TECHPROED

PROFESSIONAL TECHNOLOGY EDUCATION

WELCOME TO TECHPROED JAVA TUTORIAL

Testi baslatmak icin asagidaki adımları takip ediniz

Go to www.socrative.com

Click on Login

Click on Student Login

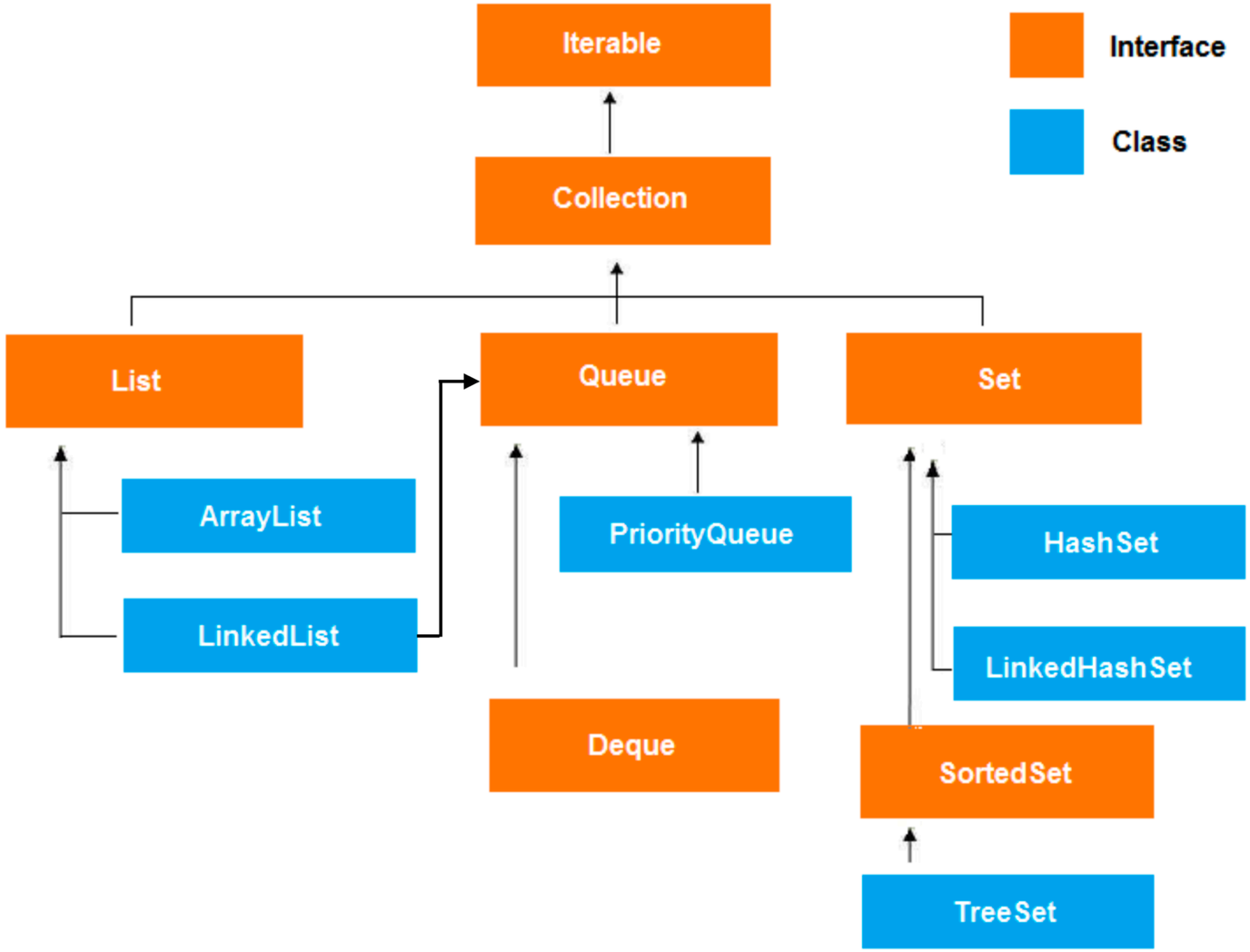
***Room Name*: A L P T E K I N 3 5 2 3**

Kayıtta kullandığınız ismi tam olarak yazınız

Time: 11 Minutes

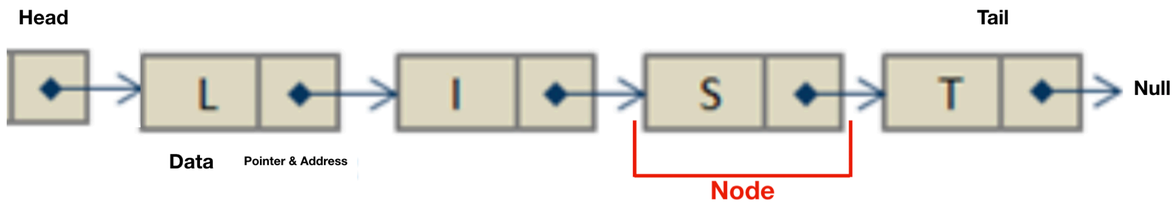
TECHPROED
PROFESSIONAL TECHNOLOGY EDUCATION

Collections



TECHPROED
PROFESSIONAL TECHNOLOGY EDUCATION

Linked List *(Class)*



1) Head'de sadece address vardır.

2) Son eleman Null'i point eder.

3) Her eleman icinde data ve address kismi olmak uzere iki kisim vardir.

4) Tum elemanlar pointerlar ve addressler kullanılarak birbirine baglanir.

5) Her eleman node olarak adlandirilir.

6) Array'lerden daha dynamic'dirler insert(ekleme) ve delete(silme) islemlerinde basarilidirlar.

7) Bir elemana ulaşmada yavaştırlar cunku index kullanmazlar

TECHPROED
PROFESSIONAL TECHNOLOGY EDUCATION

```java
import java.util.*;
public class JavaExample{
   public static void main(String args[]){

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

      //Adding elements to the Linked list
      list.add("Steve");
      list.add("Carl");
      list.add("Raj");
      list.add("Negan");
      list.add("Rick");

      //Removing First element
      //Same as list.remove(0);
      list.removeFirst();

      //Removing Last element
      list.removeLast();

      //Iterating LinkedList
      Iterator<String> iterator=list.iterator();
      while(iterator.hasNext()){
         System.out.print(iterator.next()+" ");
      }

      //removing 2nd element, index starts with 0
      list.remove(1);

      System.out.print("\nAfter removing second element: ");
      //Iterating LinkedList again
      Iterator<String> iterator2=list.iterator();
      while(iterator2.hasNext()){
         System.out.print(iterator2.next()+" ");
      }
   }
}
```

Delete an element

```java
import java.util.*;
public class JavaExample{
   public static void main(String args[]){

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

      //Adding elements to the Linked list
      list.add("Steve");
      list.add("Carl");
      list.add("Raj");

      //Adding an element to the first position
      list.addFirst("Negan");

      //Adding an element to the last position
      list.addLast("Rick");

      //Adding an element to the 3rd position
      list.add(2, "Glenn");

      //Iterating LinkedList
      Iterator<String> iterator=list.iterator();
      while(iterator.hasNext()){
        System.out.println(iterator.next());
      }
   }
}
```

Insert an element

TECHPROED
PROFESSIONAL TECHNOLOGY EDUCATION

Linked List Methodlari

```
LinkedList<String> llistobj  = new LinkedList<String>();
```

1) **boolean add(Object item)**: It adds the item at the end of the list.

```
llistobj.add("Hello");
```

It would add the string “Hello” at the end of the linked list.

2) **void add(int index, Object item)**: It adds an item at the given index of the the list.

```
llistobj.add(2, "bye");
```

This will add the string “bye” at the 3rd position(2 index is 3rd position as index starts with 0).

3) **boolean addAll(Collection c)**: It adds all the elements of the specified collection c to the list. It throws NullPointerException if the specified collection is null. Consider the below example –

```
LinkedList<String> llistobj = new LinkedList<String>();
ArrayList<String> arraylist= new ArrayList<String>();
arraylist.add("String1");
arraylist.add("String2");
llistobj.addAll(arraylist);
```

This piece of code would add all the elements of ArrayList to the LinkedList.

4) **boolean addAll(int index, Collection c)**: It adds all the elements of collection c to the list starting from a give index in the list. It throws NullPointerException if the collection c is null and IndexOutOfBoundsException when the specified index is out of the range.

```
llistobj.add(5, arraylist);
```

It would add all the elements of the ArrayList to the LinkedList starting from position 6 (index 5).

TECHPROED
PROFESSIONAL TECHNOLOGY EDUCATION

5) **void addFirst(Object item)**: It adds the item (or element) at the first position in the list.

```
llistobj.addFirst("text");
```

It would add the string "text" at the beginning of the list.

6) **void addLast(Object item)**: It inserts the specified item at the end of the list.

```
llistobj.addLast("Chaitanya");
```

This statement will add a string "Chaitanya" at the end position of the linked list.

7) **void clear()**: It removes all the elements of a list.

```
llistobj.clear();
```

8) **Object clone()**: It returns the copy of the list.

For e.g. My linkedList has four items: text1, text2, text3 and text4.

```
Object str= llistobj.clone();
 System.out.println(str);
```

Output: The output of above code would be:

[text1, text2, text3, text4]

9) **boolean contains(Object item)**: It checks whether the given item is present in the list or not. If the item is present then it returns true else false.

```
boolean var = llistobj.contains("TestString");
```

It will check whether the string "TestString" exist in the list or not.

TECHPROED
PROFESSIONAL TECHNOLOGY EDUCATION

10) **Object get(int index)**: It returns the item of the specified index from the list.

```
Object var = llistobj.get(2);
```

It will fetch the 3rd item from the list.

11) **Object getFirst()**: It fetches the first item from the list.

```
Object var = llistobj.getFirst();
```

12) **Object getLast()**: It fetches the last item from the list.

```
Object var= llistobj.getLast();
```

13) **int indexOf(Object item)**: It returns the index of the specified item.

```
llistobj.indexOf("bye");
```

14) **Object remove()**: It removes the first element of the list.

```
llistobj.remove();
```

15) **Object remove(int index)**: It removes the item from the list which is present at the specified index.

```
llistobj.remove(4);
```

It will remove the 5th element from the list.

16) **Object remove(Object obj)**: It removes the specified object from the list.

```
llistobj.remove("Test Item");
```

17) **Object removeFirst()**: It removes the first item from the list.

```
llistobj.removeFirst();
```

18) **Object removeLast()**: It removes the last item of the list.

```
llistobj.removeLast();
```

TECHPROED
PROFESSIONAL TECHNOLOGY EDUCATION

19) **Object set(int index, Object item)**: It updates the item of specified index with the give value.

```
llistobj.set(2, "Test");
```

It will update the 3rd element with the string “Test”.

20) **int size()**: It returns the number of elements of the list.

```
llistobj.size();
```

21) **Collections.sort(list)** : It sorts the list elements in natural order

TECHPROED
PROFESSIONAL TECHNOLOGY EDUCATION

Set *(Interface)*

Set bir Collection'dir ve tekrarlı elementleri kabul etmez.

Note: 3 farklı Set vardır:

1) **HashSet**
2) **TreeSet**
3) **LinkedHashSet.**

1) HashSet :

HashSet elemanları hash table'da depolar.

Index	
0	
1	
-	
-	
-	
11	defabc
12	
13	
14	cdefab
-	
-	
-	
-	
23	bcdefa
-	
-	
-	
38	abcdef
-	
-	

Note: Hashing tekniği kullanılarak herbir elemana unique kodlar oluşturulur.

HashTable : Bir hash table key/value yapisini kullanarak dataları depolar.

TECHPROED
PROFESSIONAL TECHNOLOGY EDUCATION

Note: HashSet elemanları rastgele sıralar.

Note: HashSet duplicate'e izin vermez. Duplicate elemanlar eklemek isterseniz üstüne yazar.

Note: HashSet null value kullanmaya izin verir, tekrar null eklemek isterseniz üstüne yazar.

```java
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);
    }
}
```

TECHPROED
PROFESSIONAL TECHNOLOGY EDUCATION

```java
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);
    }
 }
```

2) TreeSet :

TreeSet HashSet'e benzer ama elemanları natural order gore dizer.

Note: HashSet, TreeSet'den add(), remove(), contains(), size() gibi method'larda daha hizlidir.

Note: TreeSet duplication'a izin vermez.

Note: Eger sorted Set lazimsa; once HashSet Kullanmak sonra onu TreeSet'e çevirmek direkt TreeSet kullanmaktan daha hizlidir.

TECHPROED
PROFESSIONAL TECHNOLOGY EDUCATION

HashSet'i TreeSet'e çevirmek

```
import java.util.HashSet;
import java.util.TreeSet;
import java.util.Set;
class ConvertHashSettoTreeSet{
  public static void main(String[] args) {
     // Create a HashSet
     HashSet<String> hset = new HashSet<String>();

     //add elements to HashSet
     hset.add("Element1");
     hset.add("Element2");
     hset.add("Element3");
     hset.add("Element4");

     // Displaying HashSet elements
     System.out.println("HashSet contains: "+ hset);

     // Creating a TreeSet of HashSet elements
     Set<String> tset = new TreeSet<String>(hset);

     // Displaying TreeSet elements
     System.out.println("TreeSet contains: ");
     for(String temp : tset){
        System.out.println(temp);
     }
  }
}
```

TECHPROED
PROFESSIONAL TECHNOLOGY EDUCATION

3) LinkedHashSet :

LinkedHashSet elemanlari insertion order'a gore dizer

```java
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]
```

Output

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

TECHPROED
PROFESSIONAL TECHNOLOGY EDUCATION

Queue *(Interface)*

```java
import java.util.*;
public class QueueExample1 {

   public static void main(String[] args) {

      /*
       * We cannot create instance of a Queue as it is an
       * interface, we can create instance of LinkedList or
       * PriorityQueue and assign it to Queue
       */
      Queue<String> q = new LinkedList<String>();

      //Adding elements to the Queue
      q.add("Rick");
      q.add("Maggie");
      q.add("Glenn");
      q.add("Negan");
      q.add("Daryl");

      System.out.println("Elements in Queue:"+q);

      /*
       * We can remove element from Queue using remove() method,
       * this would remove the first element from the Queue
       */
      System.out.println("Removed element: "+q.remove());

      /*
       * element() method - this returns the head of the
       * Queue. Head is the first element of Queue
       */
      System.out.println("Head: "+q.element());

      /*
       * poll() method - this removes and returns the
       * head of the Queue. Returns null if the Queue is empty
       */
      System.out.println("poll(): "+q.poll());

      /*
       * peek() method - it works same as element() method,
       * however it returns null if the Queue is empty
       */
      System.out.println("peek(): "+q.peek());

      //Again displaying the elements of Queue
      System.out.println("Elements in Queue:"+q);
   }
}
```

Queue'da elemanlar sona eklenir, bastan silinir.
First In First Out
(FIFO)

```
Elements in Queue:[Rick, Maggie, Glenn, Negan, Daryl]
Removed element: Rick
Head: Maggie
poll(): Maggie
peek(): Glenn
Elements in Queue:[Glenn, Negan, Daryl]
```

TECHPROED
PROFESSIONAL TECHNOLOGY EDUCATION

Difference Between Queue created by using LinkedList Class and Queue created by using PriorityQueue Class

```
public class QueueInJava {

public static void main(String[] args) {
    Queue<String> queue = new LinkedList<String>();
    queue.add("Ishfaq");
    queue.add("Ramzan");
    queue.add("Nagoo");
    queue.add("Bangalore");

    System.out.println("Linked List Queue is:"+ queue);
    System.out.println("Linked List Queue Peek is :"+queue.peek());

    queue.poll();
    System.out.println("Linked List Queue after remove is:"+ queue);


    Queue<Integer> queuenew = new PriorityQueue<Integer>();


    queuenew.add(2);
    queuenew.add(3);
    queuenew.add(1);
    queuenew.add(0);
    queuenew.add(4);

    System.out.println("Priority Queue is:"+ queuenew);
    System.out.println("Priority Queue Peek is :"+queuenew.peek());

    int ieleFirst=queuenew.remove();
    System.out.println("Priority Queue Element Removed is:"+ ieleFirst);
    int ieleSecond=queuenew.remove();
    System.out.println("Priority Queue Element Removed is:"+ ieleSecond);
    System.out.println("Priority Queue after remove is:"+ queuenew);
  }

}
```

Eger Queue by LinkedList Class, elements get sorted in the same sequence in which they have been added to the Set.

If you create Queue by PriorityQueue Class, it orders the element in a rule which is declared by Java.

```
Linked List Queue is:[Ishfaq, Ramzan, Nagoo, Bangalore]
Linked List Queue Peek is :Ishfaq
Linked List Queue after remove is:[Ramzan, Nagoo, Bangalore]
Priority Queue is:[0, 1, 2, 3, 4]
Priority Queue Peek is :0
Priority Queue Element Removed is:0
Priority Queue Element Removed is:1
Priority Queue after remove is:[2, 3, 4]
```

TECHPROED
PROFESSIONAL TECHNOLOGY EDUCATION