Monsoon 2019

Collections

Object Oriented Programming

by

Dr. Rajendra Prasath

Indian Institute of Information Technology

Sri City – 517 646, Andhra Pradesh, India

Recap: Objects in JAVA ?

- An entity that has **state** and **behaviour** is known as an object
  - **Examples:** Chair, bike, marker, pen, table, car etc
  - It can be physical or logical
- An object has three characteristics:
  - **State:** represents data (value) of an object
  - **Behaviour:** represents the behaviour (functionality) of an object such as deposit, withdraw and so on
  - **Identity (Internally used):**
    - Signature (unique) of the object
    - Object identity is typically implemented via a unique ID
    - The value of the ID is not visible to the external user
    - But, Internally by JVM to identify each object uniquely

Recap: Inheritance & Constructors

- Constructor of the base with no arguments gets automatically called in the derived class constructor

```
Class Parent {
        Parent() {
                System.out.println("Parent Class Constructor");
        }
}
Class Child extends Parent {
        Child() {
                System.out.println("Child Class Constructor");
        }
}
public class CodeTester {
        public static void main(String[] args) {
                Child child = new Child();
        }
}
```

Java Collections

- A collection is a group of individual objects represented as a single unit.
- Java provides Collection Framework which defines several classes and interfaces to represent a group of objects as a single unit.
- Two main "root" interfaces
  - **Collection Interface** (java.util.Collection)
    - Set
    - List
    - Queue
    - Deque
  - **Map Interface** (java.util.Map)
    - Map
    - SortedMap

Word Count - A Practical Appln

- Given n documents
- How to find the total count of words present in these n documents?
- Can we use the util collection?

Look at 3 documents

d_1- **Darjeeling** is a city and a municipality in the Indian state of West Bengal. It is located in the Lesser Himalayas at an elevation of 6,700 feet

d_2- **Darjeeling** is noted for its tea industry, its views of Kangchenjunga, the world's third-highest mountain, and the **Darjeeling** Himalayan Railway, a UNESCO World Heritage Site

d_3- **Darjeeling** is the headquarters of the **Darjeeling** District which has a partially autonomous status within the state of West Bengal. It is also a tourist destination in India

Unique words and Counts?

d_1

2 the
2 of
2 is
2 in
2 a
1 state
1 municipality
1 located
1 feet
1 elevation
1 city
1 at
1 and
1 an
1 West
1 Lesser
1 It
1 Indian
1 Himalayas
1 Darjeeling
1 Bengal
1 6,700

d_2

2 the
2 its
2 Darjeeling
1 world's
1 views
1 third-highest
1 tea
1 of
1 noted
1 mountain
1 is
1 industry,
1 for
1 and
1 a
1 World
1 UNESCO
1 Site
1 Railway
1 Kangchenjunga
1 Himalayan
1 Heritag

d_3

3 the
2 of
2 is
2 a
2 Darjeeling
1 within
1 which
1 tourist
1 status
1 state
1 partially
1 in
1 headquarters
1 has
1 destination
1 autonomous
1 also
1 West
1 It
1 India
1 District
1 Bengal

Documents – Words / Terms*

✧ How to construct Terms - documents

Doc ID	Terms	# Words
d_1	6,700 (1), Bengal. (1), Darjeeling (1), Himalayas (1), Indian (1), It (1), Lesser (1), West (1), a (2), an (1), and (1), at (1), city (1), elevation (1), feet (1), in (2), is (2), located (1), municipality (1), of (2), state (1), the (2),	22
d_2	Darjeeling (2), Heritage (1), Himalayan (1), Kangchenjunga, (1), Railway, (1), Site (1), UNESCO (1), World (1), a (1), and (1), for (1), industry, (1), is (1), its (2), mountain, (1), noted (1), of (1), tea (1), the (2), third-highest (1), views (1), world's (1),	22
d_3	Bengal. (1), Darjeeling (2), District (1), India (1), It (1), West (1), a (2), also (1), autonomous (1), destination (1), has (1), headquarters (1), in (1), is (2), of (2), partially (1), state (1), status (1), the (3), tourist (1), which (1), within (1),	22

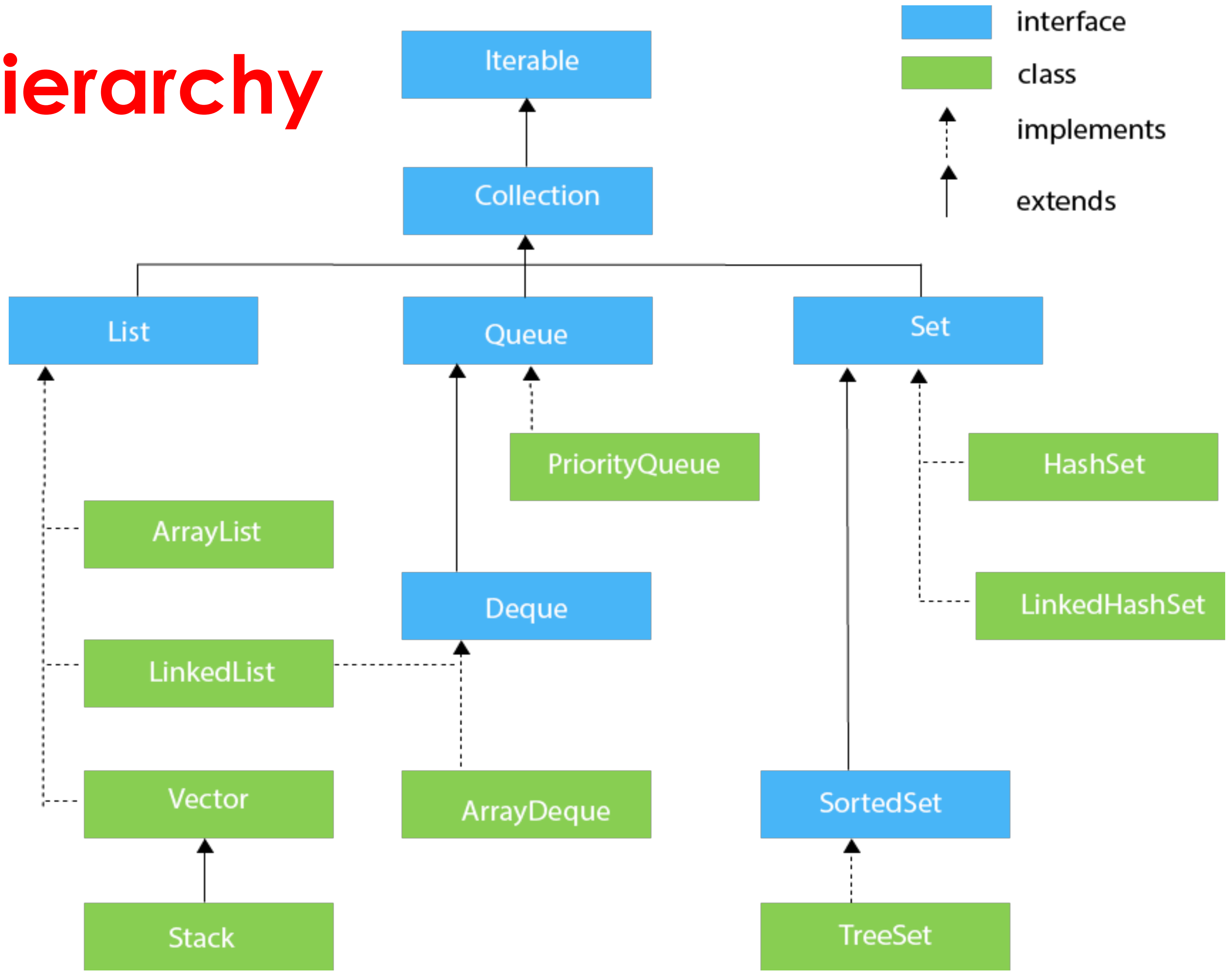
NOTE: "**Words**" and "**Terms**" are interchangeably used throughout the course

Terms - Documents

Terms	d_1	d_2	d_3	. . .	d_n
the	2	2	3	. . .	0
a	2	1	2	. . .	1
Darjeeling	1	2	2	. . .	0
is	2	1	2	. . .	0
of	2	1	2	. . .	0
in	2	0	0	. . .	1
and	1	1	0	. . .	0
Bengal	1	0	1	. . .	0
It	1	0	1	. . .	0
Its	0	2	0	. . .	2
state	1	0	1	. . .	0
West	1	0	1	. . .	1

NOTE: “**Words**” and “**Terms**” are interchangeably used throughout the course

Hierarchy


interface
class
implements
extends
Iterable
Collection
List
Queue
Set
PriorityQueue
HashSet
ArrayList
Deque
LinkedHashSet
LinkedList
Vector
ArrayDeque
SortedSet
Stack
TreeSet

Collections

- **Collection:** Root interface with basic methods like add(), remove(), contains(), isEmpty(), etc
- **Set:** Doesn't allow duplicates. Example implementations of Set interface are HashSet (Hashing based) and TreeSet (balanced BST based).
  - Note that TreeSet implements SortedSet.
- **List:** Can contain duplicates and elements are ordered. Example implementations are LinkedList (linked list based) and ArrayList (dynamic array based)
- **Queue:** Typically order elements in FIFO order except exceptions like PriorityQueue.
- **Deque:** Elements can be inserted and removed at both ends. Allows both LIFO and FIFO.
- **Map:** Contains Key value pairs. Doesn't allow duplicates. Example implementation are HashMap and TreeMap.
  - TreeMap implements SortedMap.

- Difference between **Set** and **Map** interface: in Set, we have **only keys**, whereas in Map, we have **key, value pairs**

Set Interface

- Set Interface in Java is present in java.util package
- It extends the Collection interface
- It represents the unordered set of elements which does not allow us to store the duplicate items. We can store at most one null value in Set
- Set is implemented by HashSet, LinkedHashSet, and TreeSet.
- Set can be instantiated as:

```
Set<data-type> s1 = new HashSet<data-type>();
Set<data-type> s2 = new LinkedHashSet<data-type>();
Set<data-type> s3 = new TreeSet<data-type>();
```

HashSet

- ✧ Hashing is used to store the elements in the HashSet
- ✧ It contains unique items
- ✧ If items are string then they are case-sensitive
- ✧ Example:

```
HashSet<String> setA = new HashSet<String>();
HashSet<Integer> setB = new HashSet<>(); ←
```

Diamond Interface

HashSet (contd)

- Java HashSet class is used to create a collection that uses a hash table for storage
- It inherits the AbstractSet class and implements Set interface
- Important points:
  - HashSet stores the elements by using a mechanism called hashing.
  - HashSet contains unique elements only.
  - HashSet allows one null value.
  - HashSet class is non synchronized.
  - HashSet doesn't maintain the insertion order. Here, elements are inserted on the basis of their hashcode.
  - HashSet is the best approach for search operations.

LinkedHashSet

- ✧ LinkedHashSet class represents the LinkedList implementation of Set Interface
- ✧ It extends the HashSet class and implements Set interface
- ✧ Like HashSet, It also contains unique elements
- ✧ It maintains the insertion order and permits null elements

- ✧ Syntax:

```
LinkedHashSet<String> set=new LinkedHashSet<>();
LinkedHashSet<Float> set=new LinkedHashSet<>();
LinkedHashSet<Integer> set=new LinkedHashSet<>();
```

TreeSet

- Java TreeSet class implements the Set interface that uses a tree for storage
- It inherits AbstractSet class and implements the NavigableSet interface
- The objects of the TreeSet class are stored in ascending order.
- **Important Points:**
  - Java TreeSet class contains unique elements only like HashSet
  - Java TreeSet class access and retrieval times are quiet fast
  - Java TreeSet class doesn't allow null element
  - Java TreeSet class is non synchronized
  - Java TreeSet class maintains ascending order

List

- Set contains unique elements only

- But a list can contain duplicate elements
- List allows insertion in place

- Example:

```
List<String> al = new ArrayList<String>();
al.add(1, "Sachin");
```

ArrayList

- This class uses a dynamic array for storing the elements
- It inherits AbstractList class and implements List interface

- The important points about Java ArrayList class are:
  - Java ArrayList class can contain duplicate elements.
  - Java ArrayList class maintains insertion order.
  - Java ArrayList class is non synchronized.
  - Java ArrayList allows random access because array works at the index basis.

- Manipulation is slow because a lot of shifting needed if any element is removed from the array list

LinkedList

- This class uses a doubly linked list to store the elements
- It provides a linked-list data structure
- It inherits the AbstractList class and implements List and Deque interfaces

- The important points about Java LinkedList are:
  - Java LinkedList class can contain duplicate elements
  - Java LinkedList class maintains insertion order
  - Java LinkedList class is not synchronized
  - In Java LinkedList class, manipulation is fast because no shifting needs to occur
  - Java LinkedList class can be used as a list, stack or queue

ArrayList vs. LinkedList

Array List	Linked List
ArrayList internally uses a dynamic array to store the elements	LinkedList internally uses a doubly linked list to store the elements
Manipulation with ArrayList is slow because it internally uses an array. If any element is removed from the array, all the bits are shifted in memory	Manipulation with LinkedList is faster than ArrayList because it uses a doubly linked list, so no bit shifting is required in memory
An ArrayList class can act as a list only because it implements List only	LinkedList class can act as a list and queue both because it implements List and Deque interfaces
ArrayList is better for storing and accessing data	LinkedList is better for manipulating data

Queue Interface

- ✧ Java Queue interface orders the element in FIFO(First In First Out) manner.
- ✧ In FIFO, first element is removed first and last element is removed at last.

  public interface Queue<E> extends Collection<E>

- ✧ The PriorityQueue class provides the facility of using queue, but it does not order the elements in FIFO manner
- ✧ It inherits AbstractQueue class.

PriorityQueue

- PriorityQueue does not permit null.
- Can not create a PriorityQueue of **Objects that are non-comparable**
- PriorityQueue are **unbound queues**
- The head of this queue is the least element with respect to the specified ordering. If multiple elements are tied for least value, the head is one of those elements — ties are broken arbitrarily.
- The queue retrieval operations poll, remove, peek, and accesses the element at the head of the queue
- It inherits methods from AbstractQueue, AbstractCollection, Collection and Object class

Deque

- Java Deque Interface is a linear collection that supports element insertion and removal at both ends
- Deque is an acronym for "**double ended queue**"

ArrayDeque:

- The ArrayDeque class provides the facility of using deque and resizable-array
- It inherits AbstractCollection class and implements the Deque interface.

Important Points:

- Add or remove elements from both sides
- Null elements are not allowed in the ArrayDeque
- ArrayDeque has no capacity restrictions
- ArrayDeque is faster than LinkedList and Stack

HashMap

- ✧ HashMap is a Map based collection class that is used for storing Key & value pairs, it is denoted as HashMap<Key, Value> or HashMap<K, V>
- ✧ It is not an ordered collection which means it does not return the keys and values in the same order in which they have been inserted into the HashMap
- ✧ It does not sort the stored keys and Values. You must need to import java.util.HashMap or its super class in order to use the HashMap class and methods.
- ✧ A Map does not allow duplicate keys, but you can have duplicate values
- ✧ HashMap and LinkedHashMap allow null keys and values, but TreeMap doesn't allow any null key or value.
- ✧ A Map can not be traversed, so you need to convert it into Set using keySet() or entrySet() method.

HashMap – Points to Remember

- ✧ HashMap class contains values based on the key
- ✧ Java HashMap class contains only unique keys
- ✧ Java HashMap class may have one null key and multiple null values
- ✧ Java HashMap class is non synchronized
- ✧ Java HashMap class maintains no order

Parameters for java.util.HashMap class

K: It is the type of keys maintained by this map

V: It is the type of mapped values

LinkedHashMap

- ✧ LinkedHashMap class is Hashtable and Linked list implementation of the Map interface, with predictable iteration order
- ✧ It inherits HashMap class and implements the Map interface.

Points to Remember:

- ✧ LinkedHashMap contains values based on the key
- ✧ LinkedHashMap contains unique elements
- ✧ LinkedHashMap may have one null key and multiple null values
- ✧ This class is non synchronized
- ✧ LinkedHashMap maintains insertion order

Exercise - 8

- **Apply the concepts of Collections for a specific application:**

  - Various data structures

  - Apply the properties of collection classes

  - You may apply suitable comparator for defining various tasks

  - Explore sorting options and searching options

Assignments / Penalties

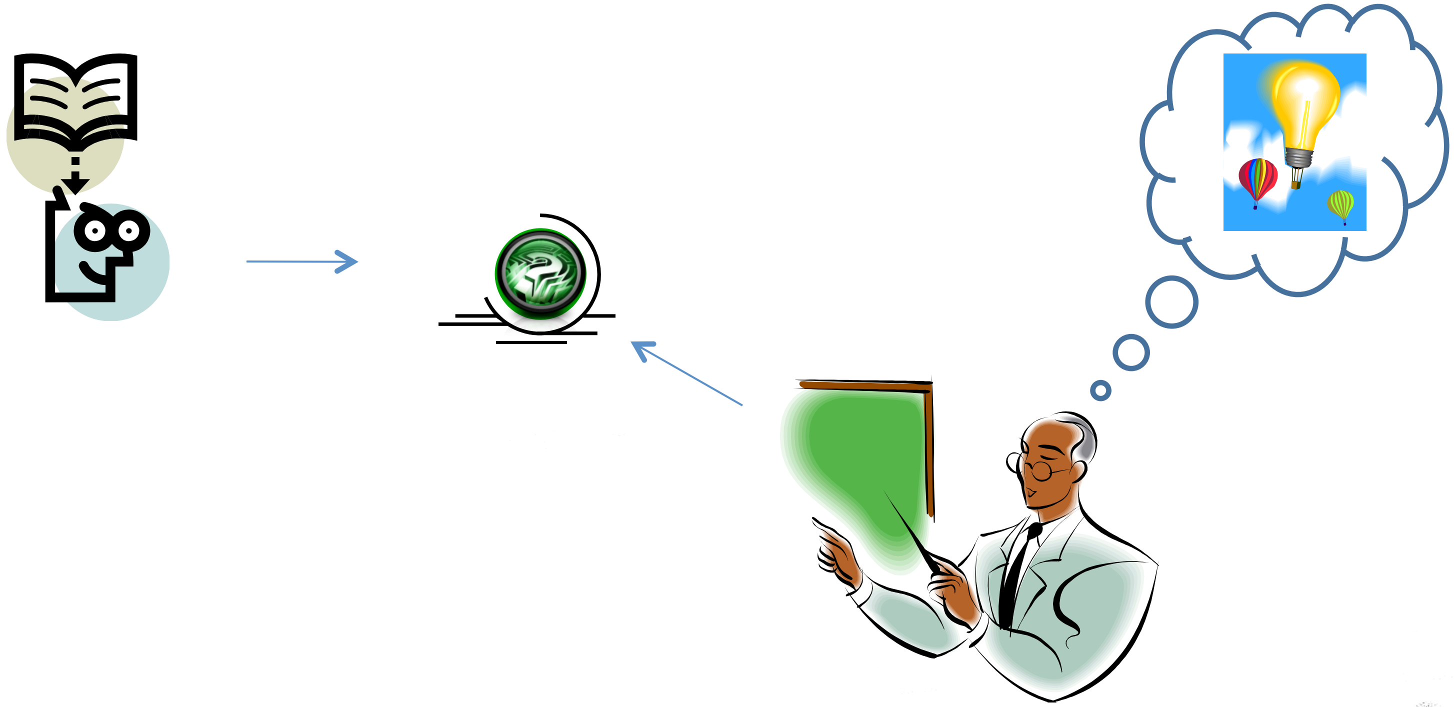
- Every Student is expected to complete the assignments and strictly follow a fair Academic Code of Conduct to avoid severe penalties

- Penalties would be heavy for those who involve in:
  - **Copy and Pasting** the code
  - **Plagiarism** (copied from your neighbor or friend – in this case, both will get "0" marks for that specific take home assignments)
  - If the candidate is **unable to explain his own solution**, it would be considered as a "copied case" !!
  - **Any other unfair means** of completing the assignments

Assistance

- You may post your questions to me at any time
- You may meet me in person on available time or with an appointment
- You may leave me an email any time
(email is the best way to reach me faster)

Thanks …



… Questions ???