```
package ouicksort;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.Arrays;
import java.util.Random;

public class OuickSort {

   /**
    * @param args the command line arguments
    */

   // Function to partion the array on the basis of the pivot value;
   static int partition(int[] array, int low, int high) {
      int j, temp, i = low + 1;
      Random random = new Random();
      int x = random.nextInt(high - low) + low;
      temp = array[low];
      array[low] = array[x];
      array[x] = temp;
      for (j = low + 1; j <= high; j++) {
         if (array[j] <= array[low] && j != i) {
            temp = array[j];
            array[j] = array[i];
            array[i++] = temp;
         } else if (array[j] <= array[low]) {
            i++;
         }
      }
      temp = array[i - 1];
      array[i - 1] = array[low];
      array[low] = temp;
      return i - 1;
   }
   // Function to implement quick sort
   static void quickSort(int[] array,int low,int high){
      if(low<high){
         int mid = partition(array,low,high);
         System.out.print(mid);
         quickSort(array,low,mid-1);
         quickSort(array,mid+1,high);
      }
   }


   public static void main(String[] args) {
```

```
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        int size;
        System.out.println("Dizinin boyutunu giriniz");
        try {
            size = Integer.parseInt(br.readLine());
        } catch (Exception e) {
            System.out.println("geçersiz giriş");
            return;
        }
        int[] array = new int[size];
        System.out.println("Dizinin elemanları");
        int i;
        for (i = 0; i < array.length; i++) {
            try {
                array[i] = Integer.parseInt(br.readLine());
            } catch (Exception e) {
                System.out.println("Hata meydana geldi");
            }
        }
        System.out.println("Girilen Dizi");
        System.out.println(Arrays.toString(array));
        quickSort(array,0,array.length-1);
        System.out.println("Sıralanmış dizi");
        System.out.println(Arrays.toString(array));

    }

}


package aramaalgoritmalari;

import java.util.Random;
import java.util.Scanner;

public class AramaAlgoritmalari {

    public static int N = 10000000;
    public static int[] sequence = new int[N];

    public static boolean sequentialSearch(int[] sequence, int key)
    {
        for (int i = 0; i < sequence.length; i++)
            if (sequence[i] == key)
                return true;
        return false;
    }
```

```
public static boolean binarySearch(int[] sequence, int key)
{
   int low = 0, high = sequence.length - 1;
   while (low <= high)
   {
      int mid = (low + high) / 2;
      if (key < sequence[mid])
         high = mid - 1;
      else if (key > sequence[mid])
         low = mid + 1;
      else
         return true;
   }
   return false;
}

public static void QuickSort(int left, int right)
{
   if (right - left <= 0)
      return;
   else
   {
      int pivot = sequence[right];
      int partition = partitionIt(left, right, pivot);
      QuickSort(left, partition - 1);
      QuickSort(partition + 1, right);
   }
}

public static int partitionIt(int left, int right, long pivot)
{
   int leftPtr = left - 1;
   int rightPtr = right;
   while (true)
   {
      while (sequence[++leftPtr] < pivot)
         ;
      while (rightPtr > 0 && sequence[--rightPtr] > pivot)
         ;

      if (leftPtr >= rightPtr)
         break;
      else
         swap(leftPtr, rightPtr);
   }
   swap(leftPtr, right);
```

```
    return leftPtr;
}

public static void swap(int dex1, int dex2)
{
    int temp = sequence[dex1];
    sequence[dex1] = sequence[dex2];
    sequence[dex2] = temp;
}

public static void main(String[] args) {
    Random random = new Random();

    for (int i = 0; i < N; i++)
        sequence[i] = Math.abs(random.nextInt(100));

    Scanner sc = new Scanner(System.in);
    System.out.println("Aranacak değeri giriniz: ");
    int k = sc.nextInt();

    System.out
            .println("Ardışıl arama için geçen süre: ");
    long startTime = System.nanoTime();
    boolean result = sequentialSearch(sequence, k);
    long endTime = System.nanoTime();

    if (result == true)
        System.out.println("Süre:  " + (endTime - startTime)
                + " nanoseconds");
    else
        System.out.println("Mevcut değil.. Süre:  "
                + (endTime - startTime) + " nanoseconds");

    System.out.println("ikili (binary) search için süre: ");
    long startTime1 = System.nanoTime();
    QuickSort(0, N - 1);

    result = binarySearch(sequence, k);
    long endTime1 = System.nanoTime();

    if (result == true)
        System.out.println("Süre: " + (endTime1 - startTime1)
                + " nanoseconds");
    else
        System.out.println("mevcut değil geçen süre "
                + (endTime1 - startTime1) + " nanoseconds");
    sc.close();
```

```
    }

}
```