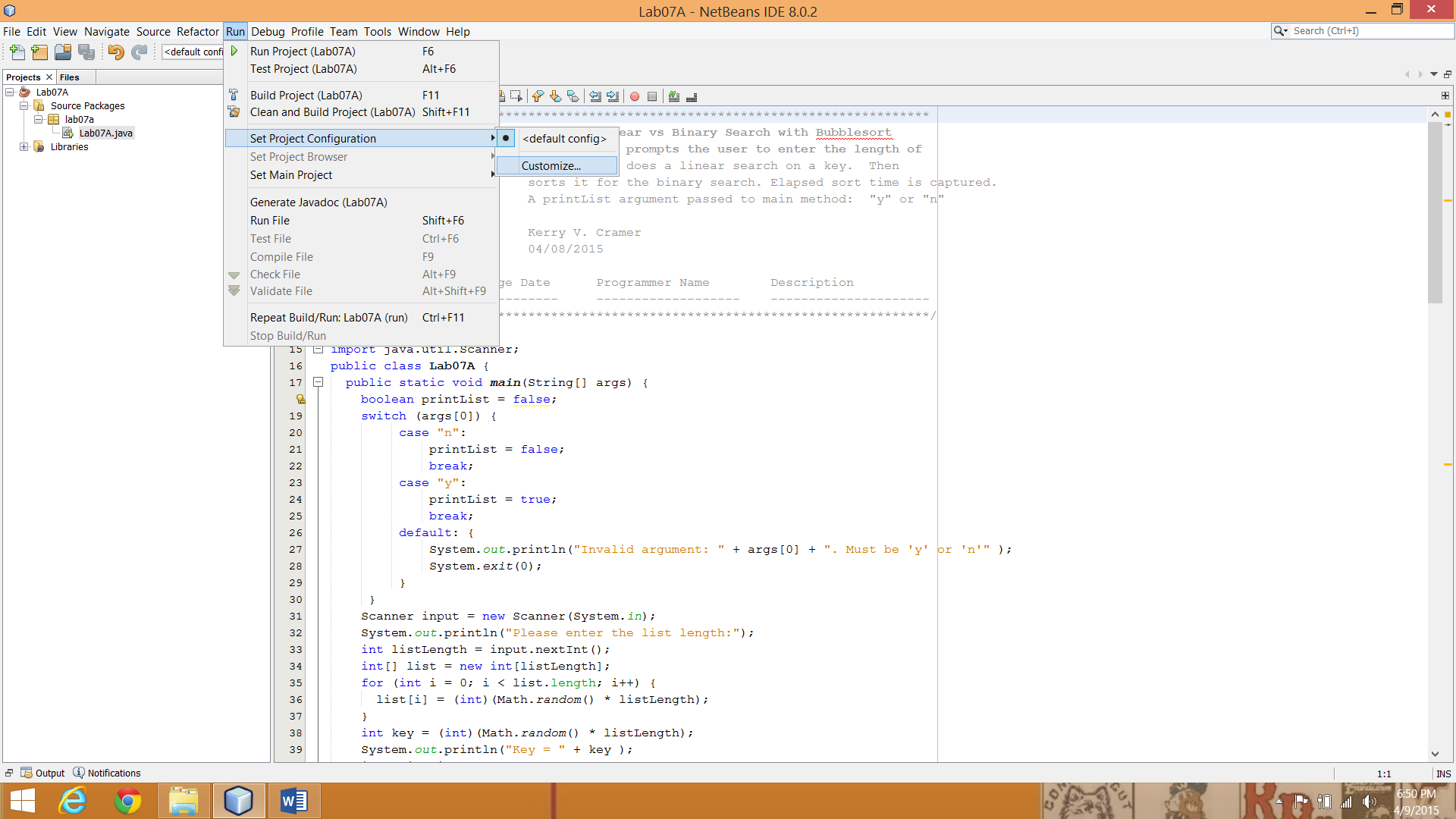
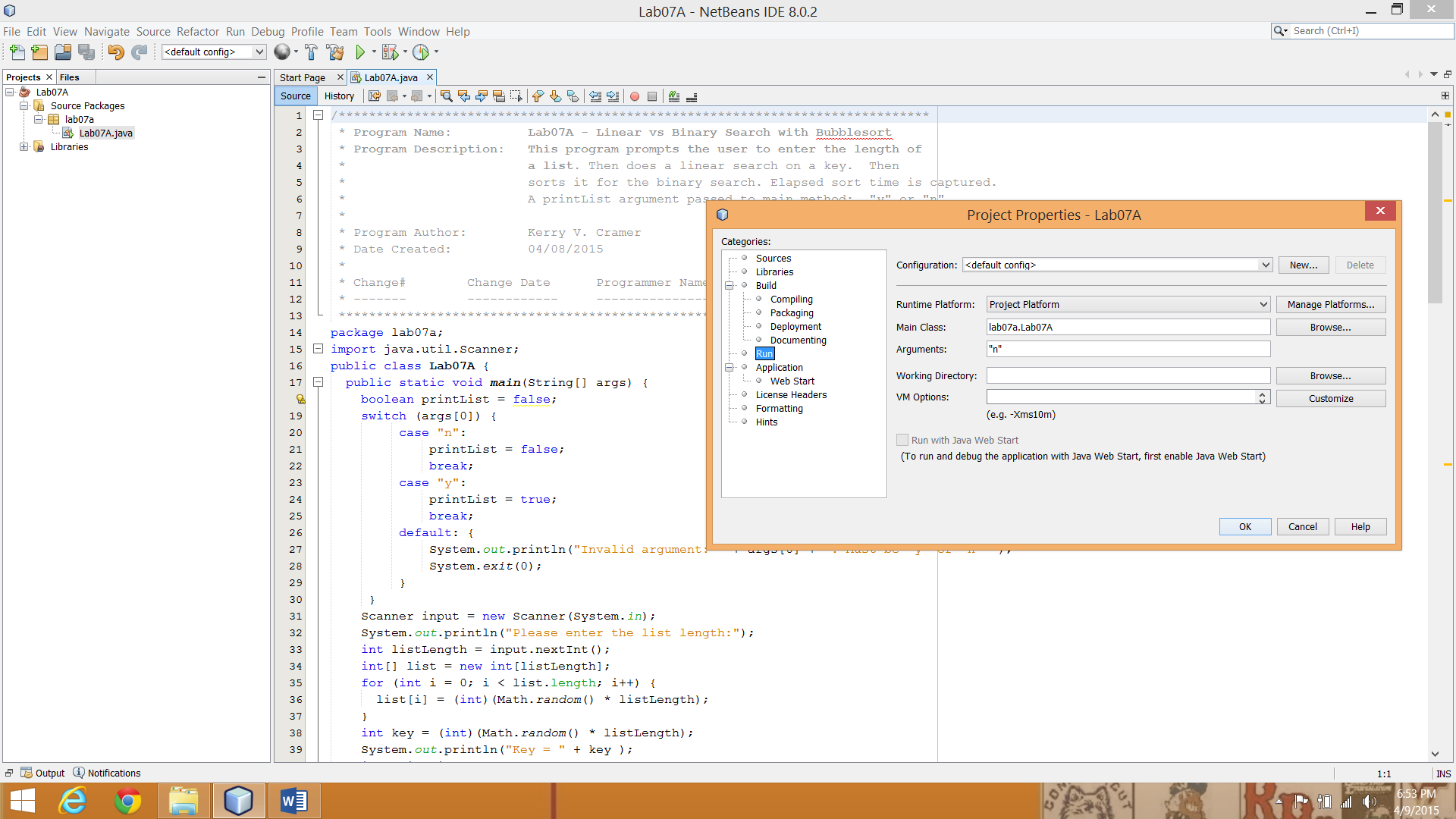
CSC108 – Introduction to Programming

Lab07A

**Linear vs Binary Search with Bubblesort:** This program passes an argument (“y” or “n”) to the main method to determine if it will print out the random list of integers or not (“y” to print the list, and “n” to not. Anything else will end the program with an “Invalid argument” message). The program initially asks for the length of the list to be created. An array of randomly generated integers is then created as well as a random integer key. Both are determined by the list length. For example, a list length of 7 will generate 7 random integers and a random key all between 0 and 6. The program does a linear search on the key and reports its position (first occurrence). Then, the program sorts the list in ascending order for the binary search. The elapsed sort time is to be captured in milliseconds. The binary search captures the number of reads to find the key. All methods should be completely written by the student **without** the use of utilities (e.g. Arrays.sort, Arrays.binarysearch, Arrays.toString). See the sample output, method headers and flowcharts, and argument passed in NetBeans below. Zip the Lab07A folder and send it to me as an attachment in Blackboard.

Methods headers are:

public static void printArray(int[] array)

public static int linearSearch(int[] list, int key)

public static void bubbleSort(int[] arrayBS)

public static int binarySearch(int[] list, int key)

Sample Java output: (printList = “n”)

Please enter the list length:

10000

Key = 6002

Key 6002 found in linear search at position: 670

Sort time is 143 milliseconds

Key 6002 found in binary search after 11 reads.

Please enter the list length:

100000

Key = 78187

Key 78187 found in linear search at position: 86427

Sort time is 18637 milliseconds

Key 78187 found in binary search after 16 reads.

Sample Java output: (printList = “y”)

Please enter the list length:

7

Key = 3

Unsorted list: 5 4 2 3 3 1 4

Key 3 found in linear search at position: 4

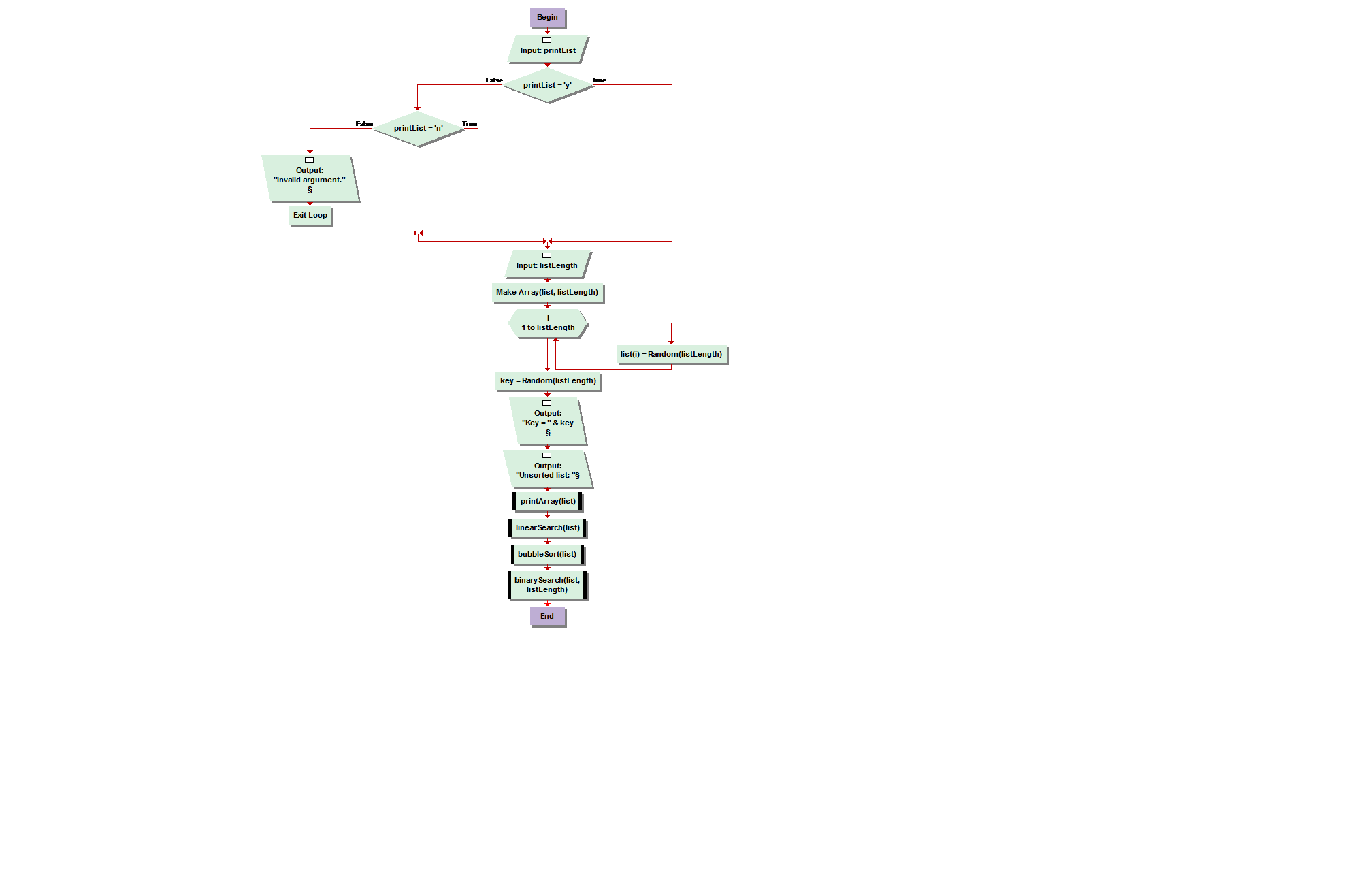
Sort time is 0 milliseconds

Sorted list: 1 2 3 3 4 4 5

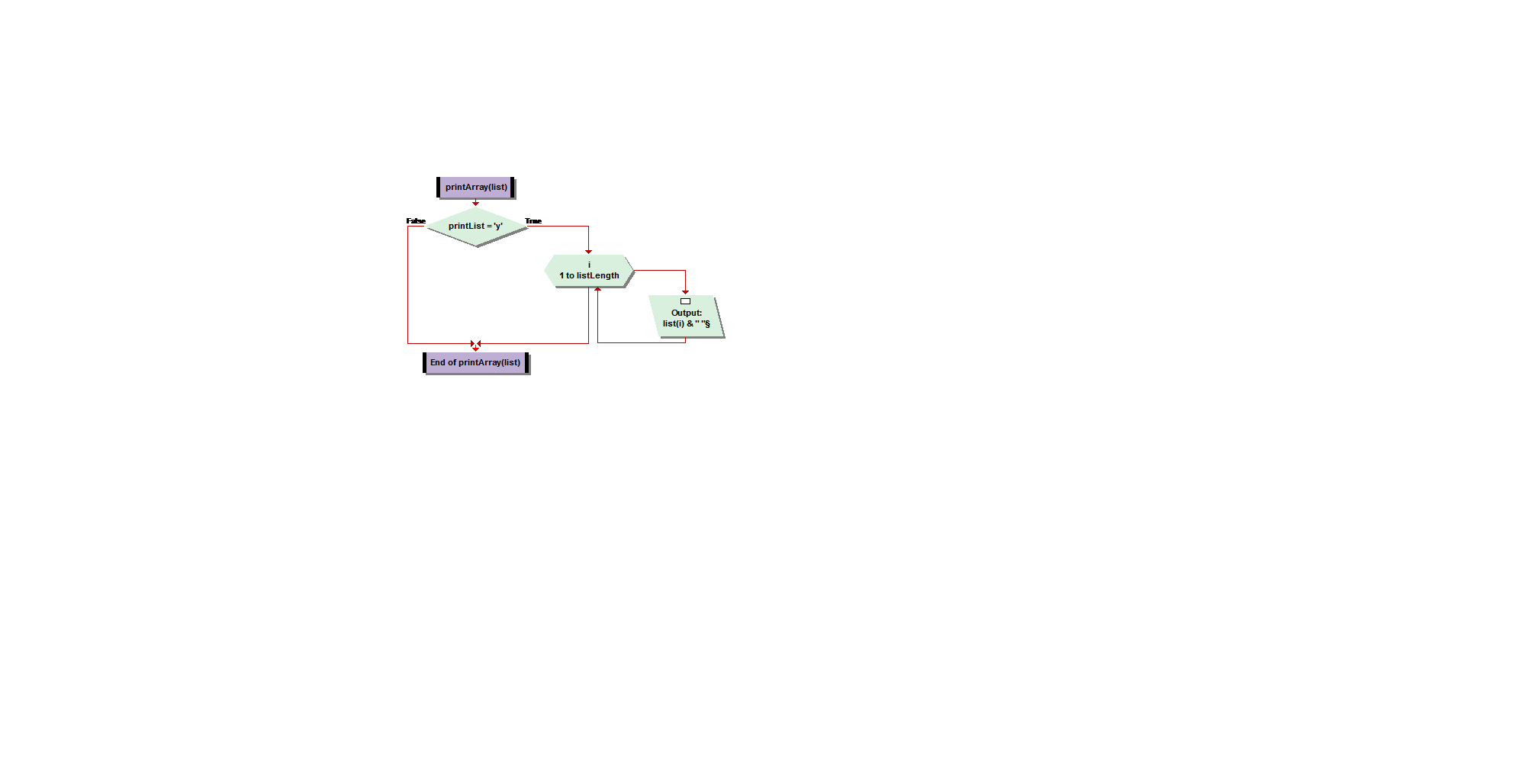
Key 3 found in binary search after 1 reads.

**Flowchart Methods:**

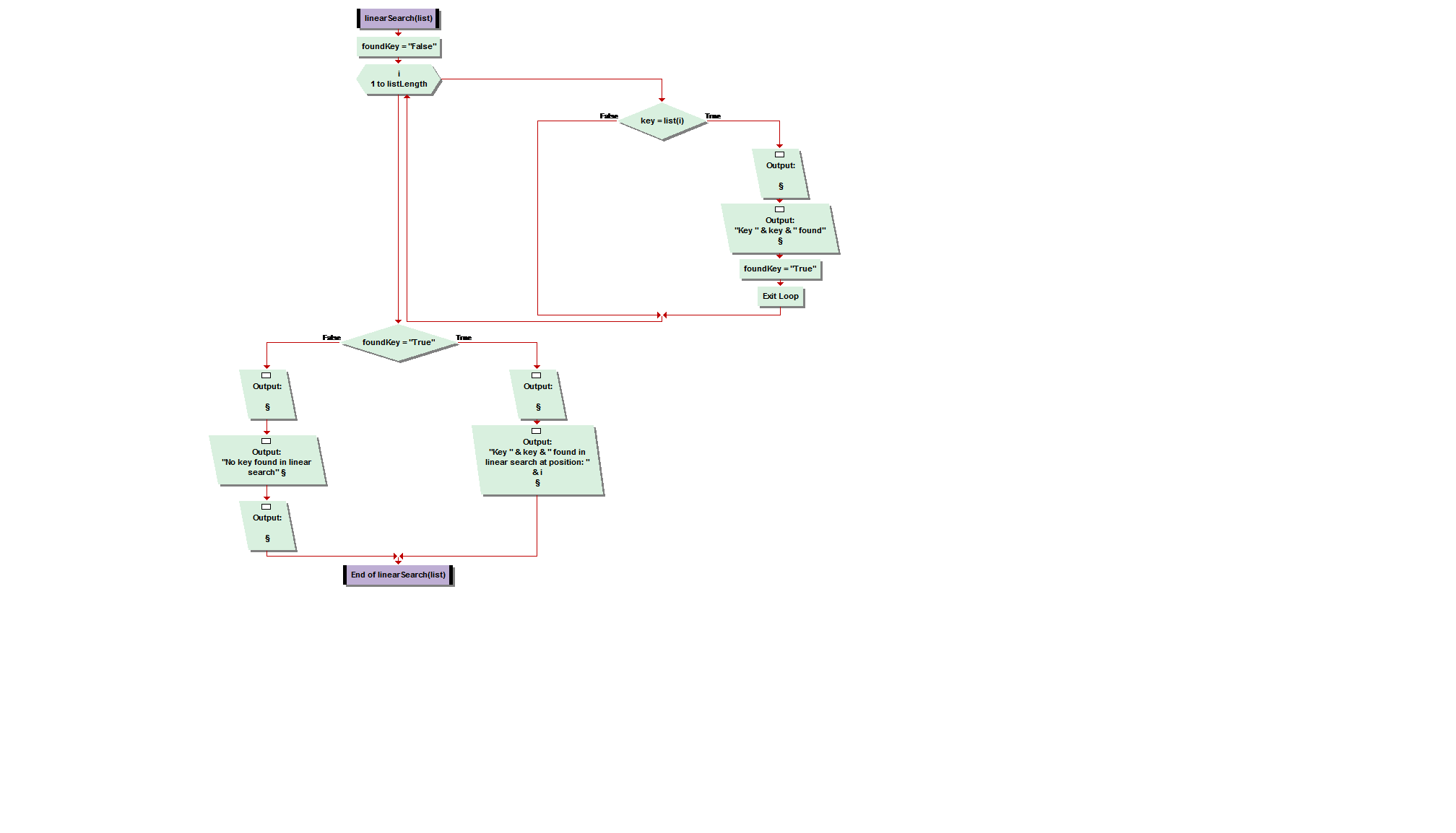
**Main:** Visual Logic does not handle argument passing to a main method. Your program must process the argument to the main method, not Scanner input.



**printArray**



**linearSearch**



package list;

import java.util.Random;

import java.util.Scanner;

public class Sortlist {

public static void main(String[] args) {

int length;

int[] list = new int[50];

Scanner in = new Scanner(System.in);

Random rn = new Random();

System.out.println("\n Enter list length:");

length = in.nextInt();

for(int i=0;i<length;i++){

int answer = rn.nextInt(length) + 1;

list[i] = answer;

}

System.out.println("\n unsorted list:");

for(int i=0; i<length; i++){

System.out.print("\t" +list[i]);

}

int key = rn.nextInt(length) + 1;

System.out.println("\n key:" + key);

if(linearSearch(key,list,length)){

bubbleSort(list,length,key);

}

}

public static boolean linearSearch(int key, int[] list, int length){

int index = 0;

while(index < length) {

if(list[index] == key) {

System.out.println("\n position of key:" + index);

return true;

}

index++;

}

return false;

}

public static void bubbleSort( int[ ] list, int size, int key)

{

int j;

boolean flag = true;

int temp, length = size;

while ( flag )

{

flag= false;

for( j=0; j < length -1; j++ )

{

if ( list[ j ] > list[j+1] )

{

temp = list[ j ];

list[ j ] = list[ j+1 ];

list[ j+1 ] = temp;

flag = true;

}

}

}

System.out.println("\n sorted list:");

for(int i=0; i<length; i++){

System.out.print("\t" +list[i]);

}

int index = binarySearch(key,list,length);

System.out.println("\n In binary search key was found at position:" +index);

}

public static int binarySearch(int key, int[] list, int length)

{

int low = 0;

int high = length - 1;

int middle = 0;

while(low <= high) {

middle = low + (high - low) / 2;

if(list[middle] < key) {

low = middle + 1;

}

else if(list[middle] > key) {

high = middle - 1;

}

else

return middle;

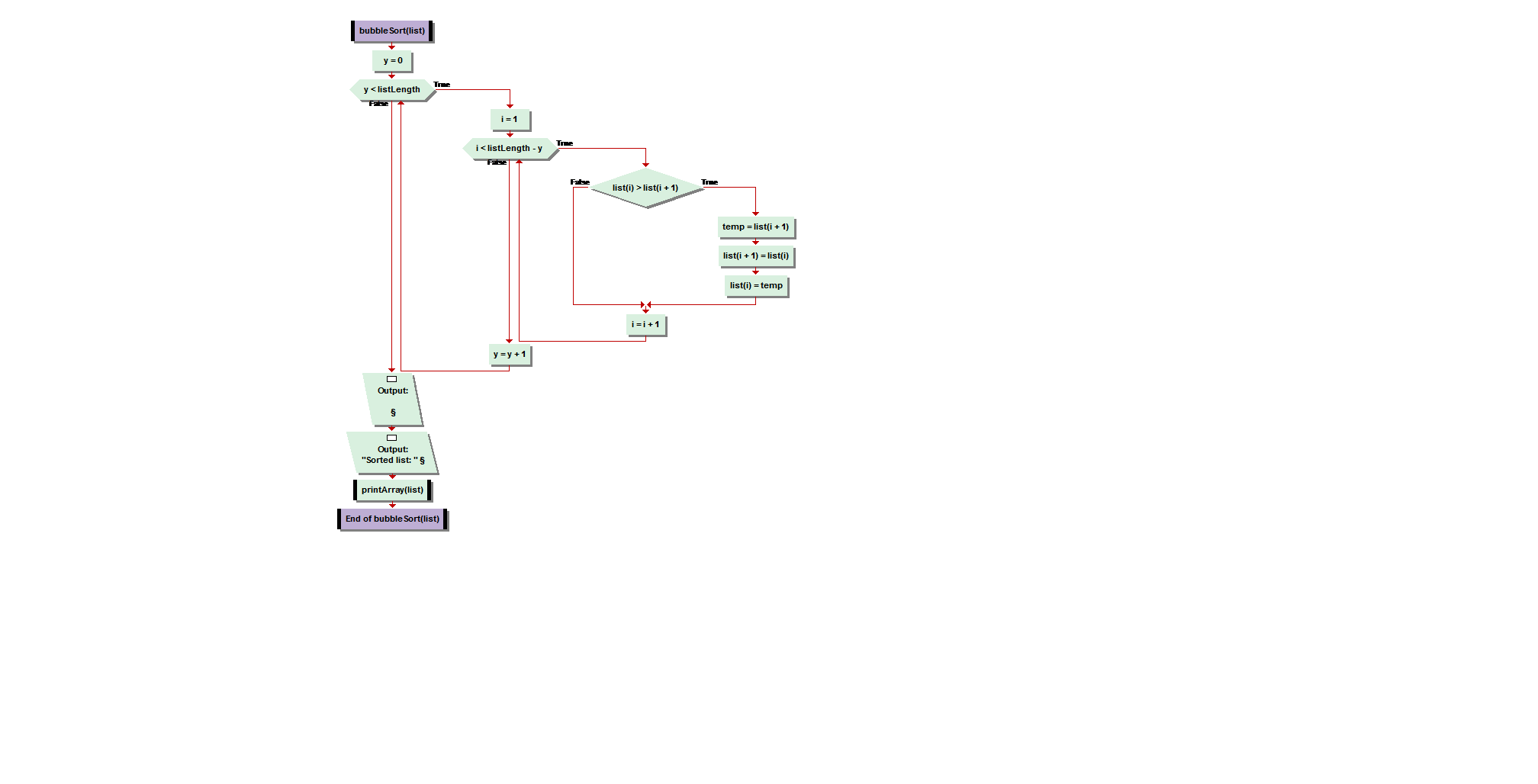
}

return middle;

}

}

**bubbleSort**



**binarySearch**

