Day 11

Searching

```
- It is a process of finding location of element inside collection.
- Searchin Techniques
        1. Linear / Sequential Search
                - We can use it to search element in any collection
                - We can use it to search element in any sorted as well as
unsorted collection.
                - If collection is having limited elements then it is
considered efficient searching algorithm.
                - If collection contain large amount of data then linear
search take more time to search element.
        2. Binary Search
                - It uses divide and conquer technique.
                - It is efficient that linear Search.
                - To use binary search algorithm, collection must be
sorted.

        3. Hashing[ Faster Searching Technique ]
                - If we want to search element in constant time then we
should use hashing.
                - Hashing technique is based on hashcode.
                - Hashcode is logical integer number that can be generated
by processing state of the object.
```

```java
//Hash Function / Method
private static int getHashcode(int data)
{
        int result = 1;
        final int PRIME = 31;
        result = result * data + PRIME * data;
        return result;
}
public static void main(String[] args)
{
        int data = 125;
        int hashcode = Program.getHashcode( data );
        System.out.println(data +"      "+hashcode );
}
```

```
- To generate hashcode, we should use hash function/method.
- If state of object is same then we will get same hashcode.
```

```
- Hashcode is required to generate index( slot ).
- By processing hashcode of two different object if we get same slot then
it is called collision.
- To avoid collision we should use collision resolution techniques
        1. Seperate Chaining( Open Hashing )
        2. Open Addressing( Closed Hashing )
                - Linear Probing
                - Qudratic Probing
                - Double Hashing / Rehashing
- In Seperate chaining, instead of creating array of element, we should
create array of collection( LinkedList, Tree ).
- Collection maintained per slot is called bucket.
- In java, in hashcode based collection, if we want to manage elements of
non final type then non final type should override equals and hashcode
method.
```

HashSet

```
- It is a non final and concrete class declared in java.util package.
- It's implementation is based on Hashtable.
- It is Set collection hence do not permot duplicate element but it
permits null element.
- It is unordered collection.
- It is unsynchronized collection. Using "Collections.synchronizedSet()"
method we can make it synchronized.
   Set s = Collections.synchronizedSet(new HashSet(...));
- It is introduced in jdk1.2
- If we want to manage elements of non final type in HashSet then non
final type should override equals and hashcode method.
- Instantiation:
        Set<Employee> empLisyt = new HashSet( );
```

> What is the difference between HashSet and TreeSet?

LinkedHashSet

```
- It is sub class of HashSet class
- It's implementation is based on Hashtable and linked list.
- It is ordered collection.
- It is unsynchronized collection. Using "Collections.synchronizedSet()"
method we can make it synchronized.
Set s = Collections.synchronizedSet(new LinkedHashSet(...));
- It is introduced in jdk 1.4.
- - If we want to manage elements of non final type in LinkedHashSet then
non final type should override equals and hashcode method.
```

> Difference beween HashSet and LinkedHashSet

Dictionary<K,V>

```
- It is abstract class declared in java.util package.
- It is introduced in jdk1.0
- If we want to manage data in key/value pair format then we should use
Dictionary instance.
- In dictionary instance we can insert null key and null value.
- NOTE: This class is obsolete. New implementations should implement the
Map interface, rather than extending this class.
- Methods of Dictionary class
        1. public abstract boolean isEmpty();
        2. public abstract V put(K key, V value);
        3. public abstract V get(Object key);
        4. public abstract V remove(Object key)
        5. public abstract Enumeration<K> keys()
        6. public abstract Enumeration<V> elements()
        7. public abstract int size();
- Hashtable is a sub class of Dictionary<K,V> class
- Instantiation
        Dictionary<Integer,String> d = new Hashtable<>( );
```

Map<K,V>

```
- It is part of collection framework but it doesn't extends Collection
interface.
- Map<K,V> interface is declared in java.util package.
- This interface takes the place of the Dictionary class, which was a
totally abstract class rather than an interface.
- Methods Of Map<K, V> interface
        1. boolean isEmpty()
        2. V put(K key, V value)
        3. void putAll(Map<? extends K,? extends V> m)
        4. boolean containsKey(Object key)
        5. boolean containsValue(Object value)
        6. V get(Object key)
        7. V remove(Object key)
        8. void clear()
        9. int size()
        10. Set<K> keySet()
        11. Collection<V> values()
        12. Set<Map.Entry<K,V>> entrySet()
```

Map.Entry<K,V>

```
- It is nested interface of Map<K,V> interface.
- Entry instance represents Key/Value pair.
- Abstarct methods of Map.Entry interface
       1. K getKey()
       2. V getValue()
       3. V setValue(V value)
- Map is collection of entries where each entry contain key/value pair.
```

Hashtable

```
- It is sub class of Dictionary class and it is also implements
java.util.Map<K,V> interface.
- It stores data in key/value pair format.
- In Hashtable, we can not insert duplicate key but we can insert
duplicate value.
- In Hashtable, key and value can not be null.
- It is synchronized collection.
- It is introduced in jdk 1.0.
- If we want to use instance of non final type as key then it must
override equals and hashcode method.
```

> What is the difference between HashSet and Hashtable?

HashMap<K,V>

```
- It is map collection.
- It stores data in key/value pair format.
- It can not contain duplicate keys but it contain duplicate values.
- It is unsynchronized collection. Using "Collections.synchronizedMap()"
method we can make it synchronized.
   Map m = Collections.synchronizedMap(new HashMap(...));
- It can contain null key and null value.
- It is introduced in jdk 1.2
- If we want to use instance of non final type as key then it must
override equals and hashcode method.
```

> What is the difference between Hashtable and HashMap?

LinkedHashMap<K,V>

```
- It is sub class of HashMap<K,V>
- Its implementation is based on Hashtable and linked list
- It is unsynchronized collection.Using "Collections.synchronizedMap()"
method we can make it synchronized.
```

```
   Map m = Collections.synchronizedMap(new LinkedHashMap(...));
- In LinkedHashMap<K,V> we can not insert duplicate key but we can insert
duplicate value.
- It can contain null key and value.
- It is ordered collection.
- It is introduced in jdk1.4
- If we want to use instance of non final type as key then it must
override equals and hashcode method.
```

> What is the difference between HashMap and LinkedHashMap?

TreeMap

```
- It is map collection.
- Its implementation is based on RED BLACk Tree.
- It can not duplicate keys but it can contain duplicate values.
- In TreeMap, key can not be null but value can be null.
- It is Sorted Collection.
- It is unsynchronized collection. Using
"Collections.synchronizedSortedMap()" method we can make it synchronized.
SortedMap m = Collections.synchronizedSortedMap(new TreeMap(...));
- It is introduced in jdk 1.2.
- If we want to use instance of non final type as a key then its type must
implement Comparable interface.
```

> What is the difference between HashMap<K,V> and TreeMap<K,V>?

File Handling

```
- File is a container that is used to save data permenantly on HDD.
- File is non java resource.
- Types of File
        1. Binary File
        2. Text File.
```

Text File:

```
- e.g. .txt, .doc, .docx, .rtf, .java, .html etc.
- We can read text file using any text editor
- Text file require more processing hence it is slower in performance.
- If we want to save data in human readable form then we should create
text file.
```

Binary File:

```
- e.g .mp3/.mp4, .jpg/.jpeg/.gif, .class etc.
- To read binary file we should use specific program.
- Binary file require less processing hence it is faster in performance.
- If we dont want to save data in human readable form then we should
create binary file.
- It is an abstraction(instance) that is used to produce(write) and
consume(read) information from source to destination.
- If we want to manipulate files in java then we should use types declared
in java.io package.
```

Interfaces declared in java.io Package

```
- Flushable
- Closeable
- FilenameFilter
- DataInput
- DataOutput
- ObjectInput
- ObjectOutput
- Serializable
```

Classes declared in java.io Package

```
- Console
- File
- Following are stream classes which are required to manipulate binary
file.
        - InputStream
        - OutputStream
        - FileInputStream
        - FileOutputStream
        - BufferedInputStream
        - BufferedOutputStream
        - DataInputStream
        - DataOutputStream
        - ObjectInputStream
        - ObjectOutputStream
        - PrintStream
- Following are stream classes which are required to manipulate text file.
        - Reader
        - Writer
        - FileReader
        - FileWriter
```

```
        - BufferedReader
        - BufferedWriter
        - InputStreamReader
        - OutputStreamWriter
        - PrintWriter
- If we want to manipulate binary files then we should use InputStream,
OutputStream and their sub classes.
- If we want to manipulate text files then we should use Reader, Writer
and their sub classes.
```

File class

```
- It is sub class of java.lang.Object that implements following
interfaces:
        1. Serializable
        2. Comparable<File>

- Instantiation :
        String pathname = "/usr/sandeep/File.txt";
        File file = new File( pathname );

- Instance of java.io.File class is not a file rather it represents OS
file,directory and drive.
- java.io.File class is used:
        1. To create empty file and directory
        2. To remove file and directory
        3. To read metadata of file, directory and drive.
- Methods of File class
        - public boolean createNewFile()throws IOException
        - public boolean mkdir()
        - public boolean delete()
        - public boolean exists()
        -
```