Listing of questions of GeeksForGeeks

Queue

Queue in Java follows a FIFO approach i.e. it orders the elements in First In First Out manner. In a queue, the first element is removed first and last element is removed in the end.

Important methods:

- add() / offer()
- poll()
- peek()

Queue implementations

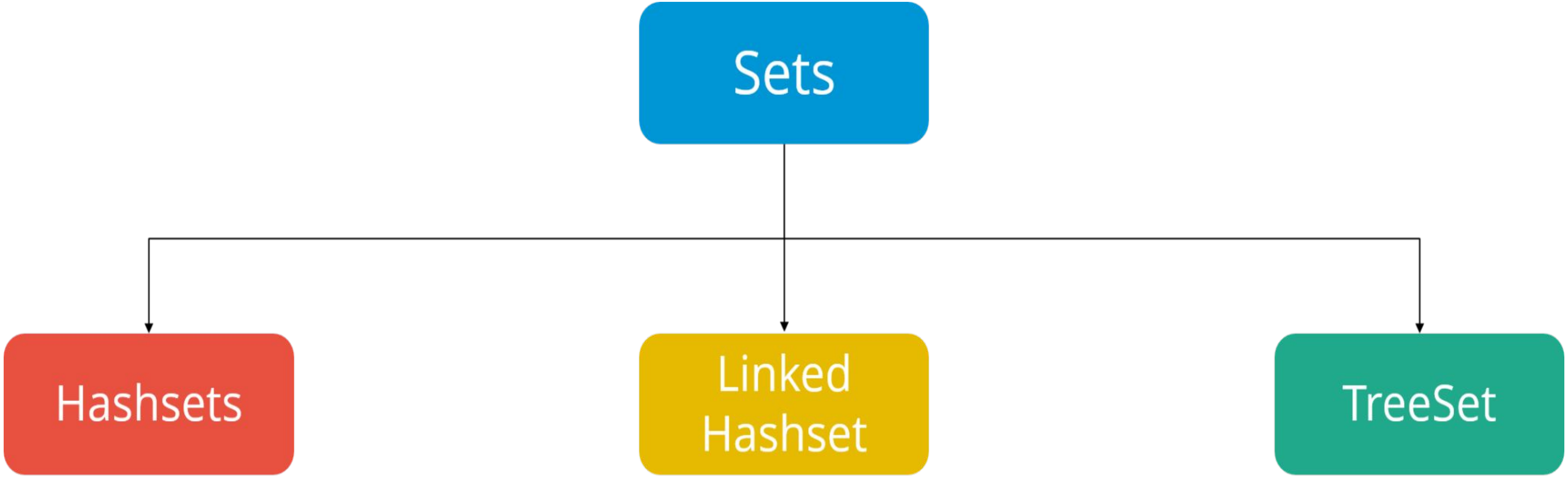
- **LinkedList**
  - head is the first element of the list
  - FIFO: Fist-In-First-Out
- **PriorityQueue**
  - The elements are ordered according to their natural ordering, or by a Comparator provided at the queue construction time

Practical uses of Queue

- Queue on ticket counter, waiting or RAC queue.
- Breadth First Search(BFS): Shortest distance between nodes.
- Level order traversal of tree.

Set Interface

A Set refers to a collection that cannot contain duplicate elements.



equals() and hashcode()

- equals() and hashCode() are **bound together by a joint contract** that specifies if two objects are considered equal using the equals() method, then they must have identical hashcode values.
- To be truly safe:
  - If override equals(), override hashCode()
  - Objects that are equals have to return identical hashcodes

Set implementations

- HashSet implements Set.
  - Hash tables as internal data structure (faster)
- LinkedHashSet extends HashSet
  - Elements are traversed by iterator according to the insertion order.
- TreeSet implements SortedSet.
  - R-B trees as internal data structure (computationally expensive)

Practical uses of Set

Find unique visitors.

Check username already exist.

Map Interface

- Data is stored in key-value pairs and every key is unique. Each key maps to a value hence the name map.
- Designed for the faster lookups.
- Analogous to Set.

Map implementations

- HashMap implements Map
  - No order
- LinkedHashMap extends HashMap
  - Insertion order
- TreeMap implements SortedMap
  - Ascending key order

Practical uses of Map

- Total hits on GeeksForGeeks country wise.
- API Rate Limiting.
- Find frequency of all char in a String.

Arrays and Collections class

Arrays and Collections classes provides several static methods that can be used to perform many tasks directly on arrays and collections.

- Fill an array/collection with a particular value.
- Sort an Arrays/Collections.
- Search in an Arrays/Collections.
- And many more.