LAB # 02

ArrayList and Vector in JAVA

Objective: To implement ArrayList and Vector.

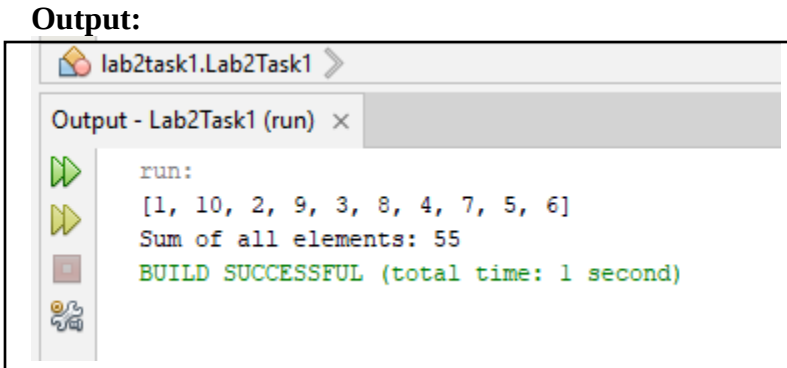
Lab Tasks:

1. Write a program that initializes Vector with 10 integers in it. Display all the integers and sum of these integers.

Source Code:

```
package lab2task1;
import java.util.*;
public class Lab2Task1 {
public static void main(String[] args) {
        int sum = 0;
        Vector <Integer> vector = new Vector < Integer > ();
        vector.add(1);
        vector.add(10);
        vector.add(2);
        vector.add(9);
        vector.add(3);
        vector.add(8);
        vector.add(4);
        vector.add(7);
        vector.add(5);
        vector.add(6);
        System.out.println(vector);
          for (Integer x: vector) {
            sum += x;
          }
          System.out.println("Sum of all elements: " + sum);}}
```

Output:



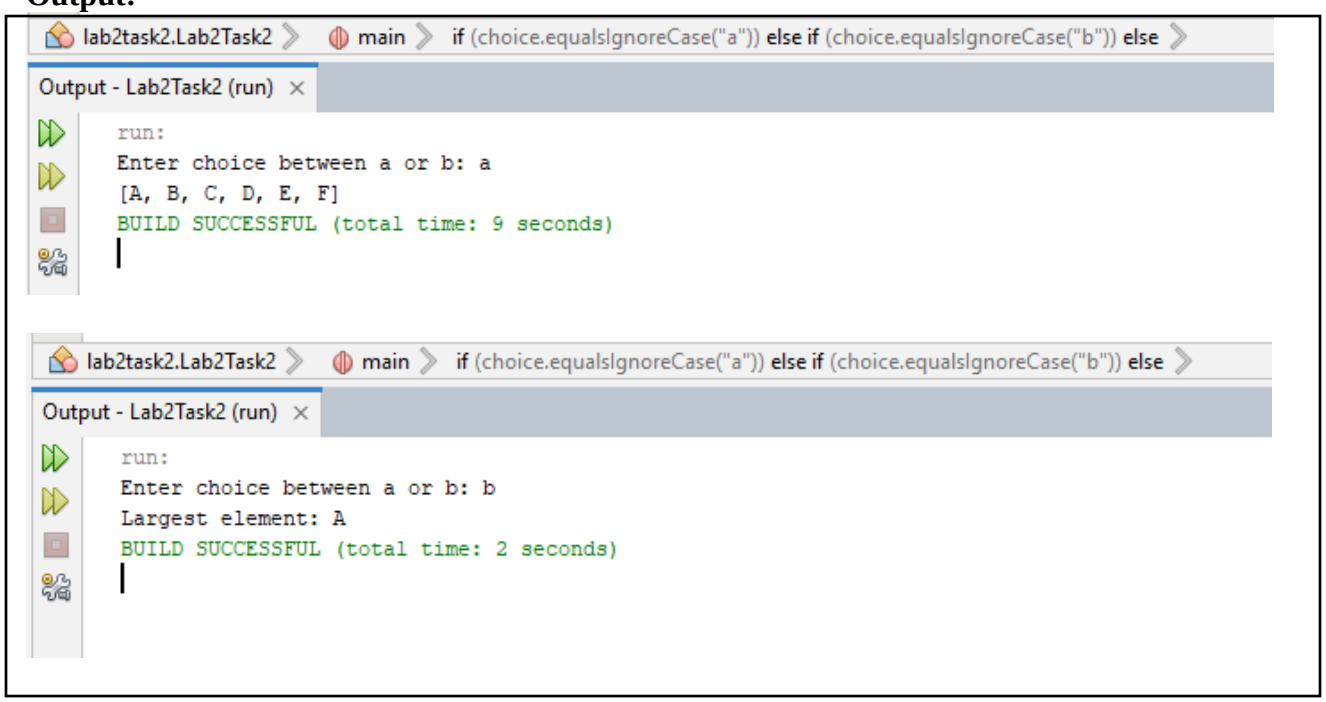
2. Create a ArrayList of string. Write a menu driven program which:

a. Displays all the elements
b. Displays the largest String

Source Code:

```
package lab2task2;
import java.util.*;
public class Lab2Task2 {
   public static void main(String[] args) {
      List<String> alist = Arrays.asList("A", "B", "C", "D", "E", "F");
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter choice between a or b: ");
      String choice = sc.next();
      if (choice.equalsIgnoreCase("a")) {
         System.out.println(alist);
      } else if (choice.equalsIgnoreCase("b")) {
         String largest = Collections.max(alist, Comparator.comparingInt(String::length));
         System.out.println("Largest element: " + largest);
      } else {
         System.out.println("Choose Between A or B!");
      }
      sc.close();}}
```

Output:



3. Create a Arraylist storing Employee details including Emp_id, Emp_Name, Emp_gender, Year_of_Joining (you can also add more attributes including these). Then sort the employees according to their joining year using Comparator and Comparable interfaces.

Source Code:

```
package lab2task3;
import java.util.*;
class Employee {
   int id, yearOfJoining;
   String name, gender;
   public Employee() {
      this(1, 1947, "John Cena", "Male");
   }
   public Employee(int id, int yearOfJoining, String name, String gender) {
      this.id = id;
      this.yearOfJoining = yearOfJoining;
      this.name = name;
      this.gender = gender;}}
public class Lab2Task3 {
   public static void main(String[] args) {
      List<Employee> emp = new ArrayList<>(Arrays.asList(
         new Employee(2, 1069, "Big Show", "Male"),
         new Employee(3, 2069, "Alexa Bliss", "Female"),
         new Employee(4, 3069, "Gigi Dolin", "Female")
      ));
      emp.sort(Comparator.comparingInt(e -> e.yearOfJoining));
      for (Employee e : emp) {
         System.out.println("Name: " + e.name + "\nGender: " + e.gender +
                        "\nId: " + e.id + "\nYear Of Joining: " + e.yearOfJoining + "\n");}}}
```

Output:

```
Output - Lab2Task3 (run)
run:
Name: Big Show
Gender: Male
Id: 2
Year Of Joining: 1069

Name: Alexa Bliss
Gender: Female
Id: 3
Year Of Joining: 2069

Name: Gigi Dolin
Gender: Female
Id: 4
Year Of Joining: 3069

BUILD SUCCESSFUL (total time: 0 seconds)
```

4. Write a program that initializes Vector with 10 integers in it.

- Display all the integers
- Sum of these integers.
- Find Maximum Element in Vector

Source Code:

```
package lab2task4;
import java.util.*;
public class Lab2Task4 {
   public static void main(String[] args) {
      Vector<Integer> vector = new Vector<>();
      for (int i = 10; i <= 100; i += 10) vector.add(i);
      System.out.println("Vector elements: " + vector);
      System.out.println("Sum of elements: " + vector.stream().mapToInt(Integer::intValue).sum());
      System.out.println("Maximum element: " + Collections.max(vector));}}
```

Output:

```
lab2task4.Lab2Task4
Output - Lab2Task4 (run)
run:
Vector elements: [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
Sum of elements: 550
Maximum element: 100
BUILD SUCCESSFUL (total time: 0 seconds)
```

5. Find the k-th smallest element in a sorted ArrayList

Source Code:

```
package lab2task5;
import java.util.*;
public class Lab2Task5 {
   public static void main(String[] args) {
      List<Integer> sortedList = List.of(1, 3, 5, 7, 9);
      int k = 3;
      System.out.println(k > 0 && k <= sortedList.size()
         ? "The " + k + "-th smallest element is: " + sortedList.get(k - 1)
         : "k is out of bounds");}}
```

Output:

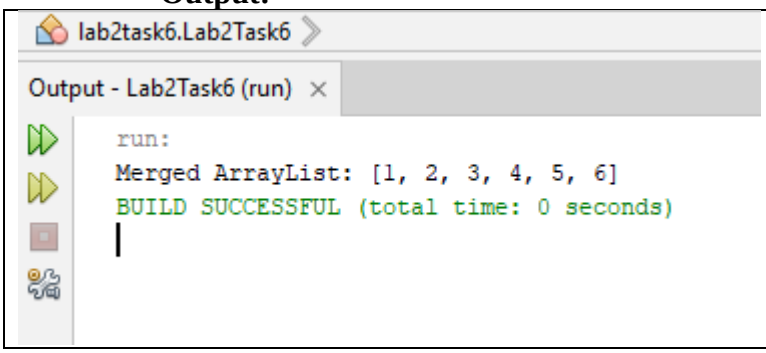
```
lab2task5.Lab2Task5 > main
Output - Lab2Task5 (run)
run:
The 3-th smallest element is: 5
BUILD SUCCESSFUL (total time: 0 seconds)
```

6. Write a program to merge two ArrayLists into one.

Source Code:

```
package lab2task6;
import java.util.ArrayList;
import java.util.List;
public class Lab2Task6 {
   public static void main(String[] args) {
      ArrayList<Integer> mergedList = new ArrayList<>(List.of(1, 2, 3, 4, 5, 6));
      System.out.println("Merged ArrayList: " + mergedList);}}
```

Output:



Home Tasks:

1. Create a Vector storing integer objects as an input.
   a. Sort the vector
   b. Display largest number
   c. Display smallest number

Source Code:

```
import java.util.Vector;
import java.util.Collections;
import java.util.Scanner;
public class Lab2HomeTask1 {
   public static void main(String[] args) {
      Scanner input = new Scanner(System.in);
      Vector<Integer> integers = new Vector<>();
      for (int i = 0; i < 5; i++) {
         System.out.print("Enter an integer: ");
         integers.add(input.nextInt());
      }
      Collections.sort(integers);
      System.out.println("Elements: " + integers);
      System.out.println("Maximum number: " + integers.lastElement());
      System.out.println("Minimum number: " + integers.firstElement());
      input.close();
   }
}System.out.println("Maximum number: "+max);
System.out.println("Minimum number: "+min);}}
```

Output:

```
Lab2HomeTask1 > main >
Output - Lab2HomeTask1 (run)
run:
Enter an integer: 12
Enter an integer: 24
Enter an integer: 36
Enter an integer: 48
Enter an integer: 60
Elements: [12, 24, 36, 48, 60]
Maximum number: 60
Minimum number: 12
BUILD SUCCESSFUL (total time: 22 seconds)
```

2. Write a java program which takes user input and gives hashcode value of those inputs using hashCode () method.

Source Code:

```
package lab2hometask2;
import java.util.Scanner;
public class Lab2HomeTask2 {
    public static void main(String[] args) {
    String s;
    Scanner input = new Scanner(System.in);
    System.out.print("Enter Anything: ");
    s = input.nextLine();
    System.out.println("hashCode: "+s.hashCode());}}
```

Output:

```
lab2hometask2.Lab2HomeTask2 >
Output - Lab2HomeTask2 (run)
run:
Enter Anything: Khurram
hashCode: 932291844
BUILD SUCCESSFUL (total time: 3 seconds)
```

3. Scenario based

Create a java project, suppose you work for a company that needs to manage a list of employees. Each employee has a unique combination of a name and an ID. Your goal is to ensure that you can track employees effectively and avoid duplicate entries in your system.

Requirements

a. Employee Class: You need to create an Employee class that includes:

- name: The employee's name (String).

- id: The employee's unique identifier (int).
- Override the hashCode() and equals() methods to ensure that two employees are considered equal if they have the same name and id.

b. Employee Management: You will use a HashSet to store employee records. This will help you avoid duplicate entries.

c. Operations: Implement operations to:

- Add new employees to the record.
- Check if an employee already exists in the records.
- Display all employees.

Source Code:

```
package lab2hometask3;
import java.util.HashSet;
import java.util.Scanner;
class Employee {
   String name; int id;
   Employee(String name, int id) { this.name = name; this.id = id; }
   @Override public boolean equals(Object obj) {
      return this == obj || (obj instanceof Employee e && id == e.id && name.equals(e.name));}
   @Override public int hashCode() { return 31 * name.hashCode() + id; }
   @Override public String toString() { return name + " (ID: " + id + ")";}}
public class Lab2HomeTask3 {
   public static void main(String[] args) {
      HashSet<Employee> employees = new HashSet<>();
      Scanner scanner = new Scanner(System.in);
      while (true) {
         System.out.println("1. Add Employee\n2. Check Employee Exists\n3. Display Employees\n4. Exit");
         switch (scanner.nextInt()) {
            case 1:
               employees.add(new Employee(scanner.next(), scanner.nextInt()));
               System.out.println("Employee added.");
               break;
            case 2:
               System.out.println(employees.contains(new Employee(scanner.next(), scanner.nextInt())) ?
"Employee exists." : "Employee does not exist.");
               break;
            case 3:
               System.out.println("Employees: " + employees);
               break;
            case 4:
               scanner.close();
               return;
            default:
               System.out.println("Invalid choice.");}}}}
```

Output:

```
Output - Lab2HomeTask3 (run)
run:
1. Add Employee
2. Check Employee Exists
3. Display Employees
4. Exit
1
Khurram
006
Employee added.
1. Add Employee
2. Check Employee Exists
3. Display Employees
4. Exit
3
Employees: [Khurram (ID: 6)]
1. Add Employee
2. Check Employee Exists
3. Display Employees
4. Exit
4
BUILD SUCCESSFUL (total time: 25 seconds)
```

4. Create a Color class that has red, green, and blue values. Two colors are considered equal if their RGB values are the same

Source Code:

```
package lab2hometask4;
class Color {
   private int red, green, blue;
   public Color(int red, int green, int blue) {
      this.red = red;
      this.green = green;
      this.blue = blue;}
   @Override
   public String toString() {
      return "Color{red=" + red + ", green=" + green + ", blue=" + blue + "}";}}
public class Lab2HomeTask4 {
   public static void main(String[] args) {
      Color[] colors = {
         new Color(255, 0, 0), // Red
         new Color(0, 255, 0), // Green
         new Color(0, 0, 255)  // Blue
      };
      for (Color color : colors) {
         System.out.println(color);}}}
```

Output:

```
Output - Lab2HomeTask4 (run)
run:
Color{red=255, green=0, blue=0}
Color{red=0, green=255, blue=0}
Color{red=0, green=0, blue=255}
BUILD SUCCESSFUL (total time: 0 seconds)
```