Handling Exceptions

- An exception is an unexpected event that disrupts the normal flow of program

Types of Java Exception:

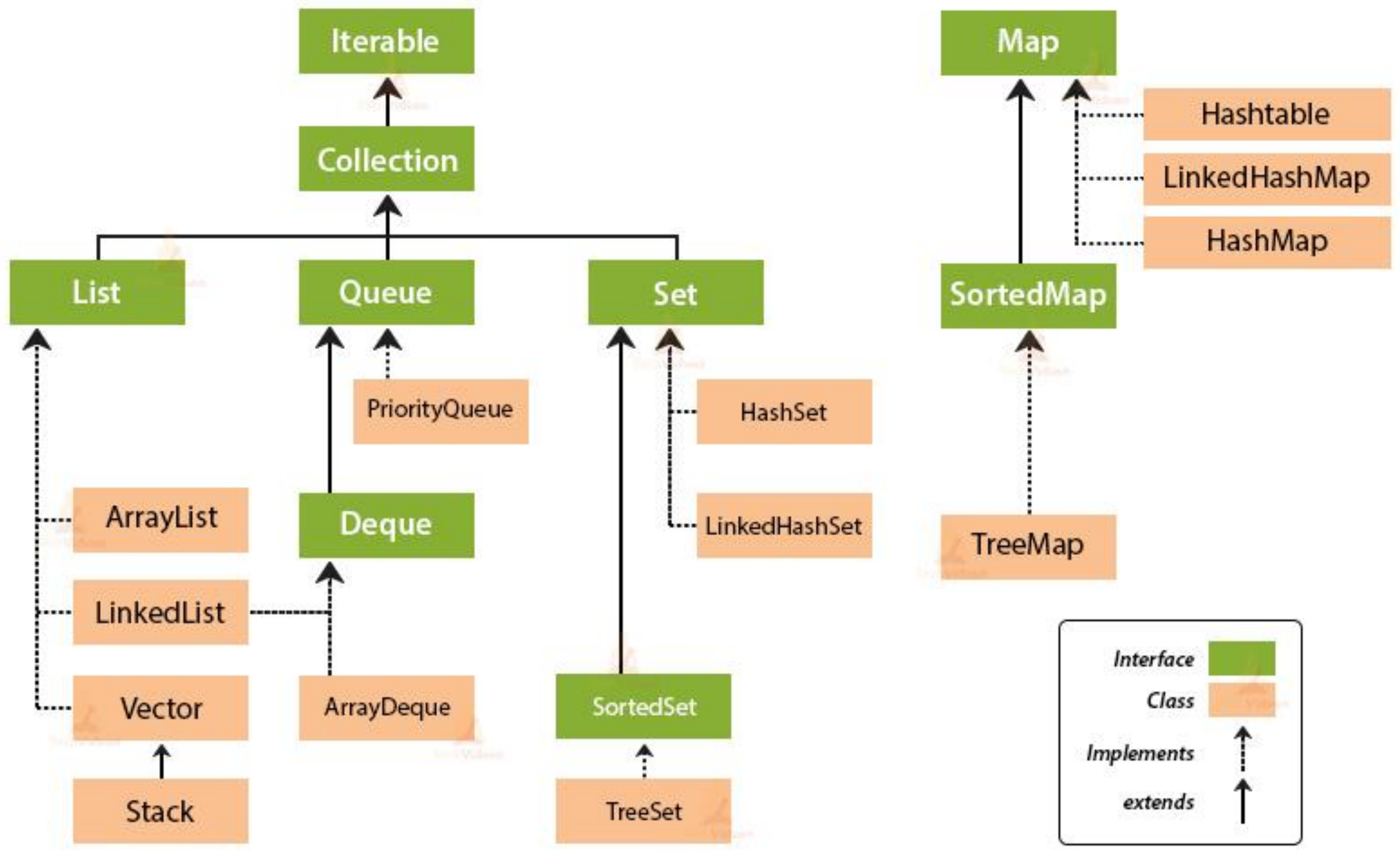
  - Checked Exception – These exceptions are checked at compile time. Ex – IOException, SQLException
  - Unchecked Exception – These exceptions are checked at run time. Ex – ArithmeticException, Null Pointer Exception, ArrayIndexOutOfBoundsException

- Exception Handling can be performed by using the below keywords:
  - try – Code which is expected to return exceptions can be placed inside try block
  - catch – It contains the code which is used to handle the exception
  - finally – Code which must execute whether or not exception occurs is kept in finally block
  - throw – It is used to throw an exception
  - throws – If a method throws an exception it must be declared by using throws keyword

Collections

- Collection is an object which represents a group of objects
- Collections framework is a unified architecture for representing and manipulating collections
- Collections can be used to perform all operations on data like sorting, insertion, manipulation and deletion
- Reduces programming efforts by providing in-built algorithms and data structures
- Increases performance by providing high performance implementations of data structures and algorithms
- Increases code reuse by providing standard interface for collections and algorithms

Collection Framework Hierarchy in Java



Array List

- ArrayList is a dynamic array for storing elements
- Elements can be added and removed easily from a array list
- It can contain duplicate elements
- It maintains the insertion order of the elements
- Methods:
  - add() – To insert specific element in specific position
  - clear() – To remove all elements from the list
  - get() – To fetch element from particular position in the list
  - set() – To change the element in the list
  - remove() – To remove the first occurrence of specified element
  - size() – To return the elements present in the list
  - indexOf() – To get index of first occurrence of an element in the list

QASCRIPT

HashMap

- It stores the data in key value pairs where keys should be unique
- It can contain one null key but multiple null values
- It doesn’t maintain order of elements
- It is non synchronized i.e. multiple threads can access it simultaneously
- It uses a technique called Hashing which converts a large string to small string that represents the same string. A shorter value helps in indexing and faster searches

Syntax:

```
ashMap<String,String> h1 = new HashMap<>();
```

HashMap Methods

- get() – Returns the value associated with specified key
- put() – Used to insert an entry in the map
- isEmpty() – Returns true if map contains no key-value mappings
- Remove() – Used to delete an entry for the specified key
- containsKey() – Returns true if the specified key exists in the map
- containsValue() – Returns true if the specified value exists in the map
- equals() – Used to compare an object with map
- clear() – Used to remove all mappings from the map
- entrySet() – Returns a Set view of mappings contained in the map
- keyset() – Returns a Set view of the keys contained in the map

HashSet

- It implements the Set interface
- Uses Hash Table to store data
- Duplicate values are not allowed
- Insertion order is not maintained
- Null values are allowed
- It is non synchronized

Syntax:

```
HashSet<String> h = new HashSet<String>();
```

QASCRIPT

HashSet Methods

- add() – Used to add the specified element to the set if it is not already present
- clear() – Used to remove all elements from the set
- contains() – Returns true if set contains the specified element
- size() – Returns the number of elements in the set
- isEmpty() – Returns true if the set contains no elements