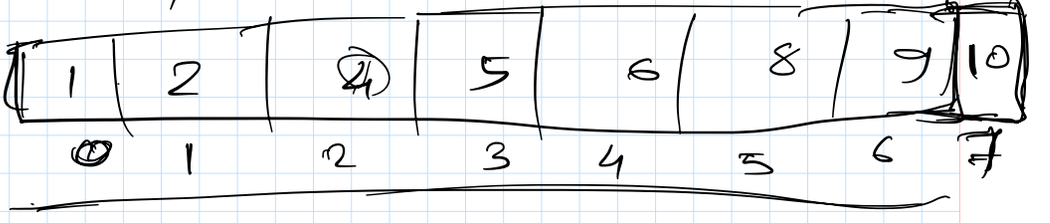
# Introduction to Data Structures & Collections & Iterable.

1) Array
2) Stack
3) Queue
4) Map
5) Heap → Advance - out of scope of this training
6) Set
7) List - LinkedList
8) Tree
9) Graph
] Advance. - out of scope of this training

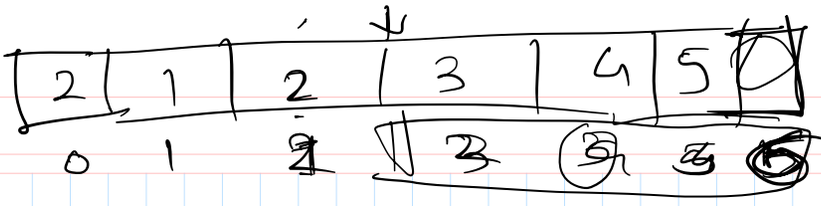
Linear DS.
1) Array
2) List
3) Map
4) Stack
5) Set

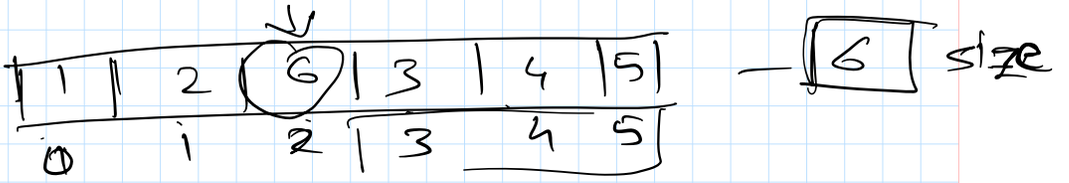
1) Array : Fixed sized, of fixed Type.

int[] a =




int [ ] number = new int [ 7 ] ;



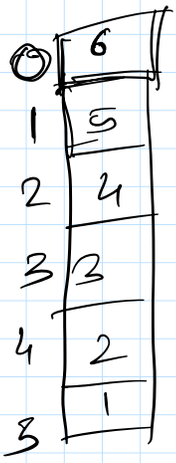

* LinkedList

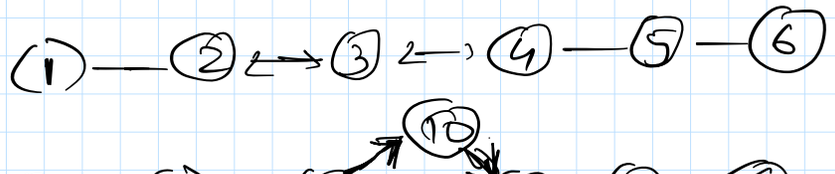
ArrayList — Dynamic, no need define size

LinkedList -

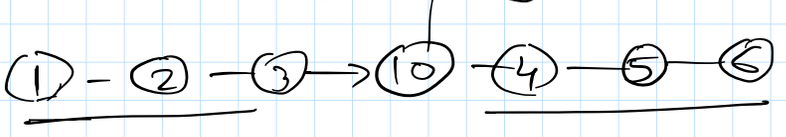
add, remove Items,

List, 1000, 10000,

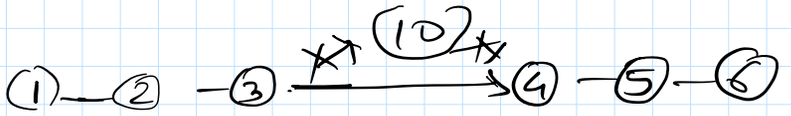







remove 10




Node.

add & removing

(1) Array

(2) LinkedList ⟶ H.W — add remove.

(3) ArrayList ⟶ H.W. — add, remove

(4) Map : <key, value> pair ; H.W.

Problem statement for map below.

```
class Pair {
pubic int rollNo;
Public String name;
    Pair ( int rollNo., String name)
     {
      this. rollNo = rollNo;
      this. name = name;
     }
}
```

```
ArrayList <Pair> list = new ArrayList<> ();
  Pair one = new Pair ( "1", "ABC");
  Pair two = new Pair ( "2", "XYZ");
  Pair three = new Pair ( "3", "KBC");
  Pair four = new Pair ( "4", "CBC");

  list: add(one);
   list. add (two);
   list. add (three);
   list. add( four);
```

10000

Dart, kotlin

for ( Pair value : list ) — syntax

{

}


```
for ( int i=0; i< list.size(); i++)
  {
    Pair data = list.get(i);
      if (data.rollNo == 4 )
      {
        print ( data.name);
      Break;
  }
```

key value

Map <Integer, Pair> map= new HashMap<>();

```
Pair one = new Pair ( "1", "ABC");
map.put (one.rollNo, one);
map.put (two.rollNo, two);
```

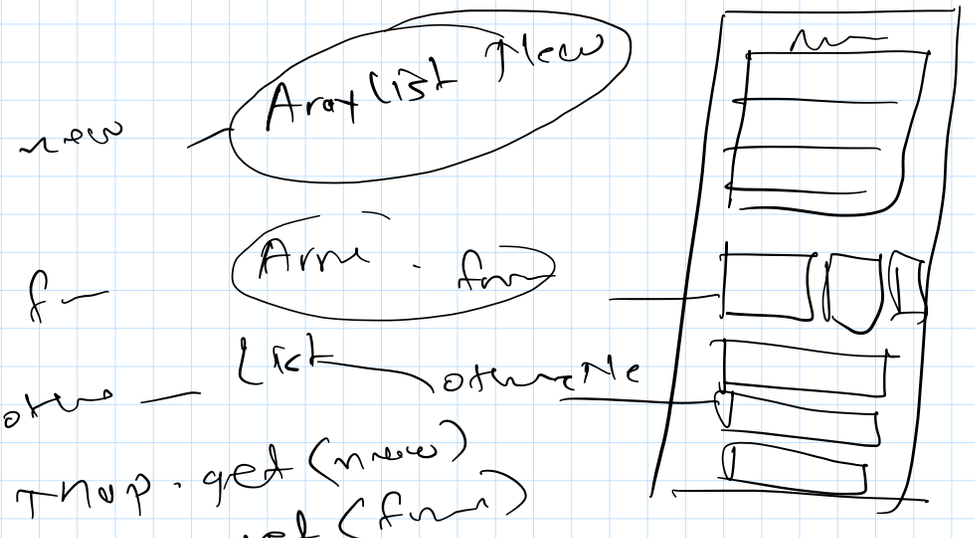
{ 1000 , 100 , 1 } — constant time operation

Pair pair = map.get(1000);

Pair pair = map.get(1);

String name = pair.name;

= abc — faster operation search




Tmap.get(new)
get(fun)

Tmap —

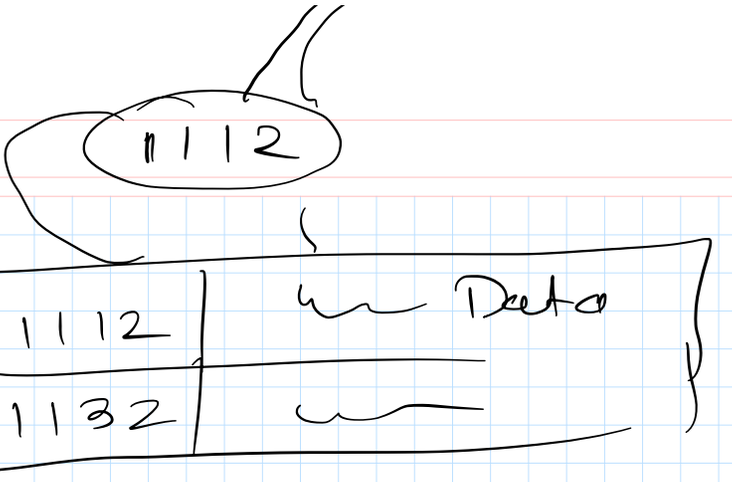
first
in — <key>, value)

100020 — Hashing mech.

Hashing (two, value)

1112

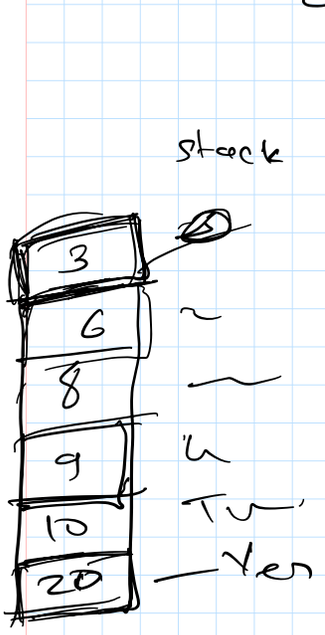
map.get(two)

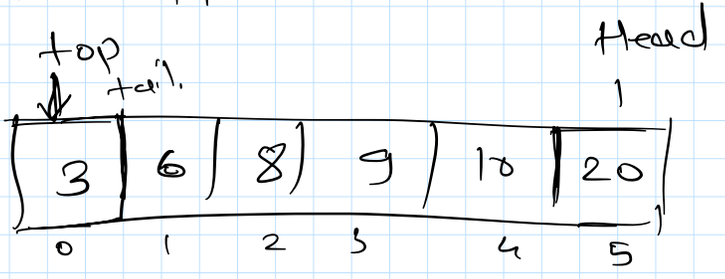


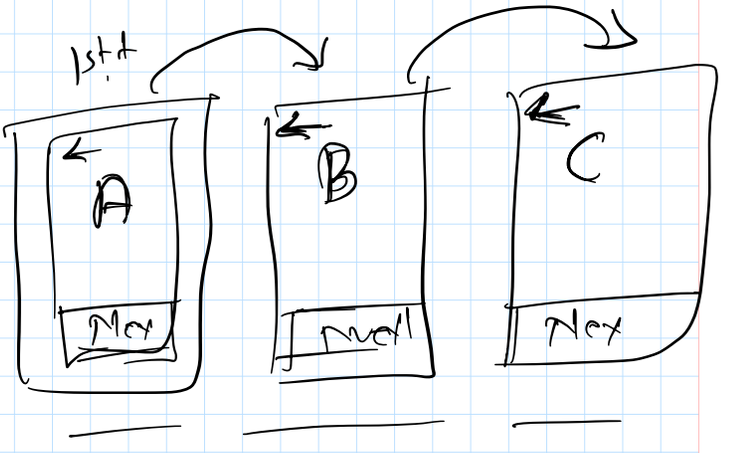

## Stack & Queue: D.S.

stack – LIFO

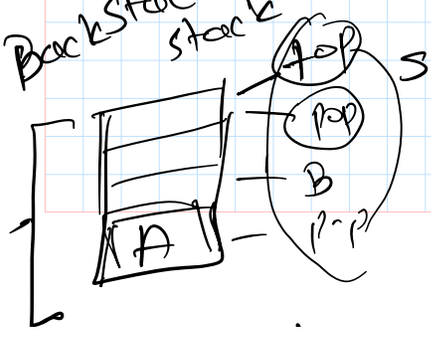
queue – FIFO




BackStack




stack < screens > screens.

screens.add(A)

scn.add(B)

scn.add(C)

Back
closing the app

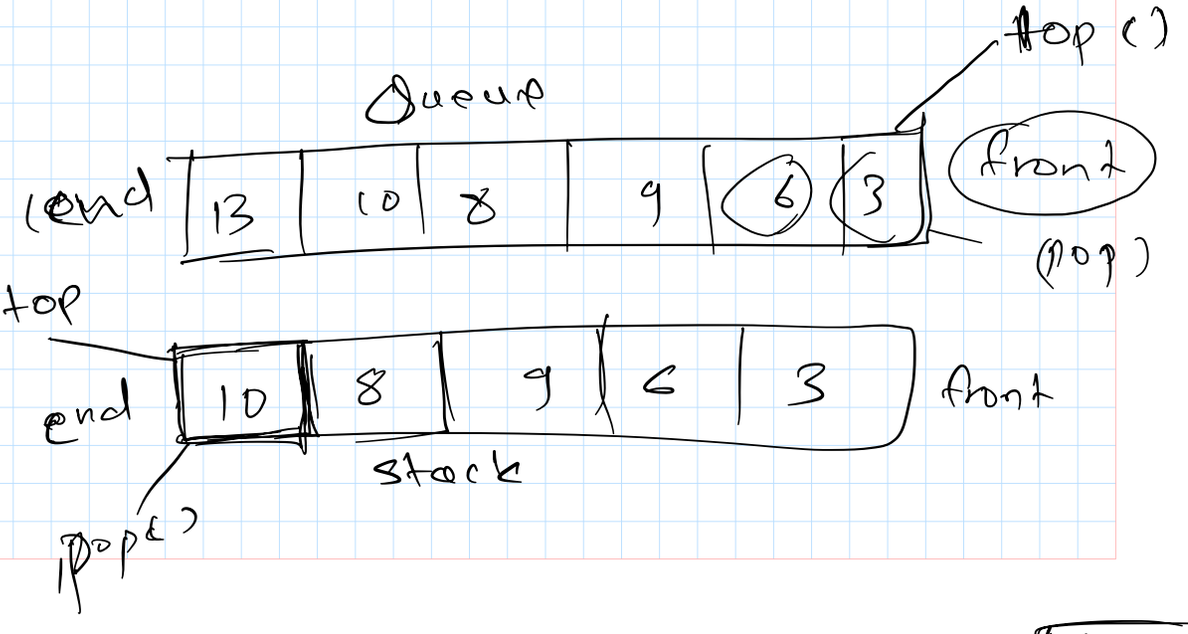
Navigation
Stack
(popBack())

.add


Applications Back
navigation
we extensively
use Stack
data structure.

Stack IPP

Queue




fixed

Queue & stack with either - Array()
+ LinkedList

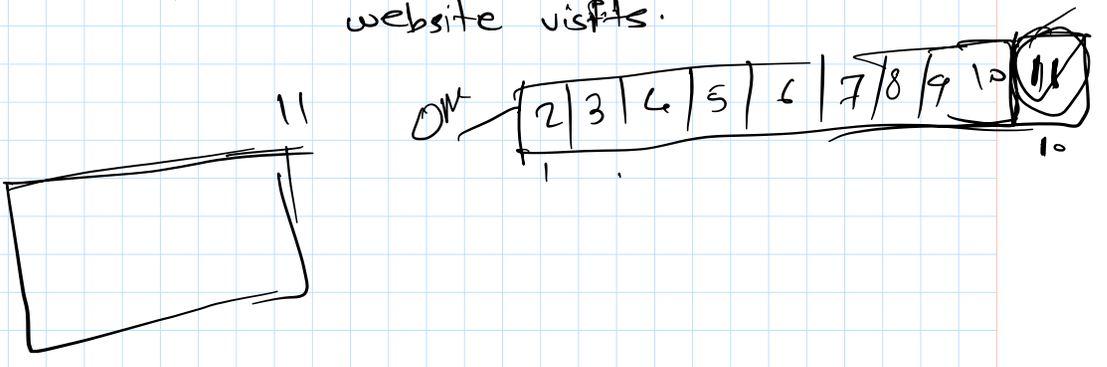
Dynamic

SDK

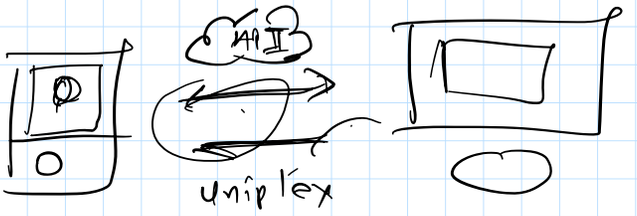
Dart SDK

## Application of Queue in Mobile.

website visits.




2) mobile




1) api call - fire2 Maps
2) - profile
3) - friends list


Queue API calls.

Queue of requests

API calls

10

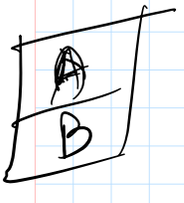
1) Queue of Network requests, API calls.

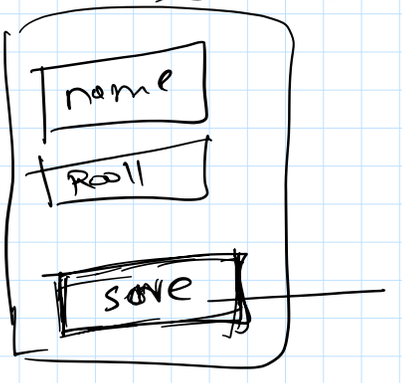
Linear DS.
1) Array
2) List - Linked, ArrayList
3) Map - key value, Hashing
4) Stack,
5) Queue
6) Set - unordered storing
doesn't allow duplicates.

Non-linear DS.
1) Tree
2) Graph

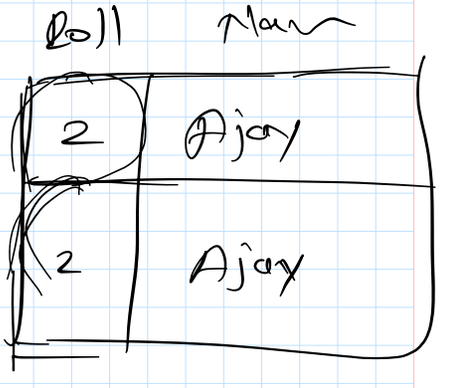
```
Set<String> set = new HashSet<>();
```

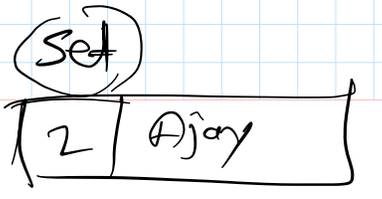
```
set.add("B");
set.add("A");
set.add("A");
set.add("C");
```

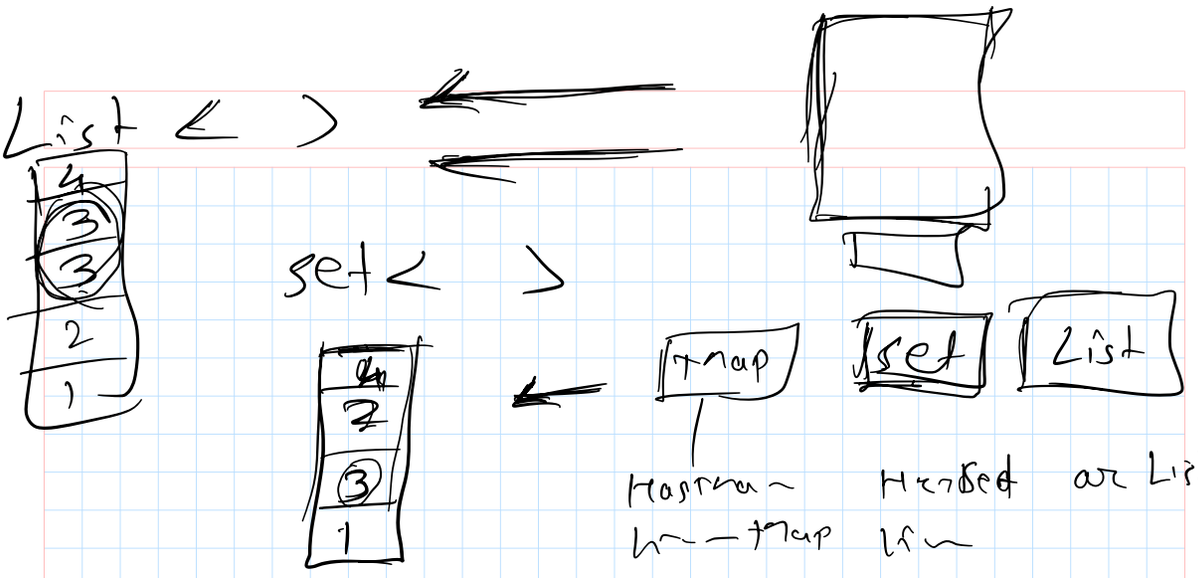



List - Duplicates






---

1) Array
2) Set
3) List, LinkedList
4) Map, HashMap
6) Stack, Queue.

Key, value

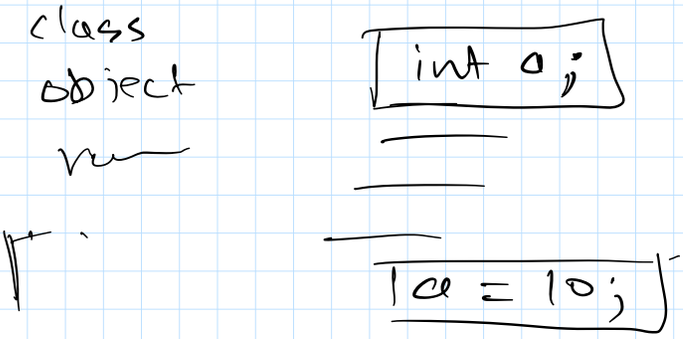
Java Collections

Iterables

---

class

object




Pair par.