Java Collection Framework

APRIL

S	M	T	W	T	F	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

MAY

S	M	T	W	T	F	S
31					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

17th week
115-250

25 Saturday

Limitations of Object Arrays

(i) Fixed Size — created an array with a fixed size, increment or decrement not possible

(ii) Homogeneous data elements — (Same type) Incompatible types

(iii) Not built on any data structure → Readymade methods are not present.

Advantages of Collections over Arrays :—

(i) Growable in nature — Incremented / decreased

(ii) Both Homogeneous / Heterogenous

(iii) Based on Data Structure elements that are ready made elements.

Limitation of Collection Objects :— (over array)

26 Sunday

(i) Performance better than arrays.

(ii) Arrays are prefered over collections when performance is the main parameter.

Definition of Collection — It is a group of individual objects as a single entity.

These Collection frameworks are either class or interface.

Notes

18th week
117-248

April

27
Monday

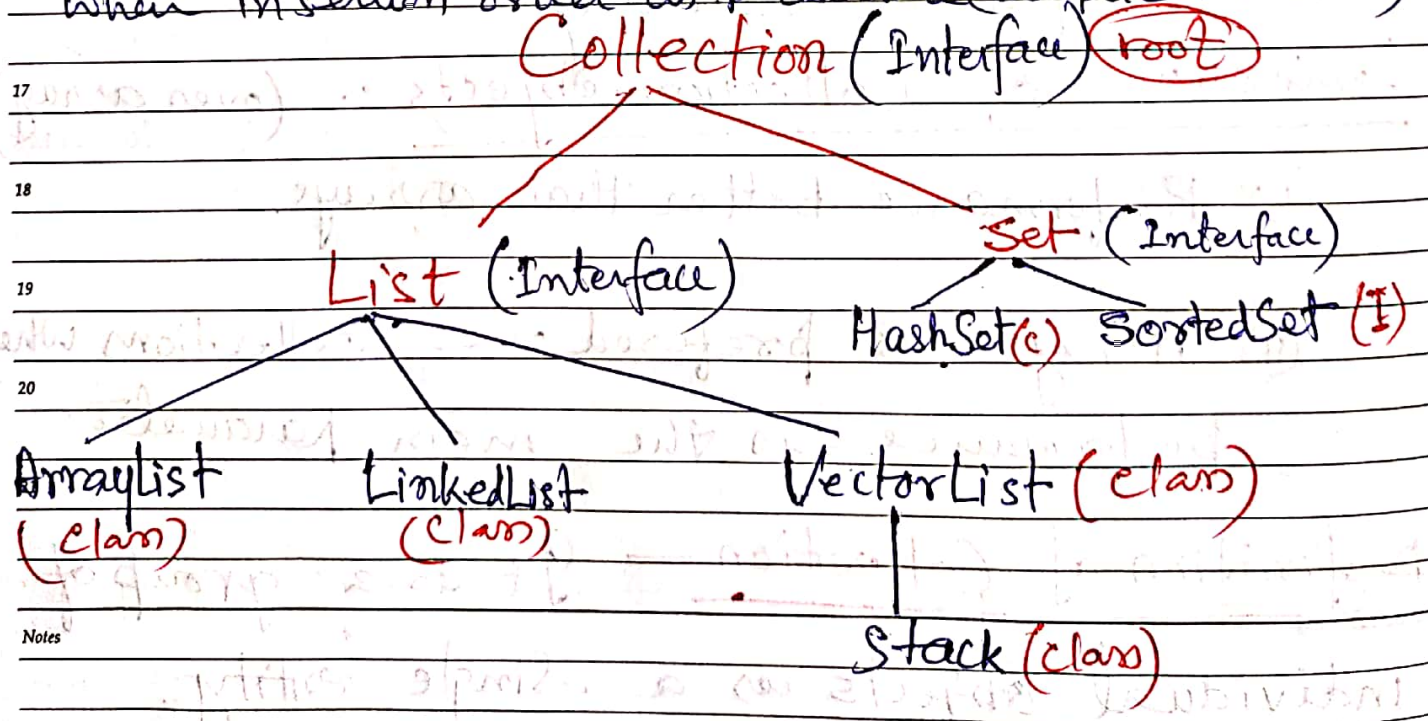
Collection framework defines several classes and interfaces which can be used to represent a group of objects as a single entity.

Elements of Collection Framework :-

(i) Collection — (Interface) Root of all collection frameworks
(ii) List

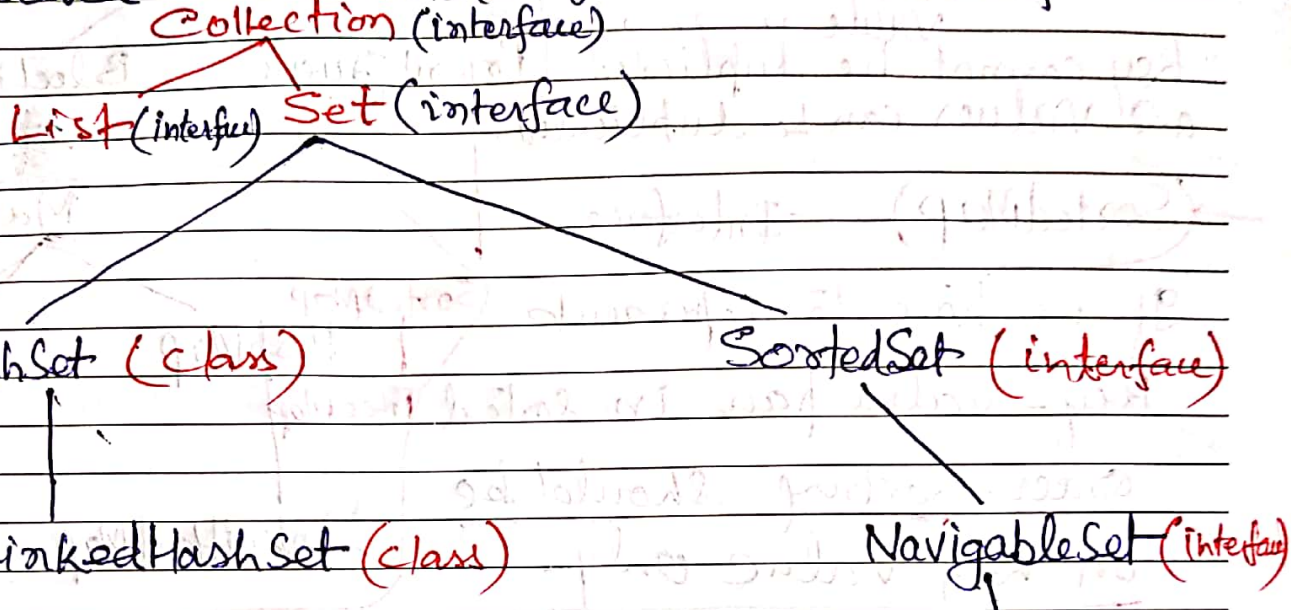
Collection is used when we need to represent group of objects as a single entity. It defines most of the methods defined for Collection framework.

List - It is child of Collection interface. Used when insertion order is preserved (duplicates also).



Notes

→ **Set** — It is an interface. In set duplicate values are not allowed. Insertion Order is not preserved. It is child of Collection interface.



→ **SortedSet** — Interface, child of set interface. Used for ~~A~~ according to some sorting order.

→ **NavigableSet** — child of SortedSet, basically used to provide several methods for navigation process. It is also an interface introduced in JDK 1.6.

→ **Ques** — It is an interface directly derived from Collection interface.
It is basically used for storing the objects prior to processing.

Notes

vksu.phy@gmail.com

29
Wednesday

April

Collection (I)

Queue (I)

PriorityQueue

BlockingQueue

→ Map → Interface. Not derived from Collection interface. Key-Value pairs can be stored. Key is also an Object Value " " " Key cannot be duplicated and values can be duplicated.

Map (Interface)

SortedMap

HashMap

HashTable

TreeMap

LinkedHashMap

→ SortedMap → Interface

If we have to represent key-value pair in sorted order. Sorting should be on key value only.

→ NavigableMap (I) → Child interface of SortedMap. Navigable process.

Map (i)
SortedMap
NavigableMap

Summary →

Collections (I)

List (i)

Set (i)

Queue (i)

ArrayList

VectorList

Stack

HashSet

LinkedHashSet

SortedSet

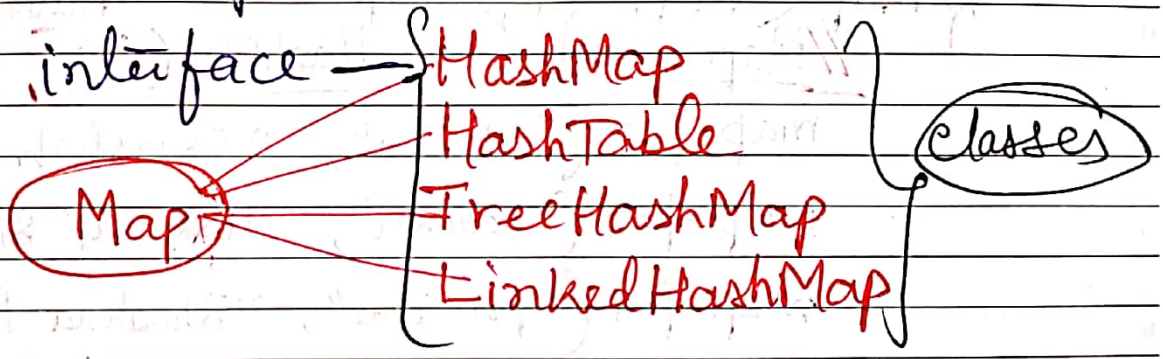
NavigableSet

PriorityQueue

BlockingQueue

If you have a question — What is Collection classes in Java. Then draw the entire hierarchy then explain in ~~fast~~ short.

→ HashMap — Since, Map is an interface and there are four classes implementing the Map interface —



We know that; we can not make a class by extending an interface so we have to extend these four classes according to requirements in our program. We can do it by using java.util package and making object as :—

Notes Map interface is used to store a pair of data which is — combination of [key — value]

01 May
Friday

→ We can store a combination of Key-value data by using put() method

→ We can get a data from a Key-value combination by using get() method.

Example:- import java.util.Map; import java.util.HashMap;
public class mapdemo{

public static void main(String args[])
{ Map<String, String> map = new HashMap<>();
//Map map = new HashMap();
map.put("Naiue", "Shershah College");
map.put("teacher", "Bharat Singh");
map.put("principal", "Krishna Prasad");
System.out.println(map);
}
}

//Output not in the sequence →

Now, anything, we want to print from the map:- we can use the function get():—

System.out.println(map.get("principal"));
System.out.println(map.get("teacher"));

prints:- Krishna Prasad
prints:- Bharat Singh

Notes

Similarly, we have a function keySet() by using this in for loop, we get all the values stored in entire keySet of a map as :—

```
Set<String> keys = map.keySet();
for(String key : keys)
{
    System.out.println(key + " " + map.get(key));
}
```

This will print all key & value in proper order.

Hash Function → It is also known as M.D-5 (Message Digest-5). It is a compression function. It is mathematical function that converts a numerical input-value into another compressed numerical value.

→ Its output is always of fixed length.

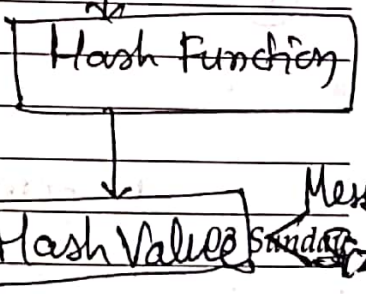
Features of Hash Function —

→ Fixed length output

→ Compression function

→ Digest (Smaller representation of larger data)

"Any Message of any length"



Properties — (1) M → H (Easy) (easy) H (Hash Value) → Message (very hard)

(i) Message to Hash Value Conversion is easy

(ii) Hash Value to Message Conversion should be difficult or impossible.

(iii) Two Hash Value is not same (anyway).

M.D-5 :— Developed by Ron Rivest. It is very fast and produces 128-bit message digests.

04 May
Monday

Working of MD-5 :-

(I) Original Message Padding.

Padding is done such that total length is 64 bit less than exact multiple of 512.

Eg:- Suppose we have original message of 1000 bits.

Now 512 multiple should be

$512 \times 2 = 1024$

$512 \times 3 = 1536$

$- 64$

1472

Hence, 1000 bits + 472 bits = 1472 which is 64 bit less than exact multiple of 512.

(II) Append the original length before padding.

Original message	Padding	length

Before Padding we have (in above case)

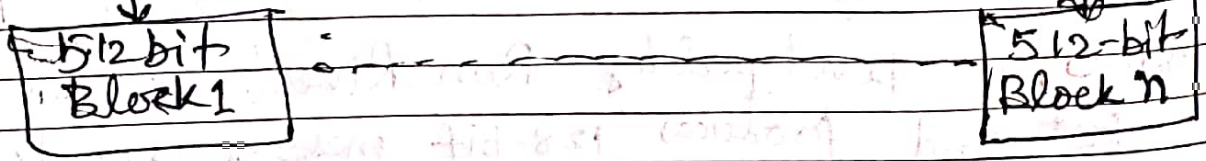
1000

Now calculate modulo 64] 1000 length $\bmod 2^{64}$

This will be the exact multiple of 512

(III) Dividing it 512-bit blocks.

Original message	Padding	length



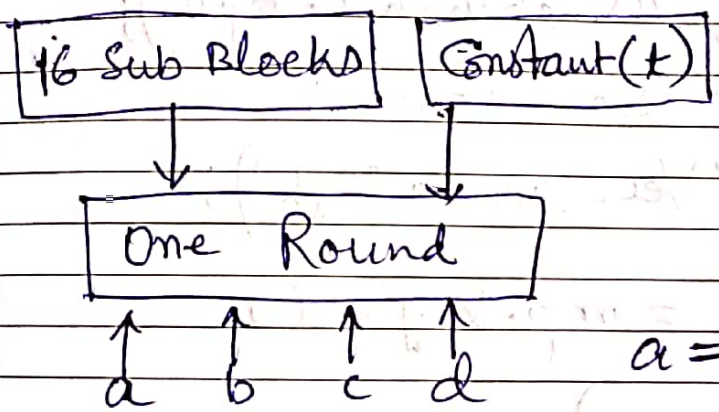
Notes

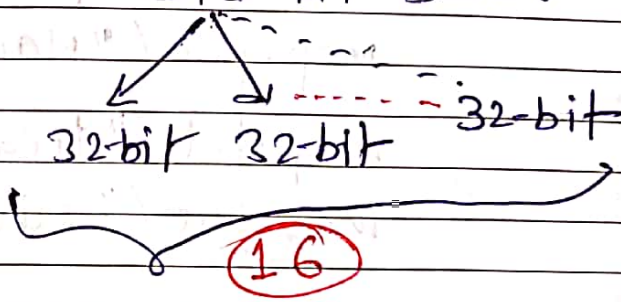
(iv) Initialize a chaining variables.
(32-bit, A, B, C and D)

(v) Process Blocks → Copy four chaining variables into corresponding variables
{A=a, B=b, C=c, D=d}

→ Divide 512-bit block into 16 (32-bit blocks)

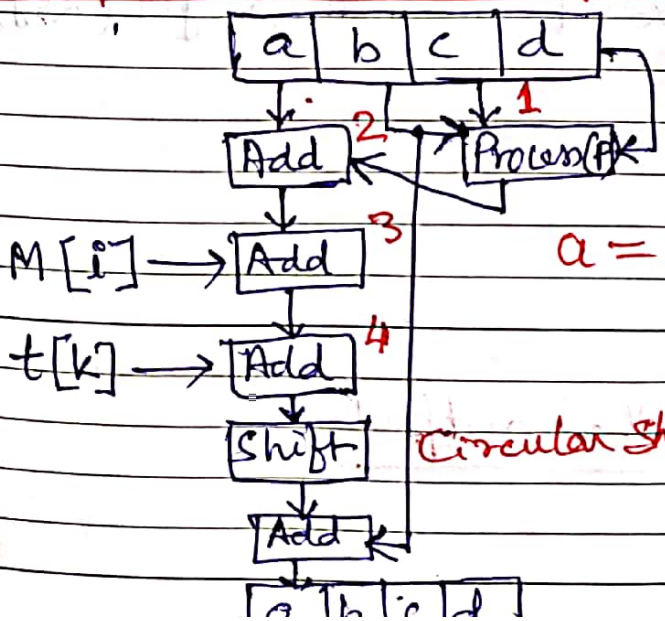
→ Four Rounds





$a = b + ((a + \text{Process}, P(b, c, d) + M[i] + T[k])) << \text{Shift}.$

Complete Process Diagram of MD5 Operation:-



$a = b + (\text{Process}(b, c, d) + M[i] + t[k] << s)$

Circular Shift (s)

Notes

vksu.ohu@gmail.com

06 Wednesday May

19th week 126-239

Using HashTable in Map Example

Synchronized but HashMap is not

We can implement Map using HashTable

Example -

```
import java.util.Map;
import java.util.Hashtable;
import java.util.Set;

public class mapdemo2
{
    public static void main(String args[])
    {
        Map<String, String> map = new Hashtable<>();
        map.put("Name", "Shershah College");
        map.put("location", "Sasaram");
        map.put("univ", "Veer kunwar Singh");

        Set<String> keys = map.keySet();
        for(String key : keys)
        {
            System.out.println(key+" "+map.get(key));
        }
    }
}
```

Compile & see the output