```java
import java.io.*;
import java.util.*;

class SOS
{
    public static void main(String [] args) throws IOException
    {
        //prompts user to enter their list size and reads in response
        System.out.println("Please enter the size of list you want to sort: ");
        Scanner cin=new Scanner(System.in);
        int size=cin.nextInt();

        //fills array with random numbers
        Integer [] list;
        list=new Integer[size];
        for(int i=0; i<list.length; i++)
            { list[i]=new Integer((int)(list.length*Math.random())); }

        //prints out array if there are less than 100 numbers
        if(size<=100)
        {
            System.out.println("Random array: ");
            for(int i=0; i<list.length; i++)
                { System.out.print(list[i]+" "); }
            System.out.println();
        }

        //declaring variables for run time
        long timeBefore=0;
        long timeAfter=0;
        long runTime=0;

        //bubble sort
        //copies array
        Integer [] listCopy;
        listCopy=new Integer[size];
        for(int i=0; i<list.length; i++)
            { listCopy[i]=list[i]; }

        //calls running time, sorts, then checks running time agter
        System.out.println("\n* Bubble Sort *");
        timeBefore=System.currentTimeMillis();
        Sorts.bubble(listCopy);
        timeAfter=System.currentTimeMillis();
        //subtracts time after and before to find running time
        runTime=timeAfter-timeBefore;
        System.out.println("Run time: "+runTime);

        //prints sorted array if less than 100 numbers
        if(size<=100)
        {
```

```
    for(int i=0; i<listCopy.length; i++)
        { System.out.print(listCopy[i]+" "); }
    System.out.println();
}
//notes for all sorts are the same

//insertion
Integer [] listCopy2;
listCopy2=new Integer[size];
for(int i=0; i<list.length; i++)
    { listCopy2[i]=list[i]; }

System.out.println("\n* Insertion Sort *");
timeBefore=System.currentTimeMillis();
Sorts.insertion(listCopy2);
timeAfter=System.currentTimeMillis();
runTime=timeAfter-timeBefore;
System.out.println("Run time: "+runTime);

if(size<=100)
{
    for(int i=0; i<listCopy2.length; i++)
        { System.out.print(listCopy2[i]+" "); }
    System.out.println();
}

//selection
Integer [] listCopy3;
listCopy3=new Integer[size];
for(int i=0; i<list.length; i++)
    { listCopy3[i]=list[i]; }

System.out.println("\n* Selection Sort *");
timeBefore=System.currentTimeMillis();
Sorts.selection(listCopy3);
timeAfter=System.currentTimeMillis();
runTime=timeAfter-timeBefore;
System.out.println("Run time: "+runTime);

if(size<=100)
{
    for(int i=0; i<listCopy3.length; i++)
        { System.out.print(listCopy3[i]+" "); }
    System.out.println();
}

//quick
Integer [] listCopy4;
listCopy4=new Integer[size];
for(int i=0; i<list.length; i++)
    { listCopy4[i]=list[i]; }
```

```
        System.out.println("\n* Quick Sort *");
        timeBefore=System.currentTimeMillis();
        Sorts.quick(listCopy4);
        timeAfter=System.currentTimeMillis();
        runTime=timeAfter-timeBefore;
        System.out.println("Run time: "+runTime);

        if(size<=100)
        {
            for(int i=0; i<listCopy4.length; i++)
                { System.out.print(listCopy4[i]+" "); }
            System.out.println();
        }

        //shell
        Integer [] listCopy5;
        listCopy5=new Integer[size];
        for(int i=0; i<list.length; i++)
            { listCopy5[i]=list[i]; }

        System.out.println("\n* Shell Sort *");
        timeBefore=System.currentTimeMillis();
        Sorts.shell(listCopy5);
        timeAfter=System.currentTimeMillis();
        runTime=timeAfter-timeBefore;
        System.out.println("Run time: "+runTime);

        if(size<=100)
        {
            for(int i=0; i<listCopy5.length; i++)
                { System.out.print(listCopy5[i]+" "); }
            System.out.println();
        }
    }
}
```

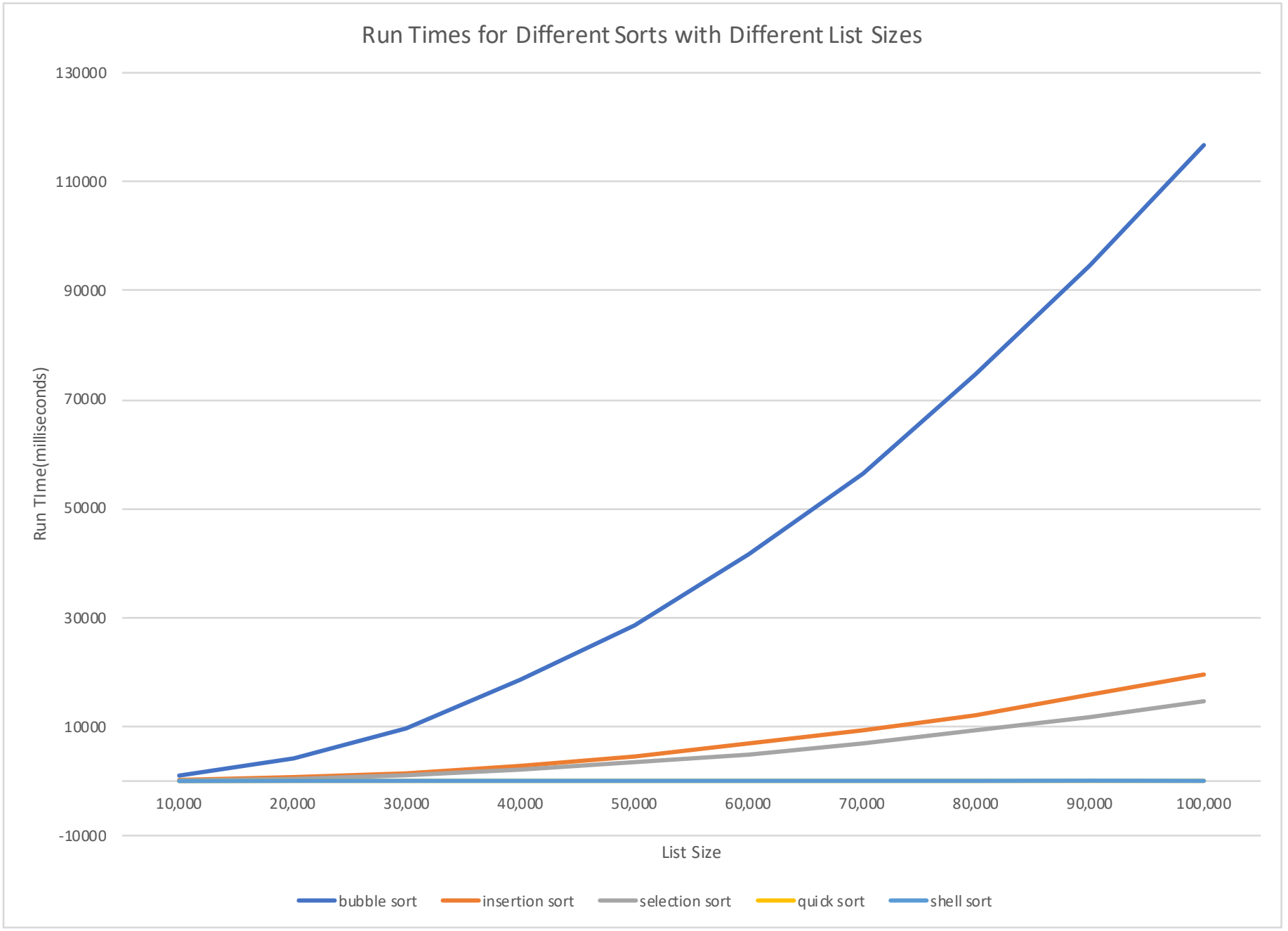
```
thomas% javac SOS.java
thomas% java SOS
Please enter the size of list you want to sort:
1000

* Bubble Sort *
Run time: 23

* Insertion Sort *
Run time: 8

* Selection Sort *
Run time: 9

* Quick Sort *
Run time: 1

* Shell Sort *
Run time: 1
thomas% clear




thomas% javac SOS.java
thomas% java SOS
Please enter the size of list you want to sort:
10000

* Bubble Sort *
Run time: 1011

* Insertion Sort *
Run time: 219

* Selection Sort *
Run time: 108

* Quick Sort *
Run time: 5

* Shell Sort *
Run time: 6
thomas% java SOS
Please enter the size of list you want to sort:
20000

* Bubble Sort *
Run time: 4229

* Insertion Sort *
Run time: 673

* Selection Sort *
Run time: 442

* Quick Sort *
Run time: 10

* Shell Sort *
Run time: 22
thomas% java SOS
Please enter the size of list you want to sort:
30000

* Bubble Sort *
```

```
Run time: 9804

* Insertion Sort *
Run time: 1563

* Selection Sort *
Run time: 1096

* Quick Sort *
Run time: 15

* Shell Sort *
Run time: 28
thomas% java SOS
Please enter the size of list you want to sort:
40000

* Bubble Sort *
Run time: 18481

* Insertion Sort *
Run time: 2919

* Selection Sort *
Run time: 2153

* Quick Sort *
Run time: 18

* Shell Sort *
Run time: 42
thomas% java SOS
Please enter the size of list you want to sort:
50000

* Bubble Sort *
Run time: 28667

* Insertion Sort *
Run time: 4625

* Selection Sort *
Run time: 3386

* Quick Sort *
Run time: 20

* Shell Sort *
Run time: 56
thomas% java SOS
Please enter the size of list you want to sort:
60000

* Bubble Sort *
Run time: 41556

* Insertion Sort *
Run time: 6801

* Selection Sort *
Run time: 5007

* Quick Sort *
Run time: 28

* Shell Sort *
```

```
Run time: 45
thomas% java SOS
Please enter the size of list you want to sort:
70000

* Bubble Sort *
Run time: 56543

* Insertion Sort *
Run time: 9440

* Selection Sort *
Run time: 6936

* Quick Sort *
Run time: 34

* Shell Sort *
Run time: 65
thomas% java SOS
Please enter the size of list you want to sort:
80000

* Bubble Sort *
Run time: 74681

* Insertion Sort *
Run time: 12073

* Selection Sort *
Run time: 9163

* Quick Sort *
Run time: 42

* Shell Sort *
Run time: 67
thomas% java SOS
Please enter the size of list you want to sort:
90000

* Bubble Sort *
Run time: 94725

* Insertion Sort *
Run time: 15986

* Selection Sort *
Run time: 11734

* Quick Sort *
Run time: 38

* Shell Sort *
Run time: 68
thomas% java SOS
Please enter the size of list you want to sort:
100000

* Bubble Sort *
Run time: 116727

* Insertion Sort *
Run time: 19558

* Selection Sort *
```

```
Run time: 14653

* Quick Sort *
Run time: 47

* Shell Sort *
Run time: 64
thomas% java SOS
Please enter the size of list you want to sort:
100
Random array:
24 87 66 60 34 36 10 96 12 1 31 37 46 88 57 99 65 65 67 21 92 34 93 36 79 49 58 90 81 48 77 63 34 40
85 20 74 1 6 93 8 89 7 82 38 62 7 5 54 34 25 8 10 85 50 46 82 1 33 51 83 72 50 20 0 39 7 18 27 84 82
16 5 30 96 81 3 61 23 20 39 94 8 61 38 79 85 1 36 77 7 63 66 31 16 72 27 68 73 5

* Bubble Sort *
Run time: 3
0 1 1 1 1 3 5 5 5 6 7 7 7 7 8 8 8 10 10 12 16 16 18 20 20 20 21 23 24 25 27 27 30 31 31 33 34 34 34 3
4 36 36 36 37 38 38 39 39 40 46 46 48 49 50 50 51 54 57 58 60 61 61 62 63 63 65 65 66 66 67 68 72 72
73 74 77 77 79 79 81 81 82 82 82 83 84 85 85 85 87 88 89 90 92 93 93 94 96 96 99

* Insertion Sort *
Run time: 0
0 1 1 1 1 3 5 5 5 6 7 7 7 7 8 8 8 10 10 12 16 16 18 20 20 20 21 23 24 25 27 27 30 31 31 33 34 34 34 3
4 36 36 36 37 38 38 39 39 40 46 46 48 49 50 50 51 54 57 58 60 61 61 62 63 63 65 65 66 66 67 68 72 72
73 74 77 77 79 79 81 81 82 82 82 83 84 85 85 85 87 88 89 90 92 93 93 94 96 96 99

* Selection Sort *
Run time: 0
0 1 1 1 1 3 5 5 5 6 7 7 7 7 8 8 8 10 10 12 16 16 18 20 20 20 21 23 24 25 27 27 30 31 31 33 34 34 34 3
4 36 36 36 37 38 38 39 39 40 46 46 48 49 50 50 51 54 57 58 60 61 61 62 63 63 65 65 66 66 67 68 72 72
73 74 77 77 79 79 81 81 82 82 82 83 84 85 85 85 87 88 89 90 92 93 93 94 96 96 99

* Quick Sort *
Run time: 0
0 1 1 1 1 3 5 5 5 6 7 7 7 7 8 8 8 10 10 12 16 16 18 20 20 20 21 23 24 25 27 27 30 31 31 33 34 34 34 3
4 36 36 36 37 38 38 39 39 40 46 46 48 49 50 50 51 54 57 58 60 61 61 62 63 63 65 65 66 66 67 68 72 72
73 74 77 77 79 79 81 81 82 82 82 83 84 85 85 85 87 88 89 90 92 93 93 94 96 96 99

* Shell Sort *
Run time: 0
0 1 1 1 1 3 5 5 5 6 7 7 7 7 8 8 8 10 10 12 16 16 18 20 20 20 21 23 24 25 27 27 30 31 31 33 34 34 34 3
4 36 36 36 37 38 38 39 39 40 46 46 48 49 50 50 51 54 57 58 60 61 61 62 63 63 65 65 66 66 67 68 72 72
73 74 77 77 79 79 81 81 82 82 82 83 84 85 85 85 87 88 89 90 92 93 93 94 96 96 99
thomas%
```

	bubble sort	insertion sort	selection sort	quick sort	shell sort
10,000	1011	219	108	5	6
20,000	4229	673	442	10	22
30,000	9804	1563	1096	15	28
40,000	18481	2919	2153	18	42
50,000	28667	4625	3386	20	56
60,000	41556	6801	5007	28	45
70,000	56543	9440	6936	34	65
80,000	74681	12073	9163	42	67
90,000	94725	15986	11734	38	68
100,000	116727	19558	14653	47	64


Run Times for Different Sorts with Different List Sizes
130000
110000
90000
70000
50000
30000
10000
-10000
Run TIme(milliseconds)
10,000
20,000
30,000
40,000
50,000
60,000
70,000
80,000
90,000
100,000
List Size
bubble sort
insertion sort
selection sort
quick sort
shell sort