LAB # 02

ArrayList and Vector in JAVA

OBJECTIVE:
To implement ArrayList and Vector

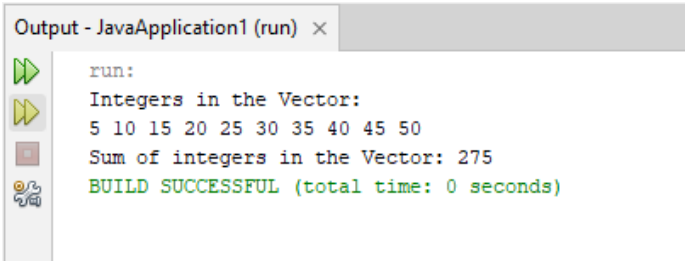
LAB TASKS:

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

CODE:

```
public static void main(String[] args) {
    Vector<Integer> numbers = new Vector<>();
    numbers.add(5); numbers.add(10); numbers.add(15); numbers.add(20); numbers.add(25);
    numbers.add(30); numbers.add(35); numbers.add(40); numbers.add(45); numbers.add(50);
    System.out.println("Integers in the Vector:");
    for (int number : numbers) {
      System.out.print(number + " ");
    }
    int sum = 0;
    for (int number : numbers) {
      sum += number;  }
    System.out.println("\nSum of integers in the Vector: " + sum); } }
```

OUTPUT



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

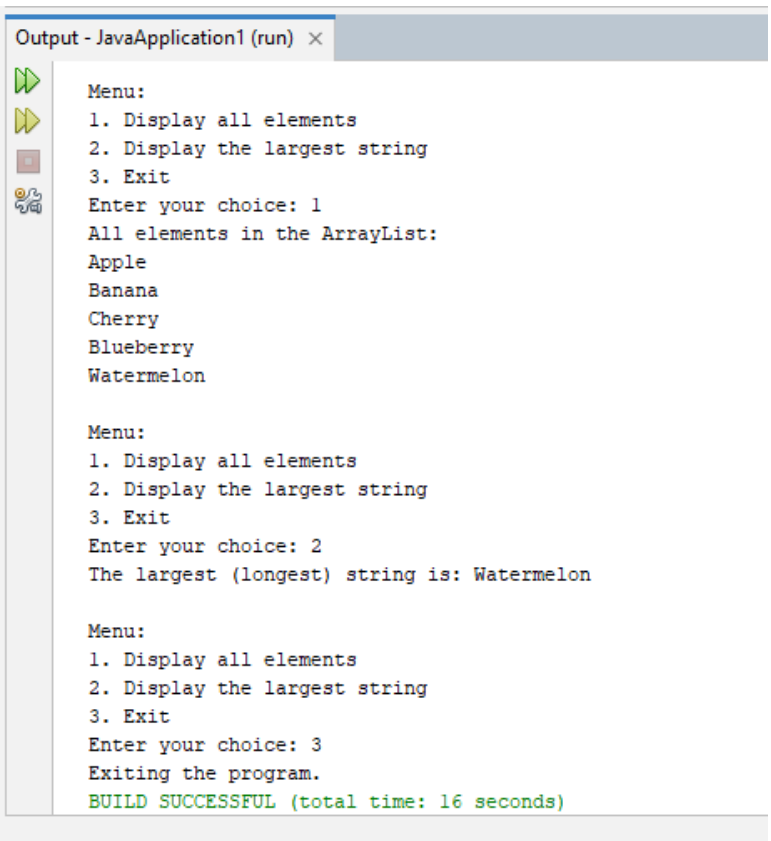
a. Displays all the elements b. Displays the largest String.

CODE:

```
public class JavaApplication1 {
   public static void main(String[] args) {
       ArrayList<String> strings = new ArrayList<>();
      strings.add("Apple");  strings.add("Banana");  strings.add("Cherry"); strings.add("Blueberry");
      strings.add("Watermelon");
      Scanner scanner = new Scanner(System.in);
      int choice;
      do {
         System.out.println("\nMenu:");
         System.out.println("1. Display all elements");
         System.out.println("2. Display the largest string");
         System.out.println("3. Exit");
         System.out.print("Enter your choice: ");
         choice = scanner.nextInt();
         scanner.nextLine(); // Consume newline character
         switch (choice) {
            case 1:
               System.out.println("All elements in the ArrayList:");
               for (String str : strings) {
                  System.out.println(str);  }
               break;
            case 2:
               if (strings.isEmpty()) {
                  System.out.println("The ArrayList is empty.");
               } else {
                  String largestString = strings.get(0);
                  for (String str : strings) {
```

```
                if (str.length() > largestString.length()) {
                    largestString = str;}}
            System.out.println("The largest (longest) string is: " + largestString);  }
        break;
    case 3:
        System.out.println("Exiting the program.");
        break;
    default:
        System.out.println("Invalid choice. Please try again.");   }
} while (choice != 3);   }
```

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.

CODE:

```
// Define the Employee class
class Employee {
   int empId;
   String empName;
   String empGender;
   int yearOfJoining;
   // Constructor to initialize Employee object
   public Employee(int empId, String empName, String empGender, int yearOfJoining) {
      this.empId = empId;
      this.empName = empName;
      this.empGender = empGender;
      this.yearOfJoining = yearOfJoining;}
   // Method to return a string representation of the employee
   @Override
   public String toString() {
      return "ID: " + empId + ", Name: " + empName + ", Gender: " + empGender + ", Year of Joining: " +
yearOfJoining;   }}
public class JavaApplication1 {
   public static void main(String[] args) {
      // Create an ArrayList to store Employee objects
      ArrayList<Employee> employees = new ArrayList<>();
      // Add Employee objects to the ArrayList
      employees.add(new Employee(101, "Kahsaf", "Female", 2015));
      employees.add(new Employee(105, "Rayan", "Male", 2014));
      employees.add(new Employee(102, "Fatima", "Female", 2012));
      employees.add(new Employee(103, "Laraib", "female", 2018));
      employees.add(new Employee(104, "Esha", "Female", 2010));
      // Sort employees by year of joining using a simple bubble sort
      for (int i = 0; i < employees.size() - 1; i++) {
```

```
        for (int j = 0; j < employees.size() - 1 - i; j++) {
            // Compare the year of joining
            if (employees.get(j).yearOfJoining > employees.get(j + 1).yearOfJoining) {
                // Swap employees if they are in the wrong order
                Employee temp = employees.get(j);
                employees.set(j, employees.get(j + 1));
                employees.set(j + 1, temp);}   }  }
    // Display the sorted employees
    System.out.println("Employees sorted by Year of Joining:");
    for (Employee emp : employees) {
        System.out.println(emp); // Print each employee's details
 }
  }
}
```

OUTPUT:

```
Output - JavaApplication1 (run)
run:
Employees sorted by Year of Joining:
ID: 104, Name: Esha, Gender: Female, Year of Joining: 2010
ID: 102, Name: Fatima, Gender: Female, Year of Joining: 2012
ID: 105, Name: Rayan, Gender: Male, Year of Joining: 2014
ID: 101, Name: Kahsaf, Gender: Female, Year of Joining: 2015
ID: 103, Name: Laraib, Gender: female, Year of Joining: 2018
BUILD SUCCESSFUL (total time: 0 seconds)
```

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

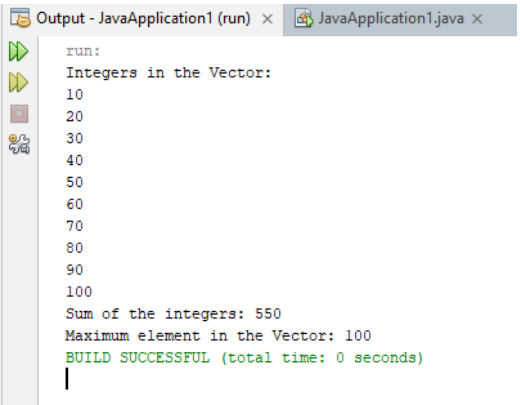
- Display all the integers ,Sum of these integers. ,Find Maximum Element in Vector

CODE:

```
public class JavaApplication1 {
```

```
 public static void main(String[] args) {
    Vector<Integer> numbers = new Vector<>();
    numbers.add(10);numbers.add(20);   numbers.add(30);  numbers.add(40);numbers.add(50);
numbers.add(60);numbers.add(70);numbers.add(80);   numbers.add(90);numbers.add(100);
    System.out.println("Integers in the Vector:");
    for (Integer number : numbers) {
       System.out.println(number);}
    int sum = 0;
    for (Integer number : numbers) {
       sum += number; }
    System.out.println("Sum of the integers: " + sum);
    int max = numbers.get(0); // Assume the first number is the maximum
    for (Integer number : numbers) {
       if (number > max) {
          max = number; // Update max if current number is greater  }}
    System.out.println("Maximum element in the Vector: " + max);
```

OUTPUT:



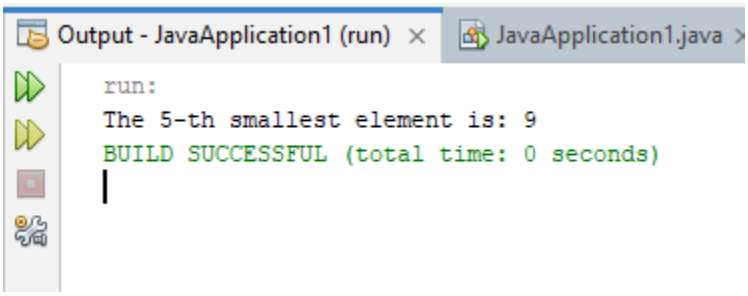
5 Find the k-th smallest element in a sorted ArrayList

CODE:

```
 // Create and initialize a sorted ArrayList of integers
```

```
        ArrayList<Integer> sortedList = new ArrayList<>();
        sortedList.add(1);sortedList.add(3);sortedList.add(5);sortedList.add(7);  sortedList.add(9);
        sortedList.add(11);  sortedList.add(13);sortedList.add(15);  sortedList.add(17);sortedList.add(19);
        int k = 5;
        if (k > 0 && k <= sortedList.size()) {
            int kthSmallest = sortedList.get(k - 1); // k-1 for 0-based index
            System.out.println("The " + k + "-th smallest element is: " + kthSmallest);
        } else {
            System.out.println("Invalid value for k. Please enter a value between 1 and " + sortedList.size());}
    }
}
```

OUTPUT:



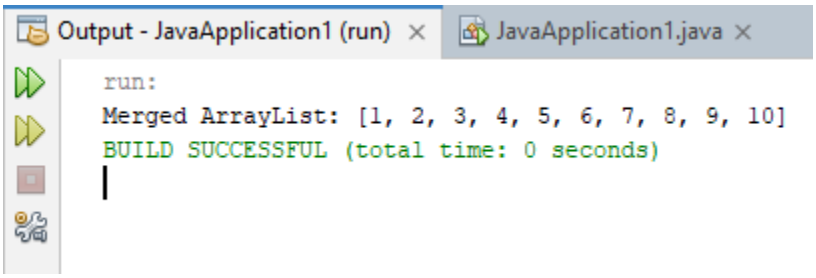
6 Write a program to merge two ArrayLists into one.

CODE:

```
public class JavaApplication1 {
    public static void main(String[] args) {
        ArrayList<Integer> list1 = new ArrayList<>();
```

```
    list1.add(1);list1.add(2);  list1.add(3);  list1.add(4);  list1.add(5);
    ArrayList<Integer> list2 = new ArrayList<>();
    list2.add(6);  list2.add(7);  list2.add(8);  list2.add(9);  list2.add(10);
    ArrayList<Integer> mergedList = new ArrayList<>();
    mergedList.addAll(list1);
    mergedList.addAll(list2);
    System.out.println("Merged ArrayList: " + mergedList);  }
}
```

OUTPUT:



HOME TASK:

Create a Vector storing integer objects as an input.

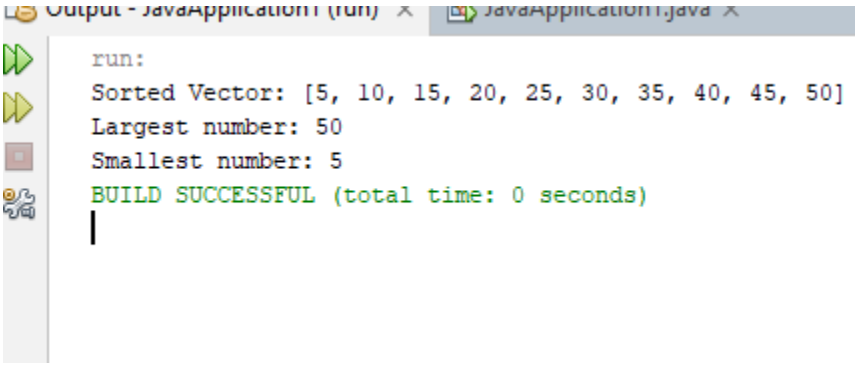
a. Sort the vector b. Display largest number c. Display smallest number

CODE:

```
public class JavaApplication1 {
   public static void main(String[] args) {
```

```
Vector<Integer> numbers = new Vector<>();
numbers.add(10);   numbers.add(5);numbers.add(20);numbers.add(15);  numbers.add(30);
numbers.add(25);numbers.add(40);numbers.add(35);numbers.add(50);numbers.add(45);
for (int i = 0; i < numbers.size() - 1; i++) {
   for (int j = 0; j < numbers.size() - 1 - i; j++) {
      if (numbers.get(j) > numbers.get(j + 1)) {
         int temp = numbers.get(j);
         numbers.set(j, numbers.get(j + 1));
         numbers.set(j + 1, temp);   }  }  }
System.out.println("Sorted Vector: " + numbers);
int largestNumber = numbers.get(numbers.size() - 1);
System.out.println("Largest number: " + largestNumber);
int smallestNumber = numbers.get(0);
System.out.println("Smallest number: " + smallestNumber);  }  }
```

OUTPUT:



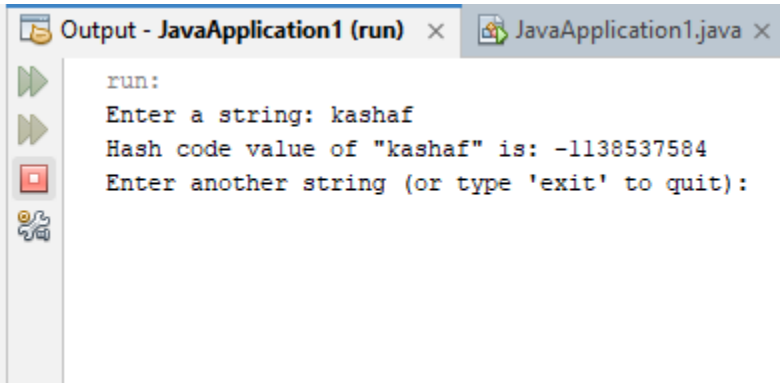
Write a java program which takes user input and gives hashcode value of those inputs using hashCode () method.

CODE:

```
public class JavaApplication1 {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      System.out.print("Enter a string: ");
      String userInput = scanner.nextLine(); // Read the user input
```

```
    int hashCodeValue = userInput.hashCode();
    System.out.println("Hash code value of \"" + userInput + "\" is: " + hashCodeValue);
    while (true) {
        System.out.print("Enter another string (or type 'exit' to quit): ");
        userInput = scanner.nextLine(); // Read the user input
        if (userInput.equalsIgnoreCase("exit")) {
            break; // Exit the loop
        }
        hashCodeValue = userInput.hashCode();
        System.out.println("Hash code value of \"" + userInput + "\" is: " + hashCodeValue);
    }
    System.out.println("Program exited.");  }}
```

OUTPUT:



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

CODE:

Employee class:

```
public class Employee {
    private String name; // Employee's name
```

```
    private int id;      // Employee's unique identifier
    public Employee(String name, int id) {
        this.name = name;
        this.id = id;}
    public String getName() {
        return name;  }
    public int getId() {
        return id;}

    // Override hashCode method
    @Override
    public int hashCode() {
        return Objects.hash(name, id);   }
    @Override
    public boolean equals(Object obj) {
        if (this == obj) return true; // Check for reference equality
        if (obj == null || getClass() != obj.getClass()) return false; // Check if the object is null or of
different class
        Employee employee = (Employee) obj; // Cast the object to Employee
        return id == employee.id && Objects.equals(name, employee.name); // Check for equality
    }
    @Override
    public String toString() {
        return "Employee{" +
                "name='" + name + '\'' +
                ", id=" + id +
                '}';}}
```

Employee management class:

```
public class EmployeeManagement {
   private HashSet<Employee> employeeSet; // HashSet to store employee records
   // Constructor
   public EmployeeManagement() {
      employeeSet = new HashSet<>(); // Initialize the HashSet
   }
   public void addEmployee(String name, int id) {
      Employee newEmployee = new Employee(name, id);
      if (employeeSet.add(newEmployee)) {
         System.out.println("Employee added successfully.");
      } else {
         System.out.println("Employee already exists in the records.");  }  }
   public void displayEmployees() {
      if (employeeSet.isEmpty()) {
         System.out.println("No employees found.");
      } else {
         System.out.println("Employee Records:");
         for (Employee employee : employeeSet) {
            System.out.println(employee);  }}}}
```

Main class:

```
public class JavaApplication1 {
   public static void main(String[] args) {
    EmployeeManagement management = new EmployeeManagement(); // Create EmployeeManagement object
      Scanner scanner = new Scanner(System.in); // Create Scanner object for user input
      int choice;
```

```
do {
    System.out.println("\nEmployee Management System");
    System.out.println("1. Add Employee");
    System.out.println("2. Display Employees");
    System.out.println("3. Exit");
    System.out.print("Enter your choice: ");
    choice = scanner.nextInt();
    scanner.nextLine(); // Consume newline
    switch (choice) {
        case 1:
            // Add new employee
            System.out.print("Enter employee name: ");
            String name = scanner.nextLine();
            System.out.print("Enter employee ID: ");
            int id = scanner.nextInt();
            management.addEmployee(name, id);
            break;
        case 2:
            // Display all employees
            management.displayEmployees();
            break;
        case 3:
            System.out.println("Exiting the system.");
            break;
        default:
            System.out.println("Invalid choice. Please try again.");
    }
```

```
    } while (choice != 3);
  }
```

OUTPUT:

```
Employee Management System
1. Add Employee
2. Display Employees
3. Exit
Enter your choice: 1
Enter employee name: kashaf
Enter employee ID: 248
Employee added successfully.

Employee Management System
1. Add Employee
2. Display Employees
3. Exit
Enter your choice: 1
Enter employee name: fatima
Enter employee ID: 221
Employee added successfully.
```

```
Employee Management System
1. Add Employee
2. Display Employees
3. Exit
Enter your choice: 2
Employee Records:
Employee{name='kashaf', id=248}
Employee{name='fatima', id=221}
```

```
Employee Management System
1. Add Employee
2. Display Employees
3. Exit
Enter your choice: 3
Exiting the system.
BUILD SUCCESSFUL (total time: 1 minute 29 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

CODE:

COLOUR CLASS:

```
public class Color {
    private int red;   // Red value (0-255)
   private int green; // Green value (0-255)
   private int blue;  // Blue value (0-255)
   public Color(int red, int green, int blue) {
      this.red = red;
      this.green = green;
```

```
        this.blue = blue;}
    public int getRed() {
        return red;
    public int getGreen() {
        return green;   }
    public int getBlue() {
        return blue;  }
    @Override
    public int hashCode() {
        return Objects.hash(red, green, blue);}
    @Override
    public boolean equals(Object obj) {
        if (this == obj) return true; // Check for reference equality
        if (obj == null || getClass() != obj.getClass()) return false; // Check if the object is null or of different class
        Color color = (Color) obj; // Cast the object to Color
        return red == color.red && green == color.green && blue == color.blue; // Check for RGB equality}
    @Override
    public String toString() {
        return "Color{" +
                "red=" + red +  ", green=" + green +  ", blue=" + blue +  '}';
```

MAIN CLASS:

```
public class ColorTest {
     public static void main(String[] args) {
        Color color1 = new Color(255, 0, 0); // Red
        Color color2 = new Color(0, 255, 0); // Green
        Color color3 = new Color(0, 0, 255); // Blue
```

```
Color color4 = new Color(255, 0, 0); // Another Red
System.out.println("Is color1 equal to color2? " + color1.equals(color2));
System.out.println("Is color1 equal to color4? " + color1.equals(color4));
System.out.println("Color 1: " + color1);   System.out.println("Color 2: " + color2);
System.out.println("Color 3: " + color3);
System.out.println("Color 4: " + color4); }}
```

OUTPUT: