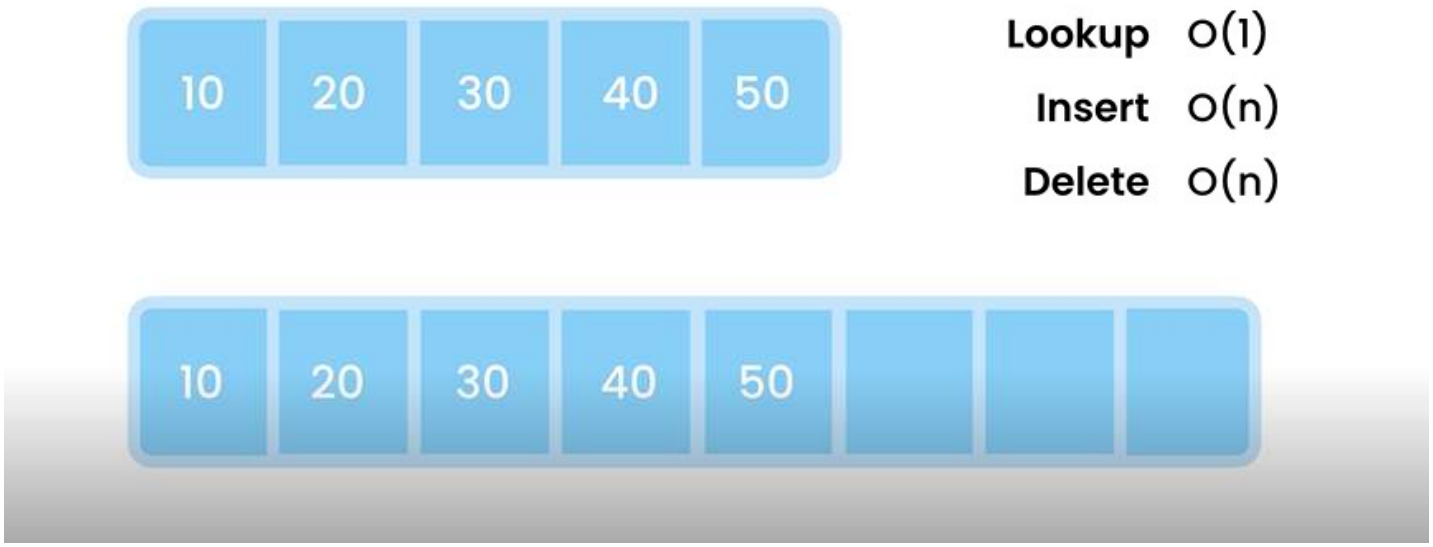
DataStructure

Array

- Нэг төрлийн өгөгдөл хадгалахад ашиглана. Тогтмол урттай. Ахин өргөтгөх боломжгүй. Ямар нэгэн элмент устгахад зөөх үйлдлийг хийнэ.

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
```

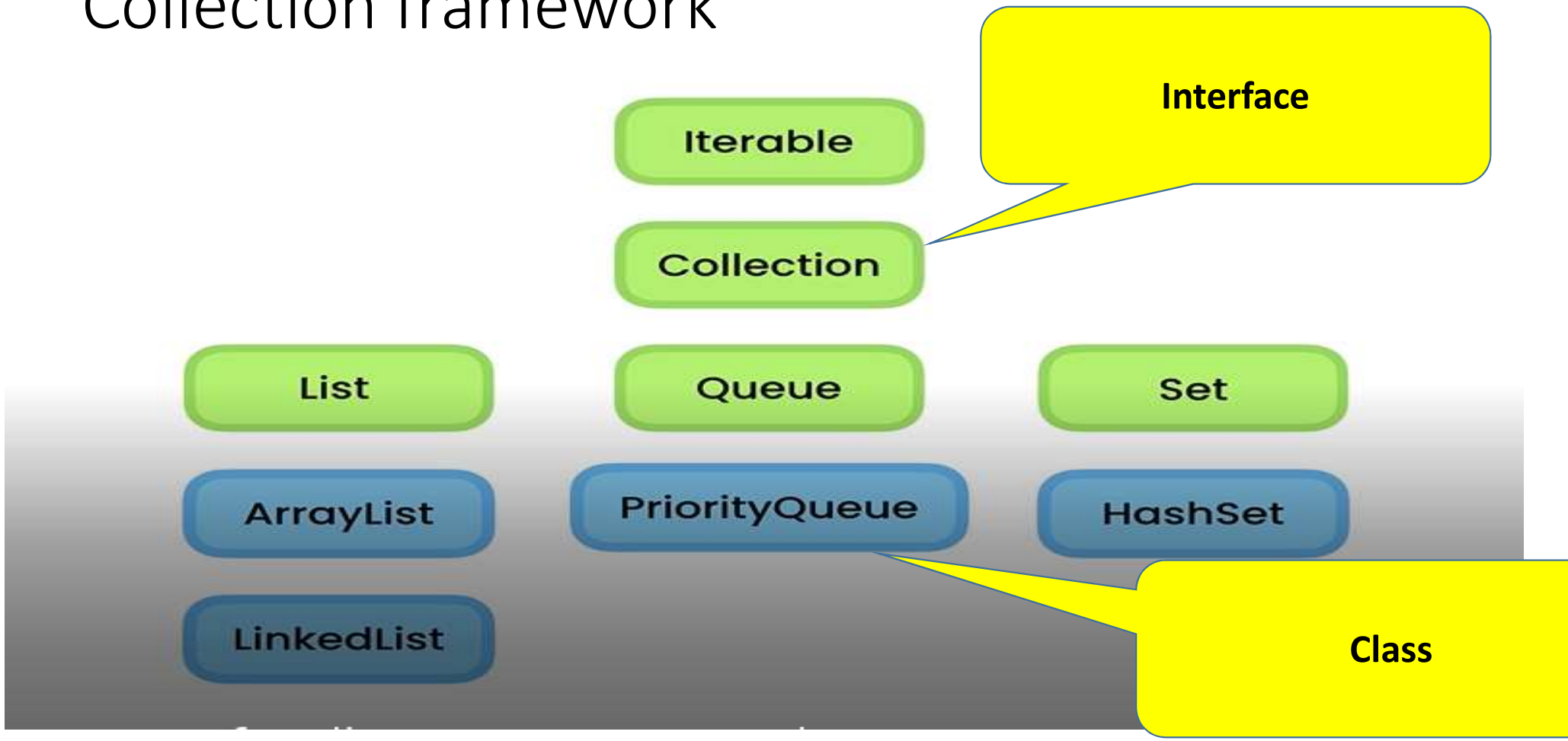


Lookup	O(1)
Insert	O(n)
Delete	O(n)

```
int[] aryNums;
aryNums = new int[6];
```

Collections

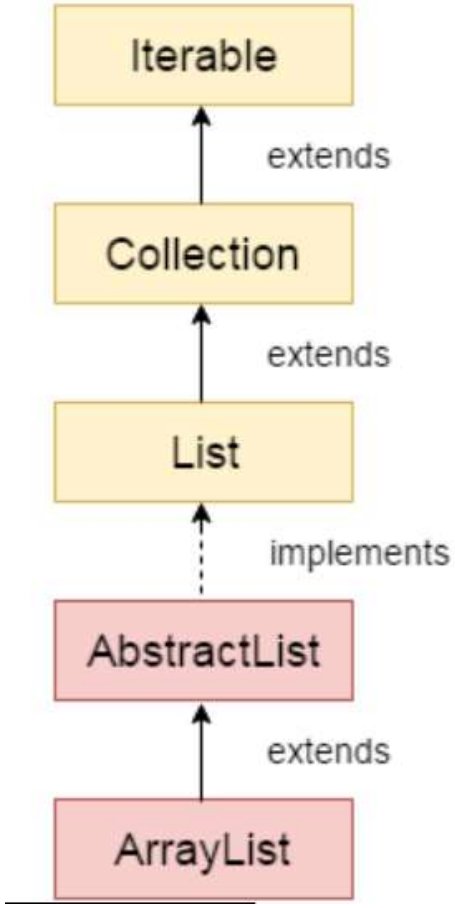
Collection framework



Java ArrayList

- Java **ArrayList** class uses a *dynamic array* for storing the elements. It is like an array, but there is *no size limit*. We can add or remove elements anytime. So, it is much more flexible than the traditional array. It is found in the *java.util* package. It is like the Vector in C++.
- 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.
- In ArrayList, manipulation is little bit slower than the LinkedList in Java because a lot of shifting needs to occur if any element is removed from the array list.
- **ArrayList<String> list=new ArrayList<String>();**

```
import java.util.*;
 public class ArrayListExample1{
 public static void main(String args[]){
  ArrayList<String> list=new ArrayList<String>();//Creating arraylist
      list.add("Mango");//Adding object in arraylist
      list.add("Apple");
      list.add("Banana");
      list.add("Grapes");
      //Printing the arraylist object
      System.out.println(list);
 }
}
```



`ArrayList` methods (10.1)

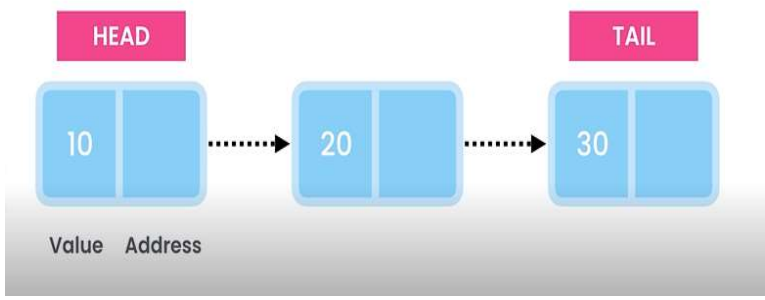
`add(`**value**`)`	appends value at end of list
`add(`**index**`, `**value**`)`	inserts given value just before the given index, shifting subsequent values to the right
`clear()`	removes all elements of the list
`indexOf(`**value**`)`	returns first index where given value is found in list (-1 if not found)
`get(`**index**`)`	returns the value at given index
`remove(`**index**`)`	removes/returns value at given index, shifting subsequent values to the left
`set(`**index**`, `**value**`)`	replaces value at given index with given value
`size()`	returns the number of elements in list
`toString()`	returns a string representation of the list such as `"[3, 42, -7, 15]"`

Use an `ArrayList` for storing and accessing data, and `LinkedList` to manipulate data.

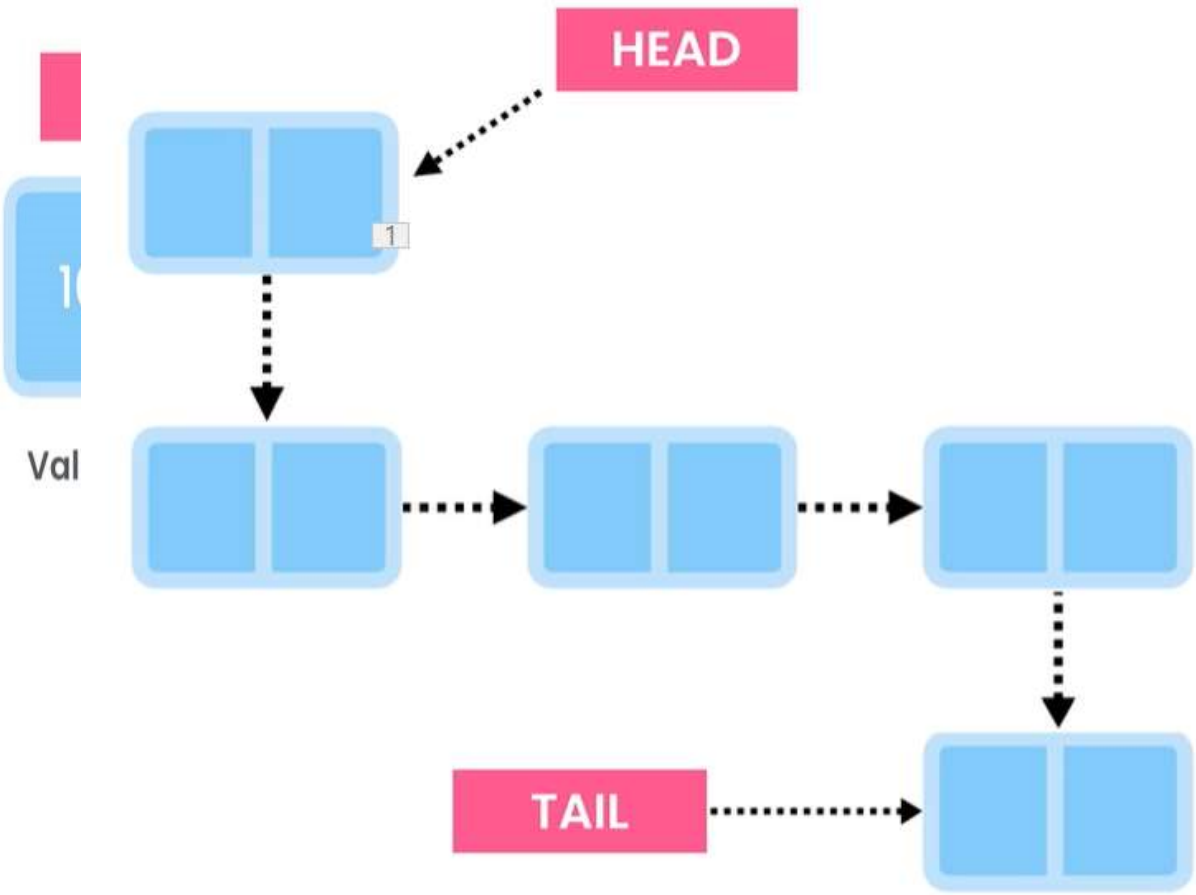
LinkedList

The `LinkedList` stores its items in "containers." The list has a link to the first container and each container has a link to the next container in the list. To add an element to the list, the element is placed into a new container and that container is linked to one of the other containers in the list.

Method	Description	Try it
addFirst()	Adds an item to the beginning of the list.	Try it »
addLast()	Add an item to the end of the list	Try it »
removeFirst()	Remove an item from the beginning of the list.	Try it »
removeLast()	Remove an item from the end of the list	Try it »
getFirst()	Get the item at the beginning of the list	Try it »
getLast()	Get the item at the end of the list	Try it »



LinkedList



LOOKUP

By Value	O(n)
By Index	

INSERT

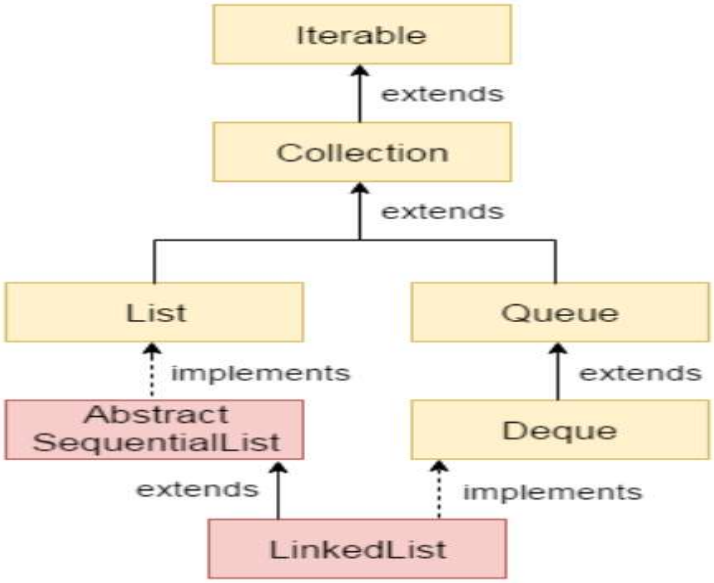
At the End	O(1)
At the Beginning	O(1)
In the Middle	O(n)

DELETE

From the Beginning	O(1)
From the End	O(n)
From the Middle	O(n)

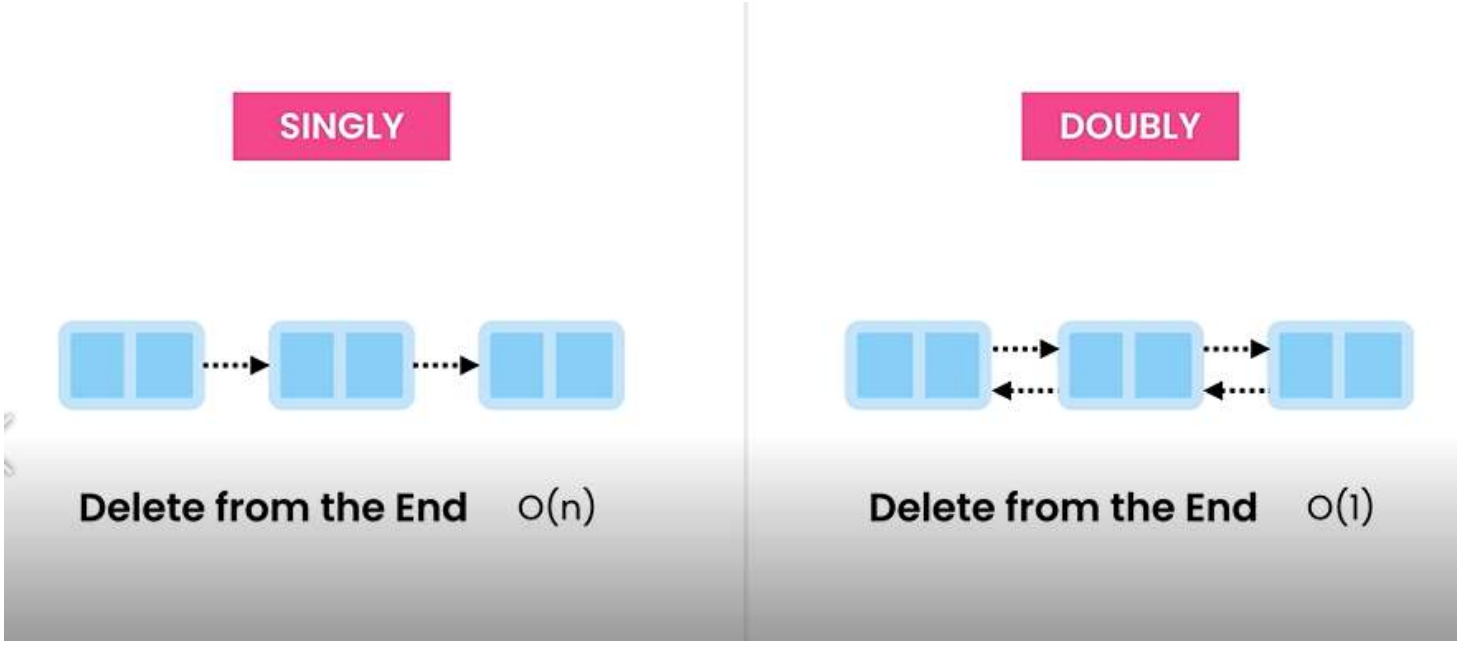
```java
import java.util.LinkedList;

public class Main {
    public static void main(String[] args) {
        LinkedList list = new LinkedList();
        list.addLast( e: 10);
        list.addLast( e: 20);
        list.addLast( e: 30);
    }
}
```

- Java LinkedList 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.

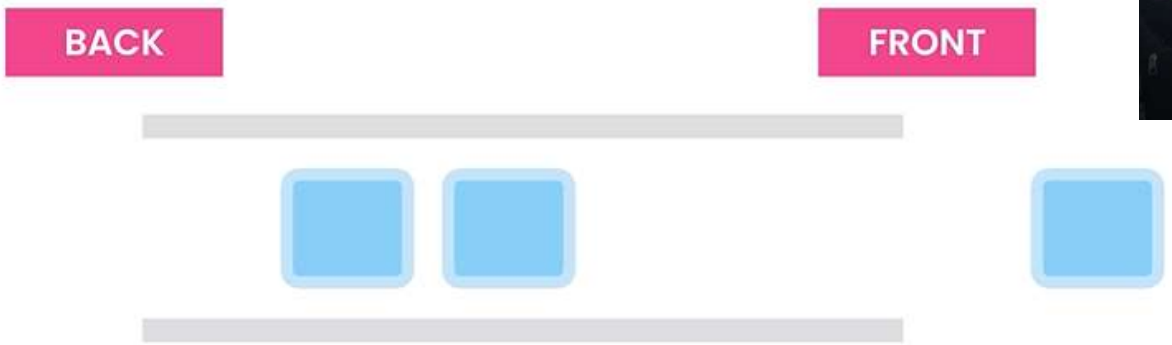


Use an `ArrayList` for storing and accessing data, and `LinkedList` to manipulate data

	ARRAYS	LINKED LISTS
Lookup		
By Index	O(1)	O(n)
By Value	O(n)	O(n)
Insert		
Beginning/End	O(n)	O(1)
Middle	O(n)	O(n)
Delete		
Beginning	O(n)	O(1)
Middle	O(n)	O(n)
End	O(n)	O(n)


SINGLY
Delete from the End O(n)
DOUBLY
Delete from the End O(1)

Java Queue-Interface



First-In First-Out (FIFO)

Last-In First-Out (LIFO)

https://docs.oracle.com/javase/8/docs/api/java/util/Queue.html

QUEUES

- Printers
- Operating systems
- Web servers
- Live support systems

OPERATIONS

enqueue	O(1)
dequeue	O(1)
peek	O(1)
isEmpty	O(1)
isFull	O(1)

All Known Implementing Classes:

AbstractQueue, ArrayBlockingQueue, ArrayDeque, ConcurrentLinkedDeque, ConcurrentLinkedQueue, De
LinkedBlockingQueue, LinkedList, LinkedTransferQueue, PriorityBlockingQueue, PriorityQueue, Syn

Queue-г хэрэгжүүлэх

```
import java.util.
import java.util.

public class Mair
    public static
        Queue<Int
        queue.add
        queue.add
        queue.add
        System.ou
    }
}
```

```
public class Main {
    public static void main(String[] args) {
        Queue<Integer> queue = new ArrayDeque<>();
        queue.add(10);
        queue.add(20);
        queue.add(30);
        var front = queue.remove();
```

Main › main()

un: Main

```
/Library/Java/JavaVirtualMachines/jdk-12.0.1.jdk/C
10
[20, 30]

Process finished with exit code 0
```

```
[10, 20, 30]
```

ArrayDeque-Stack-г хэрэгжүүлдэг.

- https://docs.oracle.com/javase/7/docs/api/java/util/Stack.html

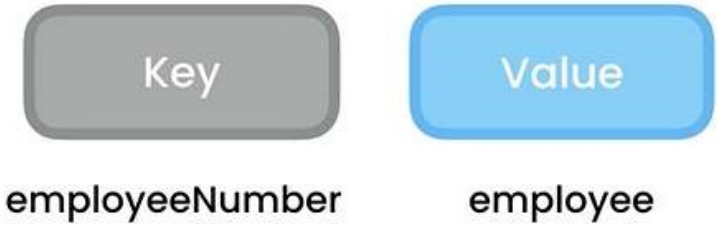
```java
import java.util.*;

class GFG {
    public static void main (String[] args) {
        Deque<Character> stack = new ArrayDeque<Character>();
        stack.push('A');
        stack.push('B');
        System.out.println(stack.peek());
        System.out.println(stack.pop());
    }
}
```

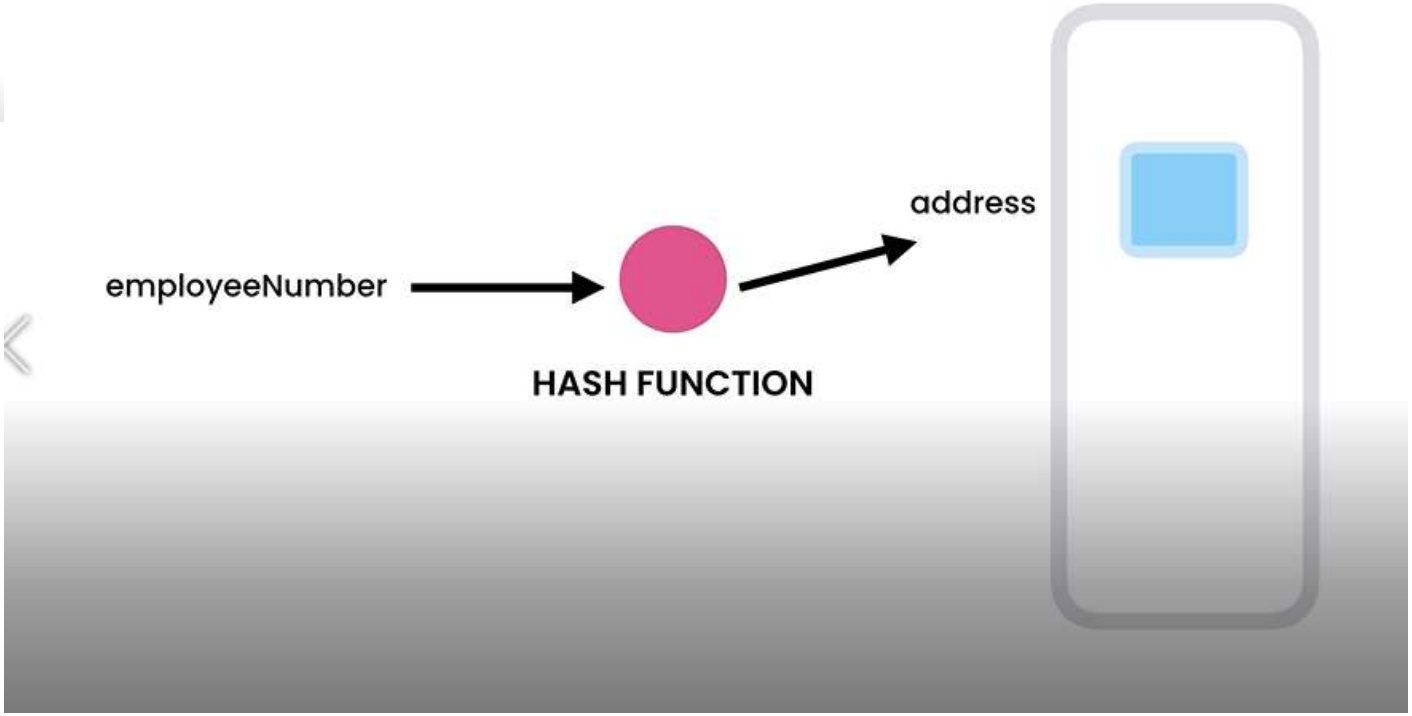
Java Hashmap-Хэрэглээ

- Өгөгдлийг түлхүүр, утга гэсэн 2 утгатайгаар хадгалдаг. Хайлт хийхэд маш хурдан.

IMPLEMENTATIONS

HashMap	Object	Dictionary	Dictionary
Java	JavaScript	Python	C#


Key
Value
employeeNumber
employee


address
employeeNumber
HASH FUNCTION

Interface Map<K,V> https://docs.oracle.com/javase/8/docs/api/java/util/Map.html

HASH TABLES

Insert	O(1)
Lookup	O(1)
Delete	O(1)

PREV CLASS NEXT CLASS FRAMES NO FRAMES ALL CLASSES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

compact1, compact2, compact3

java.util

Interface Map<K,V>

Type Parameters:

K - the type of keys maintained by this map

V - the type of mapped values

All Known Subinterfaces:

Bindings, ConcurrentMap<K,V>, ConcurrentNavigableMap<K,V>, LogicalMessageContext, Me
SortedMap<K,V>

All Known Implementing Classes:

AbstractMap, Attributes, AuthProvider, ConcurrentHashMap, ConcurrentSkipListMap, Enu
PrinterStateReasons, Properties, Provider, RenderingHints, SimpleBindings, TabularDa

---

```
public interface Map<K,V>
```

HashMap, Hashtable, ConcurrentHashMap

```
Map<Integer, String> map = new HashMap<>();
map.put(1, "Mosh");
map.put(2, "John");
map.put(3, "Mary");
map.put(3, "Marianne");
System.out.println(map);
```

```
{1=Mosh, 2=John, 3=Marianne}
```

Түлхүүр давтагдахгүй. Учраас сүүлийн утгыг авна.

```java
        Map<Integer, String> map = new HashMap<>();
        map.put(1, "Mosh");
        map.put(2, "John");
        map.put(3, "Mary");
        map.containsKey(3);  // O(1)
        map.containsValue("Mosh"); // O(n)
        System.out.println(map);
    }
}
```

```java
        // Value: Name (String)
        Map
        map
        map
        map
        Sys
        for (var item : map.entrySet())
            System.out.println(item);
    }
}
```

Main › main()

Main

```
{1=Mosh, 2=John, 3=Mary}
1=Mosh
2=John
3=Mary
```

Даалгавар-Counting Internet Addresses

- Internet TCP/IP addresses provide for two names for each computer
  - A host name, meaningful to humans
  - an IP address, meaningful to computers
- Problem:
  - network administrator needs to review file of IP addresses using a network gateway
- Solution:
  - read file of addresses
  - keep track of addresses and how many times each address shows up in the file

Class `AddressCounter`

- Note source code, Figure 12.1
- Attributes
  - maximum message length
  - address
  - count
- Methods
  - constructor
  - comparison method, `equals()`
  - count incrementer
  - accessors for address, count
  - to-string converter for output

Class `GatewayUsageCounter`

- Note source code
- Purpose
  - counts IP addresses using an array list
- Receives name of text file from `args[0]`
- Action:
  - reads IP address from file
  - prints listing of IP addresses and access count for each
- Note use of `ArrayList` class
  - can grow or shrink as needed

Даалгавар-2

- Stack-г хэрэгжүүл.
- Өгсөн тектэнд хамгийн их давтагдан орсон 2 үгийг гарга.

Анхаарал тавьсанд баярлалаа.