Lab # 02

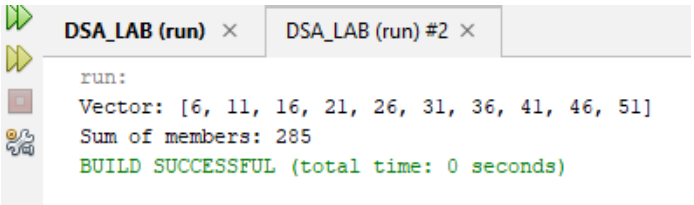
Lab Tasks:

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

Code:

```
package dsa_lab;
import java.util.Vector;

public class DSA_Lab_2 {

   public static void main(String[] args) {

Vector<Integer> vec1=new Vector<Integer>(10);
        int n=6;
        for(int i=0;i<10;i++){
           vec1.add(n);
           n=n+5;
        }
        System.out.print("Vector: ");
        System.out.print(vec1);

        int sum=vec1.stream().mapToInt(Integer::intValue).sum();
        System.out.println("\nSum of members: "+sum);
   }

}
```

Output:



Task #2: Create a ArrayList of string. Write a menu driven program which:
a. Displays all the elements
b. Displays the largest String

Code:

```
package dsa_lab;
import java.util.*;

public class DSA_Lab_2 {

   public static void main(String[] args) {
     //Task #2
      ArrayList<String> ar1 = new ArrayList<String>();
      Scanner sc = new Scanner(System.in);
      int choice;

      System.out.println("Enter five menu  Items:");
      for (int i = 0; i < 5; i++) {
         ar1.add(sc.nextLine());
      }
      do {
         System.out.println("\nMenu:");
         System.out.println("1. Display all items");
         System.out.println("2. Display the largest item");
         System.out.println("3. Exit");
         System.out.print("Enter your choice: ");
         choice = sc.nextInt();
         sc.nextLine();
         switch (choice) {
            case 1:

               System.out.print("Menu Display:");
               for (String e : ar1) {
                     System.out.println(item);
               }
               break;
            case 2:
               String largest = " ";
               for (String item : ar1) {
                  if (item.length() > largest.length()) {
                     largest = item;
                  }
               }
               System.out.println("Largest Item: " + largest);
               break;

            case 3:
               System.out.println("Exiting program.");
               break;

            default:
               System.out.println("Invalid choice. Please try again.");
         }
      }
```

```
        while (choice != 3);

          sc.close();
        }

    }

}
```

Output:

```
Output - DSA_LAB (run) #4   Java Call Hierarchy
run:
Enter five menu  Items:
Biryani
karahi
stick boti
Tikka
chowmen

Menu:
1. Display all items
2. Display the largest item
3. Exit
Enter your choice: 1
Menu Display:Biryani
karahi
stick boti
Tikka
chowmen

Menu:
1. Display all items
2. Display the largest item
3. Exit
Enter your choice: 2
Largest Item: stick boti

Menu:
1. Display all items
2. Display the largest item
3. Exit
Enter your choice: 3
Exiting program.
BUILD SUCCESSFUL (total time: 41 seconds)
```

Task #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:

```
package dsa_lab;
import java.util.*;

//Task #3
class Employee implements Comparable<Employee> {
    private String name;
    private String gender;
    private int id;
    private int yearOfJoining;

    public Employee(int id, String name, int yearOfJoining, String gender) {
        this.name = name;
        this.gender = gender;
        this.yearOfJoining = yearOfJoining;
        this.id = id;
    }

    public int getYearOfJoining() {
        return yearOfJoining;
    }

    @Override
    public int compareTo(Employee other) {
        // Ascending order by year of joining
        return this.yearOfJoining - other.yearOfJoining;
    }

    @Override
    public String toString() {
        return "ID: " + id + ", Name: " + name + ", Gender: " + gender + ", Year of Joining: " + yearOfJoining;
    }

    // Comparator for sorting by year of joining in descending order
    public static Comparator<Employee> YearDescendingComparator = new Comparator<Employee>() {
        @Override
        public int compare(Employee e1, Employee e2) {
            return e2.getYearOfJoining() - e1.getYearOfJoining();
        }
    };

    public static void main(String[] args) {
        ArrayList<Employee> employees = new ArrayList<>();
        employees.add(new Employee(103, "Hassan", 2020, "Male"));
        employees.add(new Employee(180, "Jameel", 2024, "Male"));
        employees.add(new Employee(150, "Aisha", 2019, "Female"));
        employees.add(new Employee(80, "Zia", 2009, "Male"));

        // Sorting employees by year of joining in ascending order (Comparable)
        Collections.sort(employees);
```

```
        System.out.println("Employees sorted by year of joining (ascending):");
        for (Employee employee : employees) {
            System.out.println(employee);
        }

        // Sorting employees by year of joining in descending order (Comparator)
        Collections.sort(employees, Employee.YearDescendingComparator);
        System.out.println("\nEmployees sorted by year of joining (descending):");
        for (Employee employee : employees) {
            System.out.println(employee);
        }
    }
}
```

Output:

```
Output - DSA_LAB (run) #4   Java Call Hierarchy
run:
Employees sorted by year of joining (ascending):
ID: 80, Name: Zia, Gender: Male, Year of Joining: 2009
ID: 150, Name: Aisha, Gender: Female, Year of Joining: 2019
ID: 103, Name: Hassan, Gender: Male, Year of Joining: 2020
ID: 180, Name: Jameel, Gender: Male, Year of Joining: 2024

Employees sorted by year of joining (descending):
ID: 180, Name: Jameel, Gender: Male, Year of Joining: 2024
ID: 103, Name: Hassan, Gender: Male, Year of Joining: 2020
ID: 150, Name: Aisha, Gender: Female, Year of Joining: 2019
ID: 80, Name: Zia, Gender: Male, Year of Joining: 2009
BUILD SUCCESSFUL (total time: 0 seconds)
```

Task #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:

```
package dsa_lab;
import java.util.*;

public class DSA_Lab_2 {

    public static void main(String[] args) {
        //Task #4
        Vector<Integer> vec2=new Vector<Integer>(10);
        //Initializing value
        int n=6;
        for(int i=0;i<10;i++){
            vec2.add(n);
```

```
            n=n+5;
        }
        System.out.print("Vector: ");
        System.out.print(vec2);
        System.out.println();

        //Calculate sum and max value
        int sum=0;
        int max=0;
        for(int c:vec2){
            if(c>max){
                max=c;
            }
            sum+=c;
        }
        System.out.println("Sum is: "+sum);
        System.out.println("Max Value is: "+max);
    }

}
```

Output:

```
Output - DSA_LAB (run) #4   Java Call Hierarchy
run:
Vector: [6, 11, 16, 21, 26, 31, 36, 41, 46, 51]
Sum is: 285
Max Value is: 51
BUILD SUCCESSFUL (total time: 0 seconds)
```

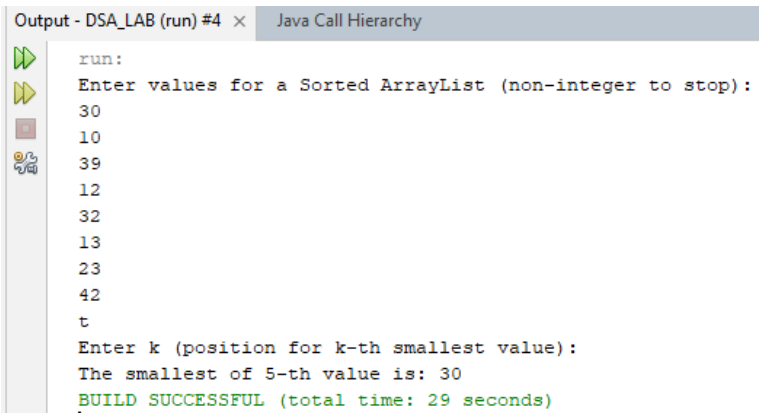
Task #5: Find the k-th smallest element in a sorted ArrayList

Code:

```
package dsa_lab;
import java.util.Vector;

public class DSA_Lab_2 {

    public static void main(String[] args) {
        ArrayList<Integer> ark = new ArrayList<>();
        Scanner sc = new Scanner(System.in);

        System.out.println("Enter values for a Sorted ArrayList (non-integer to stop):");
        while (sc.hasNextInt()) {
            ark.add(sc.nextInt());
        }

        System.out.println("Enter k (position for k-th smallest value):");
        int k = 5;

        // Sorting ArrayList
```

```
        Collections.sort(ark);

        if (k > 0 && k <= ark.size()) {  // Adjust condition to include the last element
            int kthSmallest = ark.get(k - 1); // k-1 to access k-th element correctly
            System.out.println("The smallest of " + k + "-th value is: " + kthSmallest);
        } else {
            System.out.println("k must be between 1 and " + ark.size());
        }
    }

}
```

Output:



Task #6: W rite a program to merge two ArrayLists into one.

Code:

```
package dsa_lab;
import java.util.Vector;
public class DSA_Lab_2 {
    public static void main(String[] args) {
        ArrayList<Integer> m1=new ArrayList<Integer>();
        m1.add(20);
        m1.add(59);
        m1.add(93);
        ArrayList<Integer> m2=new ArrayList<Integer>();
        m2.add(89);
        m2.add(30);

        ArrayList<Integer> merged=new ArrayList<Integer>();
        merged.addAll(m1);
        merged.addAll(m2);

        System.out.println("1st ArrayList: "+m1);
        System.out.println("2nd ArrayList: "+m2);
        System.out.println("Merged ArrayList: "+merged);
        Collections.sort(merged);
        System.out.println( "Sorted Merged ArrayList: "+merged);    }
}
```

Output:

Output - DSA_LAB (run) #4 × Java Call Hierarchy

```
run:
1st ArrayList: [20, 59, 93]
2nd ArrayList: [89, 30]
Merged ArrayList: [20, 59, 93, 89, 30]
Sorted Merged ArrayList: [20, 30, 59, 89, 93]
BUILD SUCCESSFUL (total time: 0 seconds)
```

Home Tasks:

Task #1: Create a Vector storing integer objects as an input.

- **Sort the vector**
- **Display largest number**
- **Display smallest number**

Code:

```
public class Dsa_Lab2_HomeT {
  public static void main(String[] agrs){
      /*Vector<Integer> vect=new Vector<Integer>();
      Scanner sc=new Scanner(System.in);
      System.out.println("Enter 10 integers: ");
      //Adding Integer
      for(int i=0;i<10;i++){
         vect.add(sc.nextInt());
      }
      System.out.print("\nVector: ");
      for(int i:vect){
         System.out.print(i+" ");
      }
      System.out.print("\nSorted Vector: ");
      Collections.sort(vect);
      for(int i:vect){
         System.out.print(i+" ");
      }
      //Since Vector is Sorted
      System.out.print("\nLargest Integer: ");
      System.out.println(vect.getLast());

      System.out.print("Smallest Integer: ");
      System.out.println(vect.getFirst());
    }
}
```

Output:

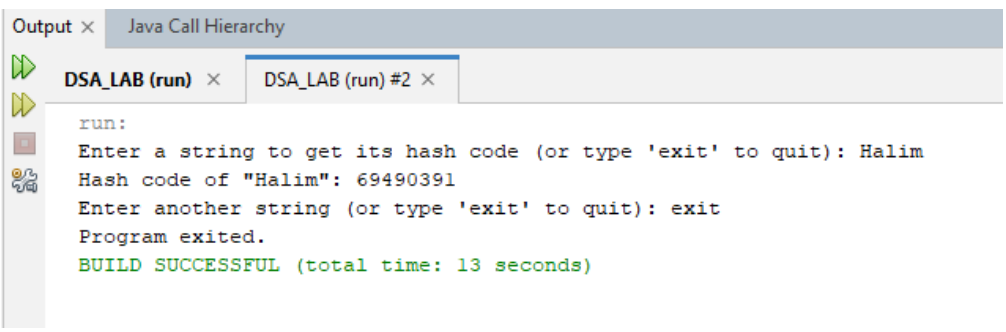
```
Output ×   Java Call Hierarchy
DSA_LAB (run) ×   DSA_LAB (run) #2 ×
run:
Enter 10 integers:
2
89
81
29
05
605094
57
12
13
42

Vector: 2 89 81 29 5 605094 57 12 13 42
Sorted Vector: 2 5 12 13 29 42 57 81 89 605094
Largest Integer: 605094
Smallest Integer: 2
BUILD SUCCESSFUL (total time: 22 seconds)
```

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

Code:

```
public class Dsa_Lab2_HomeT {
  public static void main(String[] agrs){
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter a string to get its hash code (or type 'exit' to quit): ");
        String input = scanner.nextLine();

        while (!input.equalsIgnoreCase("exit")) {
            int hashCode = input.hashCode();
            System.out.println("Hash code of \"" + input + "\": " + hashCode);

            System.out.print("Enter another string (or type 'exit' to quit): ");
            input = scanner.nextLine();
        }

        System.out.println("Program exited.");
        scanner.close();
    }
}
```

Output:



Task #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

- **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.**

Code:

```
package dsa_lab;
import java.util.*;

class Employee {

    private String name;
    private int id;

    public Employee(String name, int id) {
        this.name = name;
        this.id = id;
    }

    @Override
    public int hashCode() {
        int result = 11;
        result =/*Prime No*/ 31 * result/*11*/ + name.hashCode();
        result =/*Prime No*/ 31 * result/*11*/ + id;
        return result;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) {
            return true;
        }
        if (o == null || getClass() != o.getClass()) {
            return false;
        }

        Employee e = (Employee) o;
        return id == e.id && name.equals(e.name);
    }

    @Override
    public String toString() {
        return "Employee: {Name: " + name + " Id: " + id + "}";
    }
}

class EmployeeManagement {

    private HashSet<Employee> hse;

    public EmployeeManagement() {
        hse = new HashSet<>();

    }
```

```
    public boolean addEmployee(Employee emp) {
        if (hse.contains(emp)) {
            System.out.println("Employee already exists: " + emp);
            return false;
        } else {
            hse.add(emp);
            System.out.println("Employee added: " + emp);
            return true;

        }
    }

    public boolean employeeExists(Employee employee) {
        return hse.contains(employee);
    }

    // Display all employees
    public void displayAllEmployees() {
        if (hse.isEmpty()) {
            System.out.println("No employees found.");
        } else {
            System.out.println("Employee Records:");
            for (Employee emp : hse) {
                System.out.println(emp);
            }
        }
    }
}

public class HomeTaskEmployee {

    public static void main(String[] args) {
        EmployeeManagement em = new EmployeeManagement();
        Scanner sc = new Scanner(System.in);

        while (true) {
            System.out.println("Select an Option:");
            System.out.println("1) Add an Employee:");
            System.out.println("2) Check Employee");
            System.out.println("3) Display all");
            System.out.println("4) Exit \n");
            int choice = sc.nextInt();

            switch (choice) {
                case 1:
                    System.out.print("Enter Name: ");
                    String n = sc.nextLine(); // Corrected: properly captures name
                    System.out.print("Enter Id: ");
                    int i = sc.nextInt();
                    sc.nextLine(); // Consume newline after integer input
                    Employee emp = new Employee(n, i);
                    em.addEmployee(emp);
                    break;
                case 2:
                    System.out.print("Enter Name: ");
                    String na = sc.nextLine();
                    System.out.print("Enter Id: ");
                    int id = sc.nextInt();
                    sc.nextLine(); // Consume newline after integer input
                    Employee empl = new Employee(na, id);
                    if (em.employeeExists(empl)) {
                        System.out.println("Employee exists: " + empl);
```

```
            } else {
                System.out.println("Employee not found.");
            }
            break;
        case 3:
            em.displayAllEmployees();
            break;
        case 4:
            System.out.println("Exiting Program");
            sc.close();
            return;
        default:
            System.out.println("Invalid Choice");
            break;
    }
  }
 }
}
```

Output:

Output × Java Call Hierarchy

DSA_LAB (run) × DSA_LAB (run) #2 × DSA_LAB (run) #3 × DSA_LAB (run) #4 ×

```
run:
Select an Option:
1) Add an Employee:
2) Check Employee
3) Display all
4) Exit

1
Enter Name: Enter Id: 193
Employee added: Employee: {Name:  Id: 193}
Select an Option:
1) Add an Employee:
2) Check Employee
3) Display all
4) Exit

2
Enter Name: Enter Id: 193
Employee exists: Employee: {Name:  Id: 193}
Select an Option:
1) Add an Employee:
2) Check Employee
3) Display all
4) Exit

3
Employee Records:
Employee: {Name:  Id: 193}
Select an Option:
1) Add an Employee:
2) Check Employee
3) Display all
4) Exit

4
Exiting Program
BUILD SUCCESSFUL (total time: 23 seconds)
```

Task#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:

```
package dsa_lab;

class Color{
    private int red;
    private int green;
    private int blue;
    public Color(int r,int g,int b ){
        red=r;
        green=g;
        blue=b;
    }
    @Override
    public int hashCode(){
        int result=31*red+31*green+31*blue;
        return result;
    }
    @Override
    public boolean equals(Object obj){
        if(this==obj)return true;
        if(obj==null||getClass()!=obj.getClass())return false;
        Color c=(Color) obj;
        return red==c.red &&green==c.green&& blue==c.blue;
    }
    @Override
    public String toString() {
        return "Color{red=" + red + ", green=" + green + ", blue=" + blue + "}";
    }
}
public class HomeTaskColor {
    public static void main(String[] args ){
        Color c1=new Color(250,0,90);
        Color c2=new Color(250,0,0);
        Color c3=new Color(250,0,0);

        System.out.println("c1 compare to c2:"+c1.equals(c2));
        System.out.println("c2 compare to c3:"+c2.equals(c3));

        System.out.println("color1 hashCode: " + c1.hashCode());
        System.out.println("color2 hashCode: " + c2.hashCode());
        System.out.println("color3 says " + c3.toString());
    }
}
```

Output:

```
Output ×   Java Call Hierarchy
DSA_LAB (run) ×   DSA_LAB (run) #2 ×   DSA_LAB (run) #3 ×
run:
c1 compare to c2:false
c2 compare to c3:true
color1 hashCode: 10540
color2 hashCode: 7750
color3 says Color{red=250, green=0, blue=0}
BUILD SUCCESSFUL (total time: 0 seconds)
```