Collection

Collection(I)

1. List(I)
2. set(I)
3. queue(I)

1. List(I)

1. duplicate are allowed in list
2. allows any no of null values
3. Storage Type: index
4. order of insertion-maintain

1. Arraylist(IC)
2. vector(IC)---->legacy
3. LinkedList(IC)

2. Set(I)

1. doesn't allow duplicate values
2. Allow only 1 null value(except TreeSet)
3. Storage Type: Hashtable
4. diff Order of insertion-Random insertion/Maintained/Ascending

1. Hashset(IC)
2. LinkedHashset(IC)
3. TreeSet(IC)

Cursor

1. Iterator-- all the collection object --universal curser
2. listIterator -- only for list interface type impl classes --not universal curser
3. enumeration -- legecy --not universal curser

1. Arraylist(IC)

1. Default/initial capacity for arraylist is 10
2. Incremental capacity= (current capacity*3/2)+1 =
3. data structure: Resizable
4. best choice: retrieval operation (RandomAccess interface is implemented in Arraylist & Vector)
5. worst choice: manipulation operation i.e. insertion in between arraylist or delete .

2. Vector(IC)

1. Default capacity for Vector is 10
*2. Incremental capacity = current capacity*2 =10*2=20
*3. data structure: doubly
4. best choice: retrieval operation (random access interface is implemented in arraylist & vector)
5. worst choice: manipulation operation i.e. insertion in between Vector or delete ()
*6. Vector is legacy class.

3. LinkedList(IC)

*1. No Default capacity

*2. data structure: linear
*3. best choice: manipulation operation i.e. insertion in between linkedlist or delete()
*4. worst choice: retrieval operation (random access interface is not implemented)

2. Set(I)

1. Doesn't allow duplicate
2. Allow only 1 null value(except treeset)
3. Storage type- HashTable
4. Order of insertion-->different order of insertion (random/asscending/maintain insertion)

1. Hashset(IC)
2. LinkedHashset(IC)
3. TreeSet(IC)

1.Hashset:

---

1. Order of insertion--> random insertion(ascending order or asci value)
2. DS: Hashtable

best choice: To remove duplicate elements when order of insertion is not mandatory.

2. LinkedHashset: (Linkedlist + hashset)

---

*1. Order of insertion->maintained
*2. DS: Hybrid (linear+ hashtable)

*best choice: To remove duplicate elements when order of insertion is mandatory.

3. TreeSet:

Note: we can store only homogeneous data

*1. Order of insertion--> Ascending order.
*2. DS: Balanced tree

best choice: To remove duplicate elements when order of insertion is Ascending order.

Arraylist Vector

1. not legacy class	legacy class
2. DS: Resizable	DS: Doubly
3. I.C=(C.C.*3/2)+1	IC=CC*2
4. not-synchronized not-thread safe	synchronized thread-safe
5. performance: high	performance: low

Arraylist	linkedList
1. deafult capacity:10	1. no deafult capacity
2. DS: resizable	2. DS: linear
3. retrival: best choice	3. retrival: worst choice
4. manipulation: worst choice	4. manipulation: best choice

List	set
1. duplicate: allowded	duplicate: not allowded
2. Any no of null values: allowded	Any no of null values: only one null value-except treeset
3. order of insertion-maintain	diff insertion --> depends on asci/maintain/asscending
4. storage type- index	storage type- hashtable

Cursors in Collections

1. Iterator
    1. All the collection object (7) --> Universal curser.
    2. Using iterater and Enumeration we can traverse collection object only in forword direction not in backword --> Single directional cursor
    3. By using iterater we can perform only read and remove operation we can not perform replace and addition of new object.

2. listIterator
    1. Only applicable for list interface type implementation classes (3) --> not universal curser.
    2. Using list iterator we can traverse a List in forward direction and backword direction--> bidirectional cursor
    3. By using listIterator we can perform read, remove, replace and addition of new object operations.

3. Enumeration
    1. Only applicable for legacy classes (1) ----> not universal curser.
    2. Using Enumeration and iterater we can traverse collection object only in forword direction not a backword --> Single directional cursor
    3. By using enumeration we can get only read access.

```
package Collection;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.ListIterator;
public class ex1_ArrayList
{
	public static void main(String[] args)
	{
		//ArrayList al=new ArrayList();	//initial/by default capacity=10
		ArrayList al=new ArrayList(50);	//initial capacity=50
		al.add("rahul");
		al.add(101);
		al.add(65.5f);
		al.add('A');
		al.add(null);
		al.add(101);
		al.add(null);

		System.out.println(al);
		System.out.println(al.size());		//7
		System.out.println(al.get(0));		//rahul
		System.out.println(al.contains(101));  //true
		System.out.println(al.isEmpty());	//false

		//update data
		al.set(0, "RAHUL");
		System.out.println(al);

		//add data in between Arraylist  -> right shift operation
		al.add(4, 500);
		System.out.println(al);

		//remove data in between Arraylist  -> left shift operation
		al.remove(4);
		System.out.println(al);


		System.out.println("------Print data using : for loop-------");
		for(int i=0; i<=al.size()-1; i++)
		{
			System.out.println(al.get(i));
		}

		System.out.println("------Print data using : for each loop-------");
		for(Object s1:al)
		{
			System.out.println(s1);
		}

		System.out.println("------Print data using : Iterator cursor-------");
		Iterator itr = al.iterator();	//copy all data from Arraylist to Iterator Object
										//itr= [rahul, 101, 65.5, A, null, 101, null ]

		while(itr.hasNext())  //false
		{
			System.out.println(itr.next());
		}


		System.out.println("------Print data using : ListIterator cursor-------");
		ListIterator litr = al.listIterator();
		while(litr.hasNext())
		{
			System.out.println(litr.next());
		}

		System.out.println("--------------");
		al.clear();
		System.out.println(al.size());
	}
}
```

```
package Collection;
import java.util.Enumeration;
import java.util.Iterator;
import java.util.ListIterator;
import java.util.Vector;
public class ex2_Vector
{
	public static void main(String[] args)
	{
		Vector v=new Vector();	//initial capacity=10
		v.add("rahul");
		v.add(101);
		v.add(65.5f);
		v.add('A');
		v.add(null);
		v.add(101);
		v.add(null);


		System.out.println(v.capacity());
		System.out.println(v);
		System.out.println(v.size());	//7
		System.out.println(v.get(0));	//rahul
		System.out.println(v.contains(101));  //true
		System.out.println(v.isEmpty());	//fvse

		//update data
		v.set(0, "RAHUL");
		System.out.println(v);

		//add data in between vector  -> right shift operation
		v.add(4, 500);
		System.out.println(v);

		//remove data in between vector  -> left shift operation
		v.remove(4);
		System.out.println(v);


		System.out.println("------Print data using : for loop-------");
		for(int i=0; i<=v.size()-1; i++)
		{
			System.out.println(v.get(i));
		}

		System.out.println("------Print data using : for each loop-------");
		for(Object s1:v)
		{
			System.out.println(s1);
		}

		System.out.println("------Print data using : Iterator cursor-------");
		Iterator itr = v.iterator();	//copy all data from vector to Iterator Object
                                                                                //itr= [rahul, 101, 65.5, A,
null, 101, null ]

		while(itr.hasNext())  //
		{
			System.out.println(itr.next());
		}

		System.out.println("------Print data using : ListIterator cursor-------");
		ListIterator litr = v.listIterator();
		while(litr.hasNext())
		{
			System.out.println(litr.next());
		}

		System.out.println("------Print data using : Enumeration cursor-------");
```

```
		Enumeration enu = v.elements();
		while(enu.hasMoreElements())
		{
			System.out.println(enu.nextElement());
		}

		System.out.println("--------------");
		v.clear();
		System.out.println(v.size());
	}
}


package Collection;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.ListIterator;

public class ex3_LinkedList
{
	public static void main(String[] args)
	{
		LinkedList ll=new LinkedList();
		ll.add("amol");
		ll.add(102);
		ll.add('A');
		ll.add(77.5f);
		ll.add(null);
		ll.add(102);
		ll.add(null);

		System.out.println(ll);
		System.out.println(ll.size());
		System.out.println(ll.get(0));
		System.out.println(ll.isEmpty());
		System.out.println(ll.contains(102));

		//update data
		ll.set(0, "AMOL");
		System.out.println(ll);

		//add data in between linkedList
		ll.add(4, 600);
		System.out.println(ll);

		//remove data in between linkedList
		ll.remove(4);
		System.out.println(ll);

		System.out.println("----print data using : for loop---------");
		for(int i=0; i<=ll.size()-1; i++)
		{
			System.out.println(ll.get(i));
		}

		System.out.println("----print data using : for each loop---------");
		for(Object s1:ll)
		{
			System.out.println(s1);
		}

		System.out.println("----print data using : Iterator cursor---------");
		Iterator itr = ll.iterator();
		while(itr.hasNext())
		{
			System.out.println(itr.next());
		}

		System.out.println("----print data using : ListIterator cursor---------");
		ListIterator litr = ll.listIterator();
		while(litr.hasNext())
```

```
		{
			System.out.println(litr.next());
		}

		ll.clear();
		System.out.println(ll.size());
	}
}
```

```
package Collection;
import java.util.HashSet;
import java.util.Iterator;
public class ex4_Hashset1
{
	public static void main(String[] args)
	{
		HashSet hs=new HashSet();
		hs.add("amol");
		hs.add(101);
		hs.add('A');
		hs.add(75.5f);
		hs.add(null);
		hs.add(101);
		hs.add(null);

		System.out.println(hs);
		System.out.println(hs.size());
		System.out.println(hs.isEmpty());
		System.out.println(hs.contains("amol"));

		hs.remove('A');
		System.out.println(hs);

		System.out.println("----print data using : For each loop--------");
		for(Object s1:hs)
		{
			System.out.println(s1);
		}

		System.out.println("----print data using : Iterator--------");
		Iterator itr = hs.iterator();
		while(itr.hasNext())
		{
			System.out.println(itr.next());
		}

		System.out.println("------");
		hs.clear();
		System.out.println(hs.size());
	}
}
```

```
package Collection;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.Iterator;
public class ex4_Hashset2
{
	public static void main(String[] args)
	{
		ArrayList al=new ArrayList();
		al.add("amol");
		al.add(101);
		al.add('A');
		al.add(75.5f);
		al.add(null);
		al.add(101);
```

```
                al.add(null);
                al.add('A');

                System.out.println(al);   //[amol, 101, A, 75.5, null, 101, null, A]


                HashSet hs=new HashSet(al);    //[amol, 101, A, 75.5, null]
                System.out.println(hs);
        }
}

package Collection;
import java.util.Iterator;
import java.util.LinkedHashSet;
public class ex5_LinkedHashset1
{
        public static void main(String[] args)
        {
                LinkedHashSet lhs=new LinkedHashSet();
                lhs.add("amol");
                lhs.add(101);
                lhs.add('A');
                lhs.add(75.5f);
                lhs.add(null);
                lhs.add(101);
                lhs.add(null);
                lhs.add("amol");

                System.out.println(lhs);
                System.out.println(lhs.size());
                System.out.println(lhs.isEmpty());
                System.out.println(lhs.contains("amol"));

                lhs.remove('A');
                System.out.println(lhs);

                System.out.println("----print data using : For each loop--------");
                for(Object s1:lhs)
                {
                        System.out.println(s1);
                }

                System.out.println("----print data using : Iterator--------");
                Iterator itr = lhs.iterator();
                while(itr.hasNext())
                {
                        System.out.println(itr.next());
                }

                System.out.println("------");
                lhs.clear();
                System.out.println(lhs.size());
        }
}
```

```
package Collection;
```

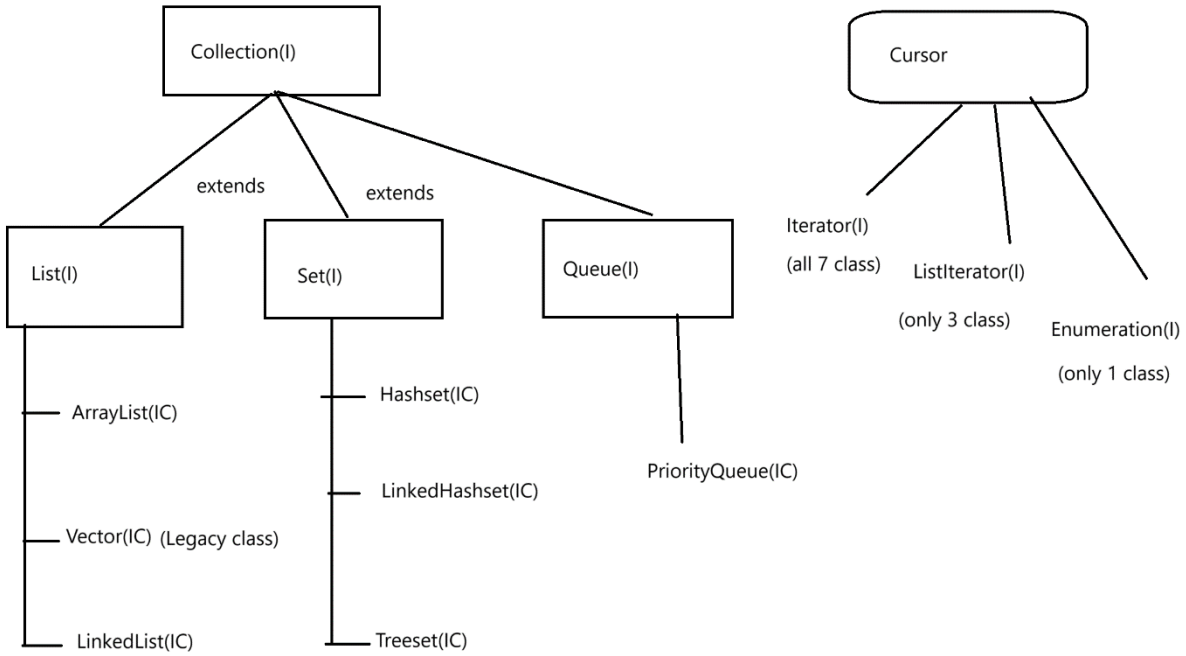
```
import java.util.TreeSet;
public class ex6_Treeset1
{
	public static void main(String[] args)
	{
		TreeSet ts=new TreeSet();
		ts.add("rahul");
		ts.add("mahesh");
		ts.add("ramesh");
		ts.add("suresh");
		ts.add("ganesh");
		ts.add("ramesh");
		//ts.add(null);    //nullPointerException

		System.out.println(ts);
	}
}
```

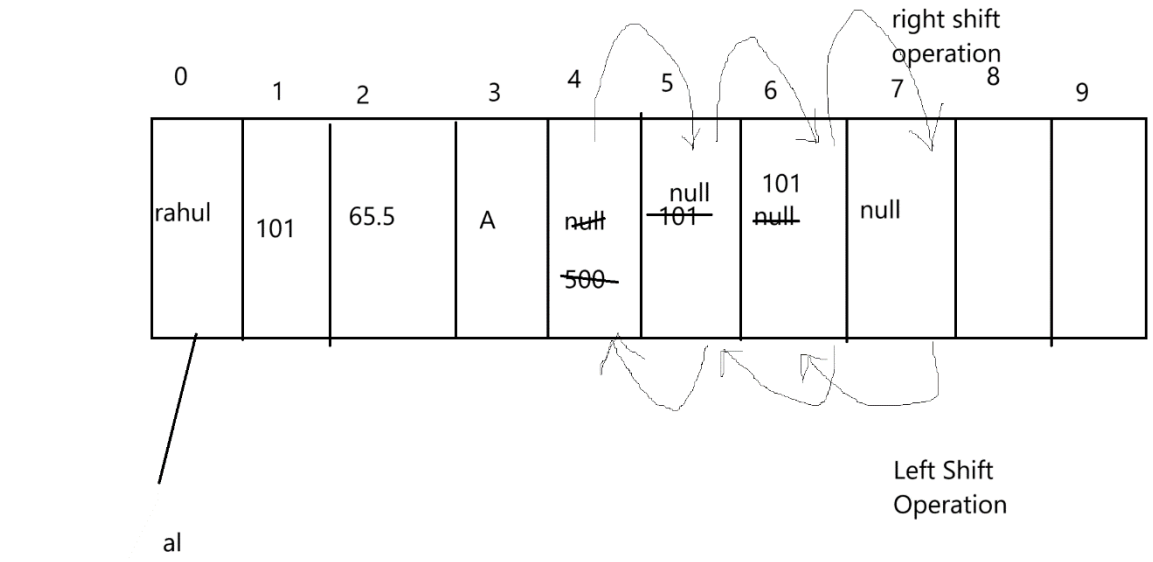
```
package Collection;
import java.util.Iterator;
import java.util.TreeSet;
public class ex6_Treeset2
{
	public static void main(String[] args)
	{
		TreeSet ts=new TreeSet();
		ts.add(104);
		ts.add(105);
		ts.add(101);
		ts.add(103);
		ts.add(102);
		ts.add(107);
		ts.add(106);
		ts.add("ubfasf");

		System.out.println(ts);
		System.out.println(ts.size());
		System.out.println(ts.contains(101));

		ts.remove(104);
		System.out.println(ts);

		System.out.println(ts.first());   //get first data
		System.out.println(ts.last());    //get last data

		ts.pollFirst();        //delete data from 1st position
		System.out.println(ts);

		ts.pollLast();                                  //delete data from last position
		System.out.println(ts);

		System.out.println("-----Print all data:  using for each loop------");
		for(Object s1:ts)
		{
			System.out.println(s1);
		}


		System.out.println("-----Print all data:  using Iterator cursor------");
		Iterator itr = ts.iterator();
		while(itr.hasNext())
		{
			System.out.println(itr.next());
		}

		System.out.println("-----Print all data:  using Descending Iterator cursor------");
		Iterator ditr = ts.descendingIterator();
		while(ditr.hasNext())
		{
			System.out.println(ditr.next());
		}
```

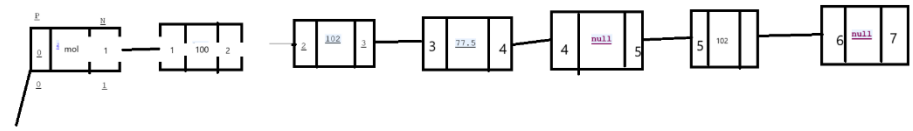
```
	}
}
```

```
package Collection;
import java.util.ArrayList;
public class ex7_generic1
{
	public static void main(String[] args)
	{
		ArrayList<String> al=new ArrayList<String>();
		al.add("mahesh");
		al.add("ganesh");
		al.add("suresh");
		al.add("suresh");

		System.out.println(al);


		for(String s1:al)
		{
			System.out.println(s1);
		}
	}
}
```

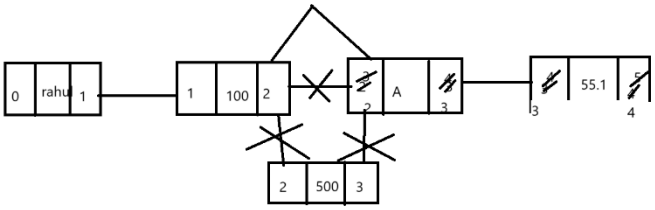
```
package Collection;
import java.util.Iterator;
import java.util.TreeSet;
public class ex7_generic2
{
	public static void main(String[] args)
	{
		TreeSet<Integer> ts=new TreeSet<Integer>();
		ts.add(104);
		ts.add(105);
		ts.add(101);
		ts.add(103);
		ts.add(102);
		ts.add(107);
		ts.add(106);


		for(Integer num:ts)
		{
			System.out.println(num);
		}
	}
}
```


Collection(I)
extends
extends
List(I)
Set(I)
Queue(I)
ArrayList(IC)
Vector(IC) (Legacy class)
LinkedList(IC)
Hashset(IC)
LinkedHashset(IC)
Treeset(IC)
PriorityQueue(IC)
Cursor
Iterator(I)
(all 7 class)
ListIterator(I)
(only 3 class)
Enumeration(I)
(only 1 class)


right shift
operation
0
1
2
3
4
5
6
7
8
9
rahul
101
65.5
A
null
500
null
101
101
null
null
Left Shift
Operation
al


F
N
0
mol
1
1
100
2
0
1
2
102
3
3
27.5
4
4
null
5
5
102
6
null
7

DS= linear


0
rahul
1
1
100
2
2
A
3
3
55.1
4
2
500
3