```
import java.util.Random;
/**
 * Class LinearArray provides several static operation
 * on linear arrays of double. Methods add(), subtract()
 * multiply() and clone() return a new array, while
 * method fillRandomData() modifies the array that is passed
 * as an argument to them.
 *
 * @author (Antonios Symvonis)
 * @version (11 April 2002)
 */
public  class LinearArray
{
    /**
     * Fills a  linear array with radmon data in the range [0..1). The ra
ndom
     * numbers have precision of 2 decimal digits.
     *
     * @param  a        The  array to be filled. It cannot be NULL.
     * @return          The result of the multiplication.
     */
    public static void fillRandomData(double a[])
    {
        Random random= new Random(0);
        int len=a.length;

        for (int i=0; i< len; i++)
            a[i]= (double) random.nextInt(100)/100;
    }//fillRandomData

    /**
     * Prints the elements of an array, all at the same line.
     *
     * @param  a   The array of which the ellements are printed.
     */
    public static void printH(double a[])
    {
        int len=a.length;
        System.out.print("{");
        for( int i=0; i< len-1; i++)
            System.out.print(a[i] + ", " );
        System.out.println(a[len-1] +"}");
    }//printH


    /**
     * Prints the elements of an array, each at a different line.
     *
     * @param  a   The array of which the ellements are printed.
     */
    public static void printV(double a[])
    {
        int len=a.length;
```

```
        for( int i=0; i< len; i++)
            System.out.println(a[i]);
        System.out.println();
    }//printV


    /**
     * Creates an exact copy of an array.
     *
     * @param  a    The array to be copied. It cannot be NULL.
     * @return      An exact copy of the argument array.
     */
    public static double[] clone(double[] a)
    {
        double clone[]=new double[a.length];
        for(int i=0; i< a.length; i++)
            clone[i]=a[i];
        return clone;
    }



    /**
     * Implements linear array addition. The two arrays to be added must
be of
     * equal length.
     *
     * @param  a1    The first array to be added. It cannot be NULL.
     * @param  a2    The second array to be added. It cannot be NULL.
     * @return       The sum of the argument arrays.
     */
    public static double[] add(double a1[], double a2[])
    {
        double sum[]= new double[a1.length];
        for (int i=0; i<sum.length; i++)
            sum[i]=a1[i]+a2[i];
        return sum;
    }




    /**
     * Implements linear array subtraction. The two arrays which particip
ate  must be of
     * equal length.
     *
     * @param  a1    The  array from which we subtract. It cannot be NULL
.
     * @param  a2    The  array to be subtracted. It cannot be NULL.
     * @return       The result of the subtraction.
     */
   public static double[] subtract(double a1[], double a2[])
```

```
    {
        double result[]= new double[a1.length];
        for (int i=0; i<a1.length; i++)
            result[i]=a1[i]-a2[i];
        return result;
    }


    /**
     * Implements scalar linear array multiplication.
     *
     * @param  factor   The  factor of the multiplication.
     * @param  a        The  array to be multiplied. It cannot be NULL.
     * @return          The result of the multiplication.
     */
    public static double[] multiply(double factor, double a[])
   {
        double result[]= new double[a.length];
        for (int i=0; i<a.length; i++)
            result[i]=factor*a[i];
        return result;
    }


    /**
     * Starting from a specified position of an array, it identifies the
     * location of the maximum element of the array. The specified positi
on is
     * included in the search.
     *
     * @param  a        The array to be searched.
     * @param  start    The search for the maximum starts from this posit
ion.
     * @return          The index of the maximum element.
     */
    public static int maxLocationFrom(double[] a,int pos)
    {
        int maxPos=pos;
        for ( int i=pos+1; i<a.length; i++)
        if( a[maxPos]<a[i])
            maxPos=i;
        return maxPos;
       }


    /**
     * Starting from a specified position of an array, it identifies the
     * location of the minimum element of the array. The specified positi
on is
     * included in the search.
     *
     * @param  a        The array to be searched.
     * @param  start    The search for the minimum starts from this posit
```

```
ion.
     * @return          The index of the minimum element.
     */
    public static int minLocationFrom(double[] a,int pos)
    {
        int minPos=pos;
        for ( int i=pos+1; i<a.length; i++)
        if( a[minPos]>a[i])
            minPos=i;
        return minPos;
     }
  // ENALAKTIKH EKDOXH THS minLocationFrom
  //   {
  //        double[] b=multiply(-1,a);
  //        return maxLocationFrom(b,pos);
  //   }

}
```

```
/**
 * Used to test the static method of class Linear Array.
 *
 * @author (Antonios Symvonis)
 * @version (11 April 2002)
 */
public class TestLinearArray
{
    public static void main(String args[])
    {

        //arrays used in testing the methods of class Linear Array
        double data1[]={1,3,5,7,9};
        double data2[]={2,4,6,8,10};
        double data3[]=new double[10];

        //test fillRandomData
        LinearArray.fillRandomData(data3);

        //print test data
        System.out.println("-----------------------------------------
-");
        System.out.println("The test arrays are:\n");
        System.out.print("data1[]:  ");
        LinearArray.printH(data1);
        System.out.print("data2[]:  ");
        LinearArray.printH(data2);
        System.out.print("data3[]:  ");
        LinearArray.printH(data3);
        System.out.println();


        //test addition, subtraction, multiplication
        double r1[]=LinearArray.add(data1,data2);
       System.out.print("data1[] + data2[]:  ");
        LinearArray.printH(r1);

        double r2[]=LinearArray.subtract(data1,data2);
        System.out.print("data1[] - data2[]:  ");
        LinearArray.printH(r2);

        double r3[]=LinearArray.multiply(2,data1);
        System.out.print("2*data1[]:  ");
        LinearArray.printH(r3);

        //test maxLocationFrom, minLocationFrom
        System.out.println("maxLocationFrom(data3,0): " + LinearArray.max
LocationFrom(data3,0));
        System.out.println("maxLocationFrom(data3,8): " + LinearArray.max
LocationFrom(data3,8));
        System.out.println("minLocationFrom(data3,0): " + LinearArray.min
LocationFrom(data3,0));
```

```
        System.out.println("minLocationFrom(data3,5): " + LinearArray.min
LocationFrom(data3,5));

        //test clone
        double r4[]=LinearArray.clone(data3);
        System.out.print("r4 = clone(data3[]):  ");
        LinearArray.printH(r4);

    }//main
}
```