Use sorting and searching algorithms to solve given problem scenario

Write an application to store ‘N’ numbers of type integers and sort it using below explained logic (Bubble sort). Your program should display the sorted numbers in a formatted way

Example: Let us take the array of numbers "5 1 4 2 8", and sort the array from lowest number to greatest number using bubble sort. In each step, elements written in bold are being compared. Three passes will be required.

First Pass:
(5 1 4 2 8 →) (1 5 4 2 8), Here, algorithm compares the first two elements, and → swaps since 5 > 1. (1 5 4 2 8) (1 4 5 2 8), Swap since 5 > 4
(1 4 5 2 8 →) (1 4 2 5 8), Swap since 5 > 2
(1 4 2 5 8 →) (1 4 2 5 8), Now, since these elements are already in order (8 > 5), algorithm does not swap them.

Second Pass:
(1 4 2 5 8 →) (1 4 2 5 8)
(1 4 2 5 8 →) (1 2 4 5 8), Swap since 4 > 2
(1 2 4 5 8 →) (1 2 4 5 8)
(1 2 4 5 8 →) (1 2 4 5 8)
Now, the array is already sorted, but our algorithm does not know if it is completed. The algorithm needs one whole pass without any swap to know it is sorted.
Third Pass:
(1 2 4 5 8 →) (1 2 4 5 8)
(1 2 4 5 8 →) (1 2 4 5 8)
(1 2 4 5 8 →) (1 2 4 5 8)
(1 2 4 5 8 →) (1 2 4 5 8)

```
package dsa;

import java.util.Arrays;
import java.util.Scanner;

public class BubbleSort {

	public static void bubbleSort(int[] arr) {

		for(int i = 0; i < arr.length-1; i++) {
			for(int j = 0; j < arr.length-1-i; j++) {
				if(arr[j] > arr[j+1]) {
					int temp = arr[j];
					arr[j] = arr[j+1];
					arr[j+1] = temp;
				}
			}
		}
	}

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		Scanner input = new Scanner(System.in);
		System.out.println("Enter number of elements : ");
		int n = input.nextInt();

		int[] arr = new int[n];
		for(int i = 0; i < n; i++) {
			arr[i] = input.nextInt();
		}

		bubbleSort(arr);

		System.out.println(Arrays.toString(arr));
	}

}
```

Use sorting and searching algorithms to solve given problem scenario

Write a method which accepts an integer array and key element to search. It should return ‘true’ if given key element found otherwise ‘false’

```
package dsa;

import java.util.Scanner;

public class LinearSearch {

	public static boolean linearSearch(int[] arr, int key) {
		for(int i : arr) {
			if(i == key) {
				return true;
			}
		}

		return false;
	}

	public static void main(String[] args) {
		Scanner input = new Scanner(System.in);
		System.out.println("Enter number of elements : ");
		int n = input.nextInt();

		int[] arr = new int[n];
		System.out.println("Enter elements : ");
		for(int i = 0; i < n; i++) {
			arr[i] = input.nextInt();
		}
		System.out.println("Enter element to find : ");
		int key = input.nextInt();

		System.out.println(linearSearch(arr, key));
		input.close();
	}

}
```

Use sorting and searching algorithms to solve given problem scenario

Write a menu driven program to implement binary search algorithm on both integer elements and strings.

Menu:

1. Binary search for Integer elements
2. Binary search for Strings
3. Exit

Write two methods:

a) **Boolean findElement(int arr[], int key):** Should return ‘true’ if key element found otherwise

‘false’

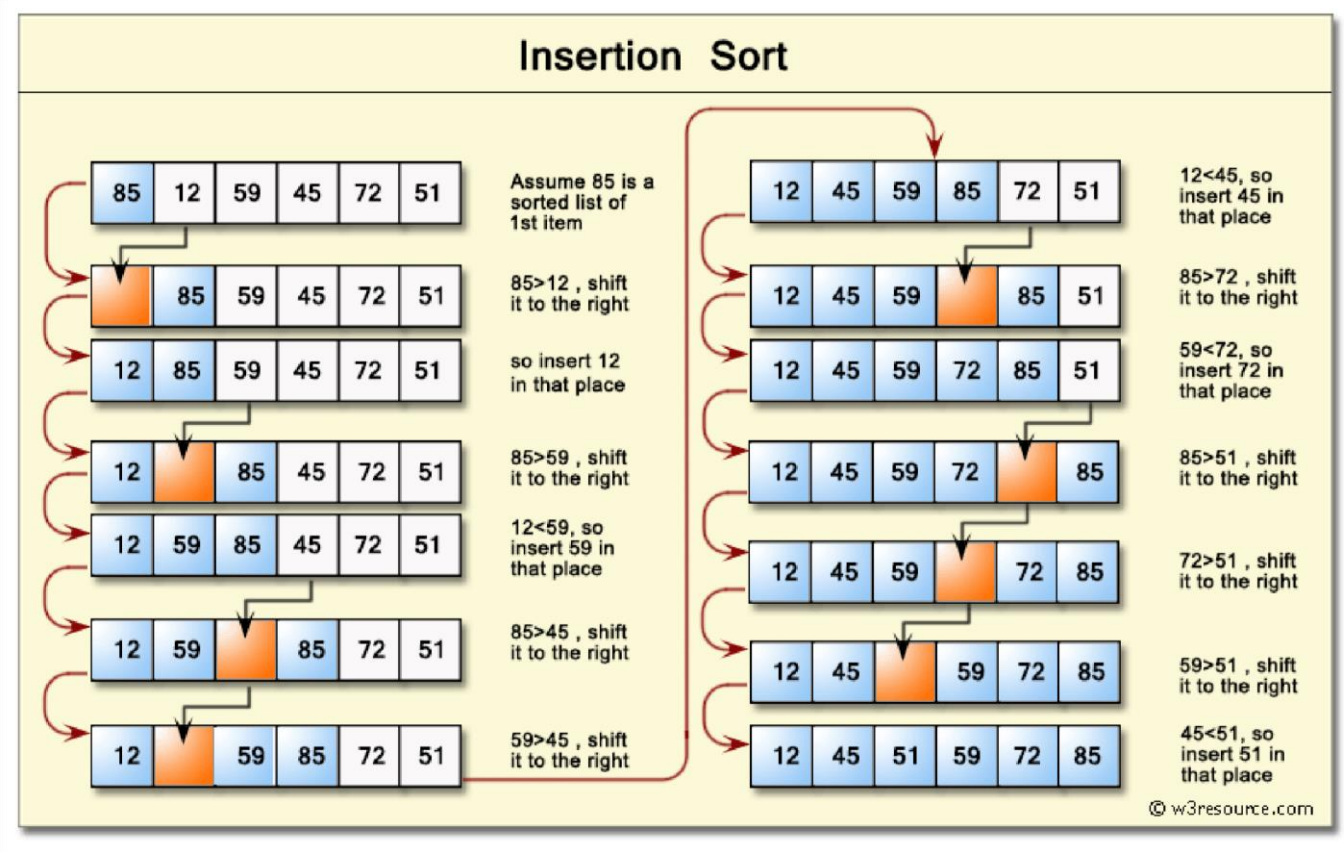
b) **Boolean findString(String names[], String name):** Should return ‘tru’ if name found in the list otherwise ‘false’.


```
package dsa;

import java.util.Scanner;

public class BinarySearch {
	public static boolean findElement(int[] arr, int key) {
		int start = 0;
		int end = arr.length - 1;

		while (start <= end) {
			int mid = start + (end - start) / 2;

			if (arr[mid] == key) {
				return true;
			} else if (key < arr[mid]) {
				end = mid - 1;
			} else {
				start = mid + 1;
			}
		}
		return false;
	}

	public static boolean findString(String names[], String name) {
		int start = 0;
		int end = names.length-1;

		while(start <= end) {
			int mid = start+(end - start) / 2;

			if(names[mid].compareTo(name) == 0) {
				return true;
			}
			else if(names[mid].compareTo(name) > 0) {
				end = mid-1;
			}
			else {
				start = mid+1;
			}
		}
		return false;
	}

	public static void main(String[] args) {
		int choice = 0;
		Scanner input = new Scanner(System.in);

		System.out.println("*********Binary Search*********\n");
		System.out.println("\n-----------------------------------------------\n");

		while (choice != 4) {
			System.out.println("\nChose one from the below options...\n");
			System.out.println("\n1.Binary search for Integer elements \n2.Binary search for Strings \n3.Exit");
			System.out.println("\n Enter your choice \n");
			choice = input.nextInt();
			switch (choice) {
			case 1: {
				System.out.println("Enter number of elements : ");
				int n = input.nextInt();

				int[] arr = new int[n];
				System.out.println("Enter elements : ");
				for (int i = 0; i < n; i++) {
					arr[i] = input.nextInt();
				}
				System.out.println("Enter element to find : ");
				int key = input.nextInt();
				System.out.println(findElement(arr, key));
				break;
			}
			case 2: {
				System.out.println("Enter number of names : ");
				int n = input.nextInt();

				String[] arr = new String[n];
				System.out.println("Enter names : ");
				for (int i = 0; i < n; i++) {
					arr[i] = input.next();
				}
				System.out.println("Enter name to find : ");
				String key = input.next();
				System.out.println(findString(arr, key));
				break;
			}
			case 3: {
				System.out.println("Exiting....");
				System.exit(0);
				break;
			}
			default: {
				System.out.println("Please Enter valid choice ");
			}
			}

		}

	}
}
```

Use sorting and searching algorithms to solve given problem scenario

Write a method which accepts array of unsorted integer elements and display elements in sorted order. Use insertion sort algorithm to sort.

Refer below diagram to understand how insertion sort works.



Source: https://www.w3resource.com/java-exercises/sorting/java-sorting-algorithm-exercise-7.php

```
package dsa;

import java.util.Arrays;
import java.util.Scanner;

public class InsertionSort {

	public static void insertionSort(int[] arr) {

		for(int i = 1; i < arr.length; i++) {
			int current = arr[i];
			int prevIndex = i-1;

			while(prevIndex >= 0 && arr[prevIndex] > current) {
				arr[prevIndex+1] = arr[prevIndex];
				prevIndex--;
			}
			arr[prevIndex+1] = current;
		}
	}

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		Scanner input = new Scanner(System.in);
		System.out.println("Enter number of elements : ");
		int n = input.nextInt();

		int[] arr = new int[n];
		System.out.println("Enter elements : ");
		for(int i = 0; i < n; i++) {
			arr[i] = input.nextInt();
		}

		insertionSort(arr);

		System.out.println(Arrays.toString(arr));
	}

}
```