Program on Stack class :

```java
package com.ravi.stack;

import java.util.EmptyStackException;
import java.util.Stack;

public class StackDemo1
{
      public static void main(String args[])
      {
            Stack<Integer> s = new Stack<>();

            try
            {
                s.push(12);
                s.push(15);
               s.push(22);
               s.push(33);
               s.push(49);
               System.out.println("After insertion elements are :"+s);

                 System.out.println("Fetching the elements using pop method");
                 System.out.println(s.pop());
                 System.out.println(s.pop());
                 System.out.println(s.pop());
                 System.out.println(s.pop());
                 System.out.println(s.pop());


               System.out.println("After deletion elements are :"+s);//[]

               System.out.println("Is the Stack empty ? :"+s.empty());
             }
          catch(EmptyStackException e)
          {
             e.printStackTrace();
          }

      }
}
```

```java
package com.ravi.stack;

//add() is Vector class method

import java.util.*;
public class StackDemo2
{
      public static void main(String args[])
      {
            Stack<Integer> st1 = new Stack<>();
            st1.add(10);
            st1.add(20);
            st1.forEach(x -> System.out.println(x));

            Stack<String> st2 = new Stack<>();
            st2.add("Java");
            st2.add("is");
            st2.add("programming");
            st2.add("language");
            st2.forEach(x -> System.out.println(x));

            Stack<Character> st3 = new Stack<>();
            st3.add('A');
            st3.add('B');
            st3.forEach(x -> System.out.println(x));

            Stack<Double> st4 = new Stack<>();
            st4.add(10.5);
            st4.add(20.5);
            st4.forEach(x -> System.out.println(x));
      }
}
```

```java
package com.ravi.stack;

import java.util.Stack;

public class StackDemo3
{
    public static void main(String[] args)
        {
            Stack<String> stk= new Stack<>();
            stk.push("Apple");
            stk.push("Grapes");
            stk.push("Mango");
            stk.push("Orange");
            System.out.println("Stack: " + stk);

            String fruit = stk.peek();
            System.out.println("Element at top: " + fruit);
            System.out.println("Stack elements are : " + stk);
        }
}
```

```java
package com.ravi.stack;

import java.util.Stack;

public class StackDemo4
{
    public static void main(String[] args) //
        {
            Stack<String> stk= new Stack<>();
            stk.push("Apple");
            stk.push("Grapes");
            stk.push("Mango");
            System.out.println("Offset Position is : " + stk.search("Mango")); //1
            System.out.println("Offser Position is : " + stk.search("Banana")); //-1
             System.out.println("Is stack empty ? "+stk.empty());  //false
            System.out.println("Index Position is : " + stk.indexOf("Mango")); //2

        }
}
```

ArrayList<E>

----

public class ArrayList<E> extends AbstractList<E> implements List<E>, **Serializable, Cloneable**, RandomAccess

* It is an implemented class of List<E> interface available from JDK 1.2V
* It can accept duplicate, null, homogeneous and hetrogeneous elements.
* It stores the element based on the index position.
* It uses **dynamic array structure so dynamically it can grow and shrink.**
* It is a dynamically Growable array.
* Default capacity is 10, next capacity would be 16 [almost 50% increment]
  next capacity can be calculated by using following formula :

$$\text{next capacity} = \frac{(\text{current capacity} * 3)}{2} + 1$$

$$= 16$$

* Methods are not synchronized so performance wise it is better than Vector.
* Iterators are Fail Fast Iterator.
* hashCode() and equals() methods are overridden.
* It is mainly used to retrieve the elements when thread safety is not required.
* It implements List, Serializable (can store the object in a file), Cloneable (can create a duplicate arraylist object) and RandomAccess (Can randonly access the elements based on index position)

Constructors of ArrayList :

In ArrayList we have 3 types of Constructor:
Constructor of ArrayList :

1) ArrayList al1 = new ArrayList();
   Will create ArrayList object with default capacity 10.

2) ArrayList al2 = new ArrayList(int initialCapacity);
   Will create an ArrayList object with user specified Capacity

3) ArrayList al3 = new ArrayList(Collection c)
   We can copy any Collection interface implemented class data to the current object   reference
(Coping one Collection data to another)

Removing Duplicates from the ArrayList :

```java
package com.ravi.arraylist;

import java.util.ArrayList;
import java.util.HashSet;

public class RemoveDuplicates
{
    public static void main(String[] args)
    {
        ArrayList<String> listOfFruits = new ArrayList<>();
        listOfFruits.add("Apple");
        listOfFruits.add("Apple");
        listOfFruits.add("Orange");
        listOfFruits.add("Mango");
        listOfFruits.add("Papaya");
        listOfFruits.forEach(System.out::println);

        System.out.println("Removing Duplicates");

        HashSet<String> fruits = new HashSet<>(listOfFruits);
        fruits.forEach(System.out::println);
    }

}
```

```java
package com.ravi.arraylist;

import java.util.ArrayList;

//Add all the elements of ArrayList
public class ArrayListDemo
{
     public static void main(String[] args)
     {
         ArrayList<Integer> numbers = new ArrayList<>(100);

         numbers.add(100);
         numbers.add(200);
         numbers.add(300);
         numbers.add(400);

         int sum = 0;

         for (int number : numbers)
         {
             sum += number;
         }
         System.out.println("Sum of numbers: " + sum);
     }
}
```

```java
package com.ravi.arraylist;


import java.util.ArrayList;
import java.util.Collections;

record Customer(Integer custId, String custName, Double custSal) implements Comparable<Customer>
{

    @Override
    public int compareTo(Customer c2)
    {
        return this.custName.compareTo(c2.custName);
    }
}

public class ArrayListDemo1
{
    public static void main(String[] args)
    {
        ArrayList<Customer> listOfCustomer = new ArrayList<>();
        listOfCustomer.add(new Customer(111, "Scott", 800D));
        listOfCustomer.add(new Customer(222, "Smith", 1200D));
        listOfCustomer.add(new Customer(333, "Alen", 1800D));
        listOfCustomer.add(new Customer(444, "Martin", 1500D));
        listOfCustomer.add(new Customer(555, "John", 1300D));

        System.out.println("Original Customer Object :");
        listOfCustomer.forEach(System.out::println);


        Collections.sort(listOfCustomer);
        System.out.println("Sorted based on the name :");
        listOfCustomer.forEach(System.out::println);
    }
}
```

```java
package com.ravi.arraylist;

//addAll() used to merge two collection + retainAll() [Common Element]

import java.util.*;
public class ArrayListDemo2
    {
        public static void main(String args[])
        {
          ArrayList<String> al1=new ArrayList<>();
          al1.add("Ravi");
          al1.add("Rahul");
          al1.add("Rohit");

          ArrayList<String> al2=new ArrayList<>();
          al2.add("Pallavi");
          al2.add("Sweta");
          al2.add("Puja");

          al1.addAll(al2);

          System.out.println("Size of al1 :"+al1.size());

           al1.forEach(str -> System.out.println(str.toUpperCase()) );

         System.out.println("................................");

          ArrayList<String> al3=new ArrayList<>();
          al3.add("Ravi");
          al3.add("Rahul");
          al3.add("Rohit");

          ArrayList<String> al4=new ArrayList<>();
          al4.add("Pallavi");
          al4.add("Rahul");
          al4.add("Raj");

          al3.retainAll(al4);

           al3.forEach(x -> System.out.println(x));
    }
 }
```

---

* How to create a Fixed length array and Immutable List :

----

a) Fixed length Array :

----

java.util.Arrays class has provided a predefined static method
asList(T ...x), It will create a fixed length array and the return type of this method is List interface.

In this fixed length array we can't perform add or remove opeartion otherwise we will get java.lang.UnsupportedOperationException but we can replace the existing element.

```java
package com.ravi.arraylist;

import java.util.Arrays;
import java.util.List;

public class FixedLengthArray
{
    public static void main(String[] args)
    {
        List<Integer> fixedLengthArray = Arrays.asList(1,2,3,4,5);
        fixedLengthArray.forEach(System.out::println);

        System.out.println("......................");

        //fixedLengthArray.add(6); // java.lang.UnsupportedOperationException
        //fixedLengthArray.remove(0); // java.lang.UnsupportedOperationException

        fixedLengthArray.set(0, 100);
        fixedLengthArray.forEach(System.out::println);
    }
}
```

b) Immutable List :

List interface has provided a predefined static method called
of(T ...x) available from java 9V.

It will create an immutable list, return type of this method is
List<E>. Once it is created after that we can't pefrom any kind of operation like add(), remove() or replace [set(int index, Object obj)] otherwise we will get java.lang.UnsupportedOperationException

```java
package com.ravi.arraylist;

import java.util.List;


public class ImmutableList {

    public static void main(String[] args)
    {
        List<Integer> immutableList = List.of(1,2,3,4,5,6,7,8,9,10,11);
        immutableList.forEach(System.out::println);

        //immutableList.add(12); // java.lang.UnsupportedOperationException
        //immutableList.remove(0); // java.lang.UnsupportedOperationException
        //immutableList.set(0, 100); // java.lang.UnsupportedOperationException

    }

}
```

```java
package com.ravi.arraylist;


import java.util.Arrays;
import java.util.Collections;
import java.util.List;
import java.util.ListIterator;

public class ArrayListDemo3
{
public static void main(String args[])
  {
     List<String> listOfName = Arrays.asList("Rohit","Akshar","Pallavi","Sweta"); //Length is
fixed

     Collections.sort(listOfName);

     //Fetching the data in both the direction
     ListIterator<String> lst = listOfName.listIterator();

     System.out.println("In Forward Direction..");
     while(lst.hasNext())
     {
        System.out.println(lst.next());
     }

     System.out.println("In Backward Direction..");
     while(lst.hasPrevious())
     {
        System.out.println(lst.previous());
     }

  }
}
```

```java
package com.ravi.arraylist;

//Serialization + De-Serialization

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.ArrayList;

public class ArrayListDemo4
{
  public static void main(String[] args) throws IOException
  {
    ArrayList<String> listOfIceCream = new ArrayList<>();
    listOfIceCream.add("Vanila");
    listOfIceCream.add("Strwbarry");
    listOfIceCream.add("Butter Scotch");

    //Serialization

    String filePath = "D:\\new\\IceCream.txt";

    var fos = new FileOutputStream(filePath);
    var oos = new ObjectOutputStream(fos);

    try(fos; oos)
    {
       oos.writeObject(listOfIceCream);
       System.out.println("Data Stored Successfully");
    }
    catch(Exception e)
    {
       e.printStackTrace();
    }

  //De-Serialization
      var fin = new FileInputStream(filePath);
      var ois = new ObjectInputStream(fin);

      try(fin; ois)
      {
         @SuppressWarnings("unchecked")
         ArrayList<String> list = (ArrayList<String>) ois.readObject();
         System.out.println(list);
      }
      catch(Exception e)
      {
         e.printStackTrace();
      }
    }
  }
```