Table of Contents

ArrayList 2

 Dupliecate values 2

 Random access 2

LinkedList 3

 Duplicate values 3

 Sequential access 3

HashSet 5

 Unique values 5

 Random order 5

TreeSet 6

 Unique values 6

 Natrual order 6

LinkedHashSet 7

 Unique values 7

 Insertion order 7

ArrayDeque 8

 Insert and remove elements from both ends 8

 "Double Ended Queue" and is pronounced "deck" 8

HashMap 10

 Unique key/value pairs 10

 Random order 10

HashMap 12

 Frequency Example 12

TreeMap 13

 Unique key/value pairs 13

 Natrual order 13

LinkedHashMap 14

 Unique key/value paris 14

 Insertion order 14

# ArrayList

## Dupliecate values

## Random access

import java.util.\*;

public class ArrayListExample {

public static void main(String args[]) {

/\* Creation of ArrayList: I'm going to add String

\* elements so I made it of string type \*/

ArrayList<String> obj = new ArrayList<String>();

/\*This is how elements should be added to the array list\*/

obj.add("Ajeet");

obj.add("Harry");

obj.add("Chaitanya");

obj.add("Steve");

obj.add("Anuj");

/\* Displaying array list elements \*/

System.out.println("Currently the array list has following elements:"+obj);

/\*Add element at the given index\*/

obj.add(0, "Rahul");

obj.add(1, "Justin");

/\*Remove elements from array list like this\*/

obj.remove("Chaitanya");

obj.remove("Harry");

System.out.println("Current array list is:"+obj);

/\*Remove element from the given index\*/

obj.remove(1);

System.out.println("Current array list is:"+obj);

}

}

Output:

Currently the array list has following elements:[Ajeet, Harry, Chaitanya, Steve, Anuj]

Current array list is:[Rahul, Justin, Ajeet, Steve, Anuj]

Current array list is:[Rahul, Ajeet, Steve, Anuj]

# LinkedList

## Duplicate values

## Sequential access

import java.util.\*;

public class LinkedListExample {

public static void main(String args[]) {

/\* Linked List Declaration \*/

LinkedList<String> linkedlist = new LinkedList<String>();

/\*add(String Element) is used for adding

\* the elements to the linked list\*/

linkedlist.add("Item1");

linkedlist.add("Item5");

linkedlist.add("Item3");

linkedlist.add("Item6");

linkedlist.add("Item2");

/\*Display Linked List Content\*/

System.out.println("Linked List Content: " +linkedlist);

/\*Add First and Last Element\*/

linkedlist.addFirst("First Item");

linkedlist.addLast("Last Item");

System.out.println("LinkedList Content after addition: " +linkedlist);

/\*This is how to get and set Values\*/

Object firstvar = linkedlist.get(0);

System.out.println("First element: " +firstvar);

linkedlist.set(0, "Changed first item");

Object firstvar2 = linkedlist.get(0);

System.out.println("First element after update by set method: " +firstvar2);

/\*Remove first and last element\*/

linkedlist.removeFirst();

linkedlist.removeLast();

System.out.println("LinkedList after deletion of first and last element: " +linkedlist);

/\* Add to a Position and remove from a position\*/

linkedlist.add(0, "Newly added item");

linkedlist.remove(2);

System.out.println("Final Content: " +linkedlist);

}

}

Output:

Linked List Content: [Item1, Item5, Item3, Item6, Item2]

LinkedList Content after addition: [First Item, Item1, Item5, Item3, Item6, Item2, Last Item]

First element: First Item

First element after update by set method: Changed first item

LinkedList after deletion of first and last element: [Item1, Item5, Item3, Item6, Item2]

Final Content: [Newly added item, Item1, Item3, Item6, Item2]

# HashSet

## Unique values

## Random order

import java.util.HashSet;

public class HashSetExample {

public static void main(String args[]) {

// HashSet declaration

HashSet<String> hset = new HashSet<String>();

// Adding elements to the HashSet

hset.add("Apple");

hset.add("Mango");

hset.add("Grapes");

hset.add("Orange");

hset.add("Fig");

//Addition of duplicate elements

hset.add("Apple");

hset.add("Mango");

//Addition of null values

hset.add(null);

hset.add(null);

//Displaying HashSet elements

System.out.println(hset);

}

}

Output

[null, Mango, Grapes, Apple, Orange, Fig]

# TreeSet

## Unique values

## Natrual order

import java.util.TreeSet;

public class TreeSetExample {

public static void main(String args[]) {

// TreeSet of String Type

TreeSet<String> tset = new TreeSet<String>();

// Adding elements to TreeSet<String>

tset.add("ABC");

tset.add("String");

tset.add("Test");

tset.add("Pen");

tset.add("Ink");

tset.add("Jack");

//Displaying TreeSet

System.out.println(tset);

// TreeSet of Integer Type

TreeSet<Integer> tset2 = new TreeSet<Integer>();

// Adding elements to TreeSet<Integer>

tset2.add(88);

tset2.add(7);

tset2.add(101);

tset2.add(0);

tset2.add(3);

tset2.add(222);

System.out.println(tset2);

}

}

Output:

[ABC, Ink, Jack, Pen, String, Test]

[0, 3, 7, 88, 101, 222]

# LinkedHashSet

## Unique values

## Insertion order

import java.util.LinkedHashSet;

public class LinkedHashSetExample {

public static void main(String args[]) {

// LinkedHashSet of String Type

LinkedHashSet<String> lhset = new LinkedHashSet<String>();

// Adding elements to the LinkedHashSet

lhset.add("Z");

lhset.add("PQ");

lhset.add("N");

lhset.add("O");

lhset.add("KK");

lhset.add("FGH");

System.out.println(lhset);

// LinkedHashSet of Integer Type

LinkedHashSet<Integer> lhset2 = new LinkedHashSet<Integer>();

// Adding elements

lhset2.add(99);

lhset2.add(7);

lhset2.add(0);

lhset2.add(67);

lhset2.add(89);

lhset2.add(66);

System.out.println(lhset2);

}

}

Output:

[Z, PQ, N, O, KK, FGH]

[99, 7, 0, 67, 89, 66]

# ArrayDeque

## Insert and remove elements from both ends

## "Double Ended Queue" and is pronounced "deck"

import java.util.Deque;

import java.util.ArrayDeque;

public class ArrayDequeExample {

public static void main(String[] args) {

/\*

\* We cannot create instance of a Deque as it is an

\* interface, we can create instance of ArrayDeque or

\* LinkedList and assign it to Deque

\*/

Deque<String> dq = new ArrayDeque<String>();

/\*

\* Adding elements to the Deque.

\* addFirst() adds element at the beginning

\* and addLast() method adds at the end.

\*/

dq.add("Glenn");

dq.add("Negan");

dq.addLast("Maggie");

dq.addFirst("Rick");

dq.add("Daryl");

System.out.println("Elements in Deque:"+dq);

/\*

\* We can remove element from Deque using remove() method,

\* we can use normal remove() method which removes first

\* element or we can use removeFirst() and removeLast()

\* methods to remove first and last element respectively.

\*/

System.out.println("Removed element: "+dq.removeLast());

/\*

\* element() method - returns the head of the

\* Deque. Head is the first element of Deque

\*/

System.out.println("Head: "+dq.element());

/\*

\* pollLast() method - this method removes and returns the

\* tail of the Deque(last element). Returns null if the Deque is empty.

\* We can also use poll() or pollFirst() to remove the first element of

\* Deque.

\*/

System.out.println("poll(): "+dq.pollLast());

/\*

\* peek() method - it works same as element() method,

\* however it returns null if the Deque is empty. We can also use

\* peekFirst() and peekLast() to retrieve first and last element

\*/

System.out.println("peek(): "+dq.peek());

//Again printing the elements of Deque

System.out.println("Elements in Deque:"+dq);

}

}

Output:

Elements in Deque:[Rick, Glenn, Negan, Maggie, Daryl]

Removed element: Daryl

Head: Rick

poll(): Maggie

peek(): Rick

Elements in Deque:[Rick, Glenn, Negan]

# HashMap

## Unique key/value pairs

## Random order

import java.util.HashMap;

import java.util.Map;

import java.util.Iterator;

import java.util.Set;

public class Details {

public static void main(String args[]) {

/\* This is how to declare HashMap \*/

HashMap<Integer, String> hmap = new HashMap<Integer, String>();

/\*Adding elements to HashMap\*/

hmap.put(12, "Chaitanya");

hmap.put(2, "Rahul");

hmap.put(7, "Singh");

hmap.put(49, "Ajeet");

hmap.put(3, "Anuj");

/\* Display content using Iterator\*/

Set set = hmap.entrySet();

Iterator iterator = set.iterator();

while(iterator.hasNext()) {

Map.Entry mentry = (Map.Entry)iterator.next();

System.out.print("key is: "+ mentry.getKey() + " & Value is: ");

System.out.println(mentry.getValue());

}

/\* Get values based on key\*/

String var= hmap.get(2);

System.out.println("Value at index 2 is: "+var);

/\* Remove values based on key\*/

hmap.remove(3);

System.out.println("Map key and values after removal:");

Set set2 = hmap.entrySet();

Iterator iterator2 = set2.iterator();

while(iterator2.hasNext()) {

Map.Entry mentry2 = (Map.Entry)iterator2.next();

System.out.print("Key is: "+mentry2.getKey() + " & Value is: ");

System.out.println(mentry2.getValue());

}

}

}

Output:

key is: 49 & Value is: Ajeet

key is: 2 & Value is: Rahul

key is: 3 & Value is: Anuj

key is: 7 & Value is: Singh

key is: 12 & Value is: Chaitanya

Value at index 2 is: Rahul

Map key and values after removal:

Key is: 49 & Value is: Ajeet

Key is: 2 & Value is: Rahul

Key is: 7 & Value is: Singh

Key is: 12 & Value is: Chaitanya

# HashMap

## Frequency Example

static int sockMerchant(int arraySize, int[] array) {

HashMap <Integer, Integer> colors = new HashMap<Integer, Integer>();

for (int color : array) {

Integer frequency = colors.get(color);

if (frequency == null) {

colors.put(color, 1);

} else {

colors.put(color, frequency + 1);

}

}

int pairs = 0;

for (Integer frequency : colors.values()) {

pairs = pairs + (frequency / 2);

}

return pairs;

# TreeMap

## Unique key/value pairs

## Natrual order

import java.util.TreeMap;

import java.util.Set;

import java.util.Iterator;

import java.util.Map;

public class Details {

public static void main(String args[]) {

/\* This is how to declare TreeMap \*/

TreeMap<Integer, String> tmap =

new TreeMap<Integer, String>();

/\*Adding elements to TreeMap\*/

tmap.put(1, "Data1");

tmap.put(23, "Data2");

tmap.put(70, "Data3");

tmap.put(4, "Data4");

tmap.put(2, "Data5");

/\* Display content using Iterator\*/

Set set = tmap.entrySet();

Iterator iterator = set.iterator();

while(iterator.hasNext()) {

Map.Entry mentry = (Map.Entry)iterator.next();

System.out.print("key is: "+ mentry.getKey() + " & Value is: ");

System.out.println(mentry.getValue());

}

}

}

Output:

key is: 1 & Value is: Data1

key is: 2 & Value is: Data5

key is: 4 & Value is: Data4

key is: 23 & Value is: Data2

key is: 70 & Value is: Data3

# LinkedHashMap

## Unique key/value paris

## Insertion order

import java.util.LinkedHashMap;

import java.util.Set;

import java.util.Iterator;

import java.util.Map;

public class LinkedHashMapDemo {

public static void main(String args[]) {

// HashMap Declaration

LinkedHashMap<Integer, String> lhmap =

new LinkedHashMap<Integer, String>();

//Adding elements to LinkedHashMap

lhmap.put(22, "Abey");

lhmap.put(33, "Dawn");

lhmap.put(1, "Sherry");

lhmap.put(2, "Karon");

lhmap.put(100, "Jim");

// Generating a Set of entries

Set set = lhmap.entrySet();

// Displaying elements of LinkedHashMap

Iterator iterator = set.iterator();

while(iterator.hasNext()) {

Map.Entry me = (Map.Entry)iterator.next();

System.out.print("Key is: "+ me.getKey() +

"& Value is: "+me.getValue()+"\n");

}

}

}

Output:

Key is: 22& Value is: Abey

Key is: 33& Value is: Dawn

Key is: 1& Value is: Sherry

Key is: 2& Value is: Karon

Key is: 100& Value is: Jim