Object Oriented Programming (JAVA)



## Semester: Fall 2024

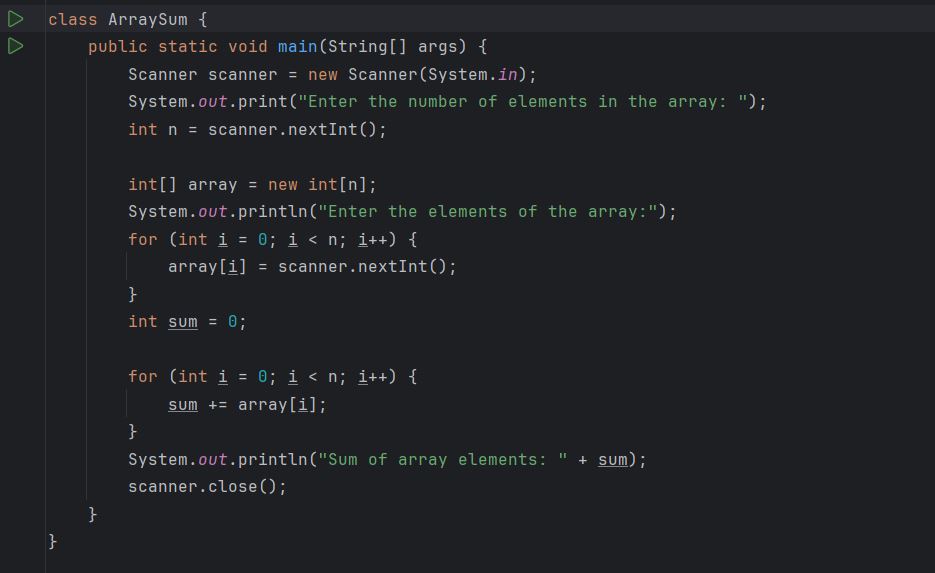
**Software Engineering**

**Faculty of Information Technology UCP Lahore, Pakistan**

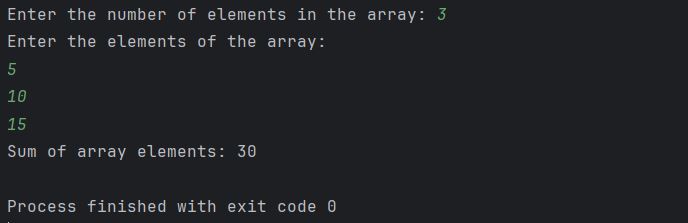
**Instructor: Danish Ali Khan**

|  |  |
| --- | --- |
| **Week 3** | |
| **Topic** | **Arrays, Strings and Array Lists in Java** |
| **Objective** | * Introduce the usage of arrays and strings in Java * Understanding ArrayLists in Java * Learn about Java string immutability and basic string operations * Practice using common Java string methods for manipulation and comparison. |

**Lab task: A program to find the sum of all elements in an array.**

****

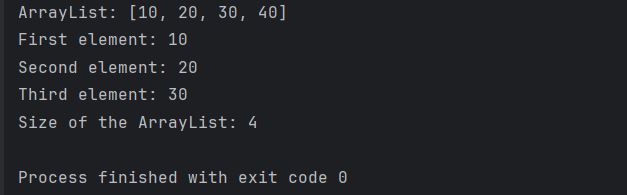
**Output:**

****

**Lab task: A program to demonstrate the use of ArrayList in java**

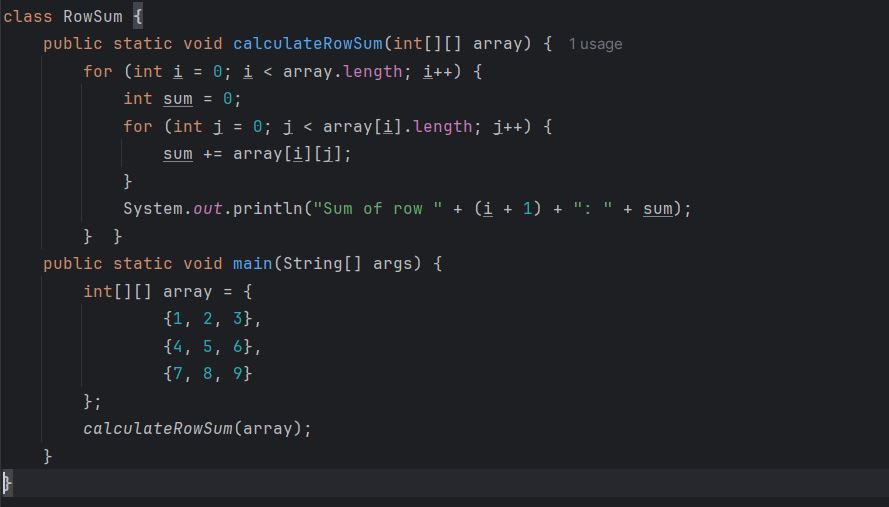
****

**Output:**

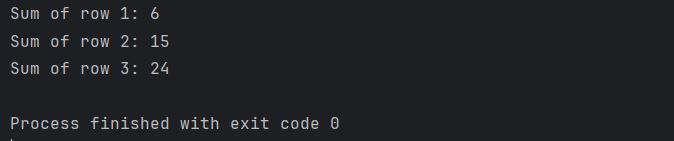
****

**In a 2D array: arrayName[rowIndex][columnIndex]**

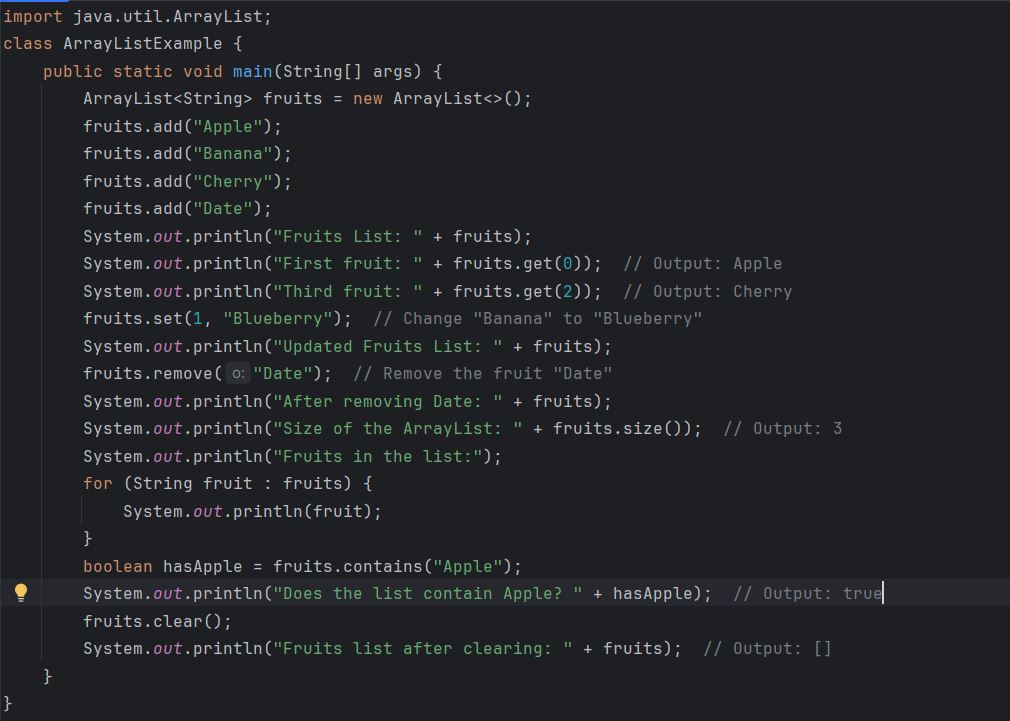
**Lab Task: A program to calculate the sum of elements in each row of a 2D array using a method:**

****

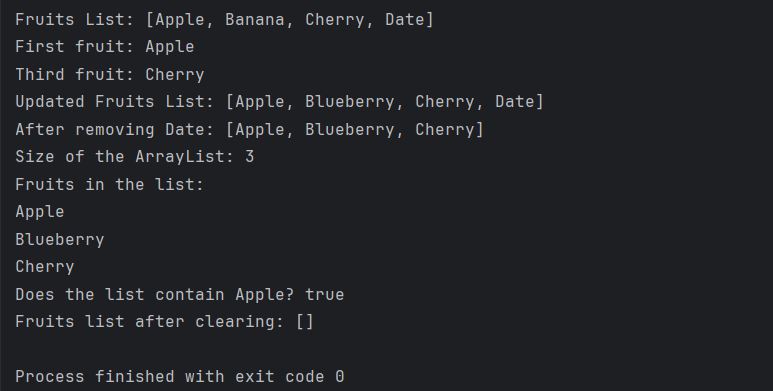
**Output:**

****

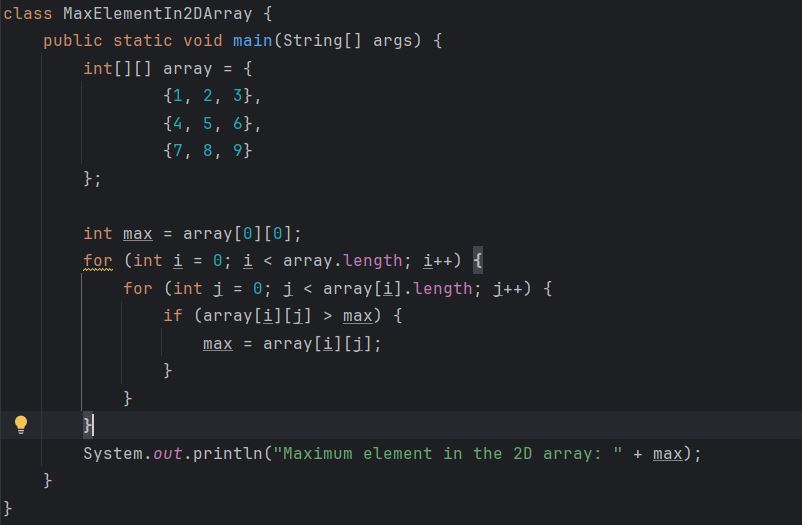
**Lab Task: Demonstrating the use of ArrayList in java with some built in functions of ArrayList.**

****

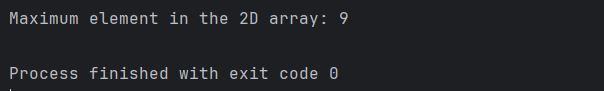
**Output:**

****

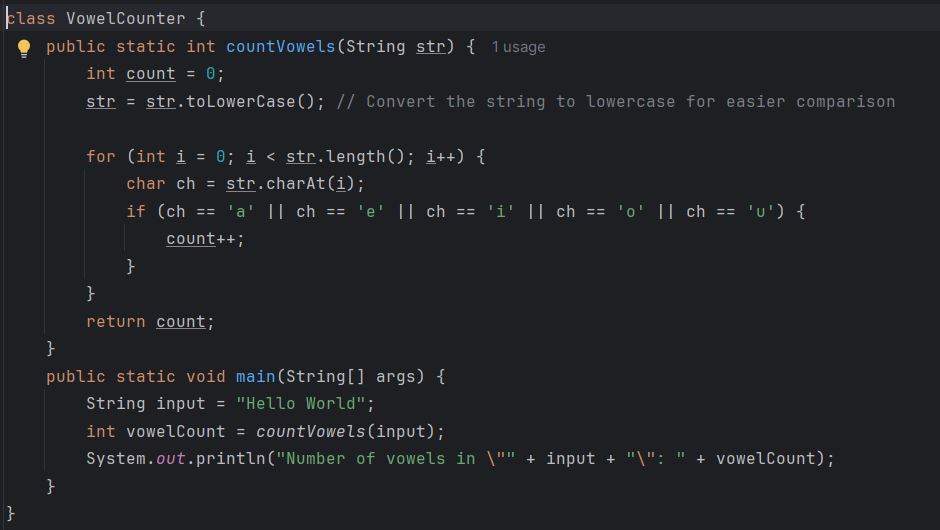
**Lab Task: A program to find the maximum element in a 2D array.**

****

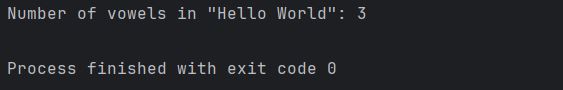
**Output:**

****

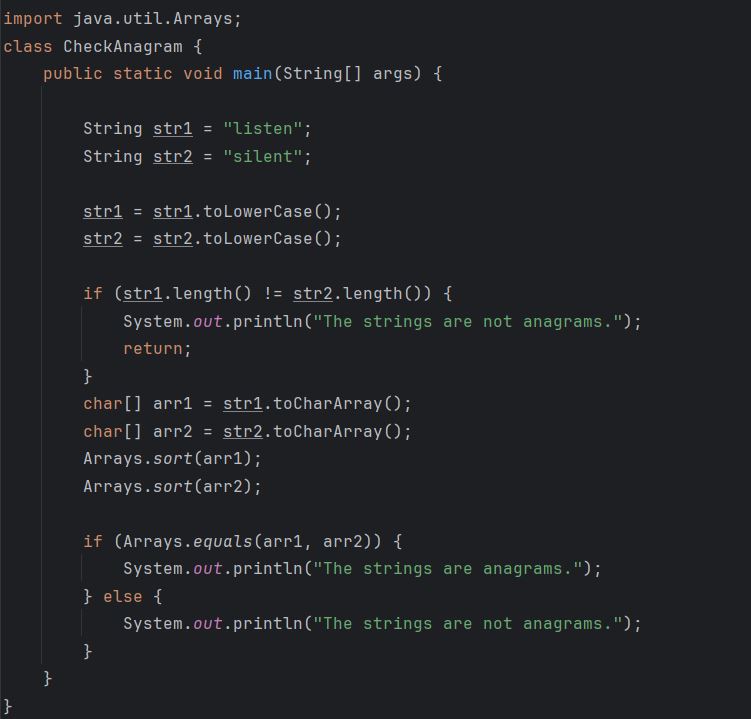
**Lab Task: A program to count the number of vowels in a given string.**

****

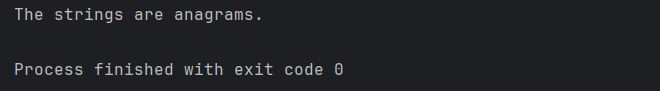
**Output:**

****

**Lab Task: A program to check if two strings are anagrams of each other. Two strings are anagrams if they contain the same characters in the same frequency.**



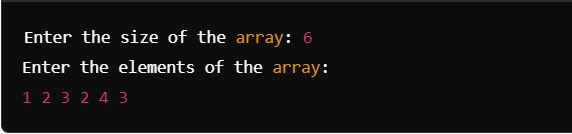
**Output:**

****

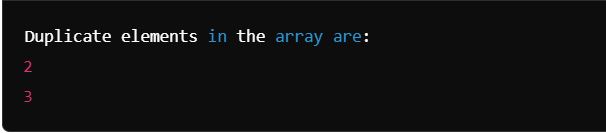
**Basic Lab Tasks:**

1. Write a Java program to find the duplicate elements in a one-dimensional array.

**Sample Input:**

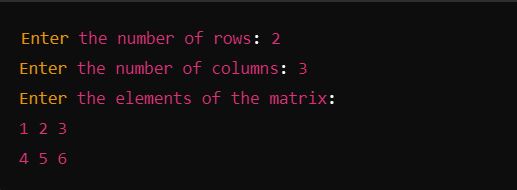


**Sample Output:**

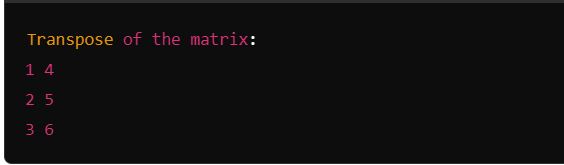


1. Write a program to find the transpose of a 2D array. The transpose of a matrix is obtained by swapping rows with columns.

**Sample Input:**

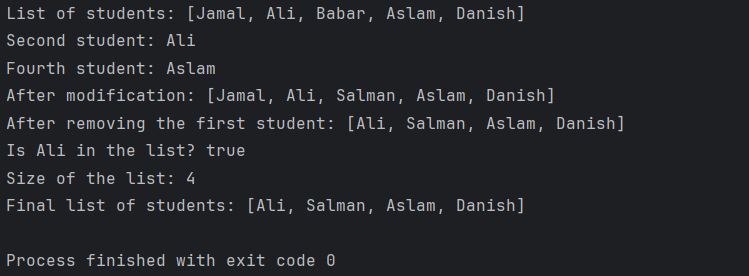
****

**Sample Output:**

****

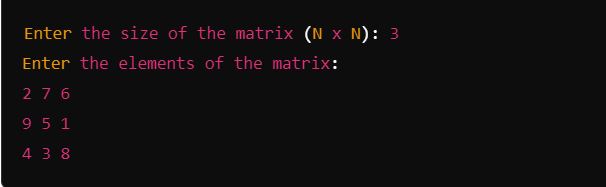
1. **Write a Java program to perform the following tasks using ArrayList:**
2. Create an ArrayList to store names of students.
3. Add at least 5 names of students to the ArrayList.
4. Print the list of students.
5. Access and print the second and fourth student's name from the list.
6. Modify the third student's name to a new name.
7. Remove the first student from the list.
8. Check if a specific student (e.g., "Ali") is present in the list.
9. Print the size of the list.
10. Print the final list of students.

**Sample Output:**



1. Write a program to check if a given 2D array is a magic square. A magic square is a grid where the sum of every row, column, and diagonal is the same.

**Sample Input:**



**Sample Output:**

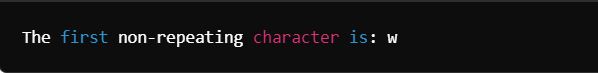
****

1. Write a program to find the first non-repeating character in a string.

**Sample Input:**

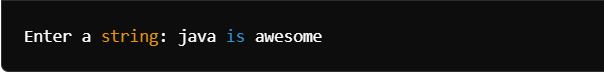


**Sample Output:**

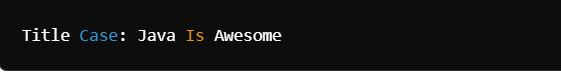


1. Write a program to convert a given string to title case (where the first letter of each word is capitalized).

**Sample Input:**

****

**Sample Output:**

****

**Scenario Based Tasks**

**Task 1: Static Arrays**

**Instructions:**

* Create a static array of 5 integers to represent brightness levels of different smart lights.
* Initialize the array with values and print the brightness levels using a loop.
* Modify the brightness of one light and reprint the array.

**Key Difference (Java vs. C++):**

In Java, arrays are objects and have a fixed size, unlike C++ where arrays are also fixed but can be manipulated more flexibly using pointers.

|  |
| --- |
| Sample output:  Initial Brightness Levels:  Light 1: 0%  Light 2: 50%  Light 3: 75%  Light 4: 100%  Light 5: 25%  Updated Brightness Levels:  Light 1: 0%  Light 2: 50%  Light 3: 90%  Light 4: 100%  Light 5: 25% |

**Task 2: Introduction to ArrayLists**

**Instructions:**

* Create an ArrayList to store brightness levels for the smart lights.
* Add brightness levels to the list dynamically.
* Modify and remove brightness levels.
* Print the brightness levels using a loop.

**Key Difference (Java vs. C++):**

Java’s ArrayList is dynamic and resizes automatically, while in C++ dynamic arrays require manual memory management (e.g., using vectors or pointers).

|  |
| --- |
| Sample output:  Brightness Levels:  Light 1: 0%  Light 2: 50%  Light 3: 75%  Light 4: 100%  Light 5: 25%  Updated Brightness Levels:  Light 1: 0%  Light 2: 90%  Light 3: 75%  Light 4: 100% |

**Task 3: Creating and Initializing Strings**

**Instructions:**

* Create a string to represent the mode of the smart light (e.g., "Energy Saving Mode").
* Concatenate the mode with other strings to simulate different smart light modes.
* Demonstrate immutability by trying to modify the string directly (and showing that it won't work).

**Key Difference (Java vs. C++):**

Strings in Java are immutable, meaning their value cannot change after creation. In C++, std::string is mutable.

|  |
| --- |
| Sample output:  Current Mode: Energy Saving Mode  Updated Mode: Energy Saving Mode - Night Time  String immutability prevents direct modification. |

**Task 4: Common String Methods**

**Instructions:**

* Create a string to represent the smart light’s mode (e.g., "Energy Saving Mode").
* Use and print the results of common string methods such as length(), toUpperCase(), toLowerCase(), charAt(), indexOf(), replace(), and substring().

**Key Difference (Java vs. C++):**

Java provides built-in methods for strings like length(), toUpperCase(), replace(), which are simpler than C++ functions from libraries like <cstring> or <string>.

|  |
| --- |
| Sample output:  Length of mode: 18  Mode in uppercase: ENERGY SAVING MODE  Mode in lowercase: energy saving mode  Character at index 5: y  Index of 'Saving': 7  Updated Mode: Energy Efficiency Mode  Substring (first 6 characters): Energy |

**Task 5: Smart Light Controller - Full Simulation**

**Instructions:**

**Part A (Static Array):**

Create a static array to represent the brightness levels of 5 smart lights. Initialize them with random values (between 0 and 100) and display them.

**Part B (Dynamic Array - ArrayList):**

Use an ArrayList to store the brightness levels dynamically. Populate the ArrayList with the same values as the static array, then allow the user to modify and remove brightness levels interactively. Ensure any new values added to the list are between 0 and 100. If a value exceeds 100, set it to 100 and display a warning.

**Part C (String Operations):**

Create a string to represent the mode of the smart light (e.g., "Energy Saving Mode"). Use string operations to:

* Check the length of the mode.
* Convert the mode to uppercase and lowercase.
* Extract and display specific substrings (e.g., the first word).
* Replace the word "Energy" with "Efficiency."

**Part D (Final Output):**

Combine the results of the static array, ArrayList, and string operations. Display the final brightness levels of the lights, the current mode, and a message confirming that the Smart Light Controller is successfully configured.

|  |
| --- |
| Sample output:  Static Array - Initial Brightness Levels:  Light 1: 25%  Light 2: 50%  Light 3: 75%  Light 4: 100%  Light 5: 30%  Modify brightness levels (Enter new values):  Enter new brightness for Light 1: 85  Enter new brightness for Light 2: 110  Warning: Brightness set to 100% (maximum allowed).  Enter new brightness for Light 3: 60  Enter new brightness for Light 4: 95  Enter new brightness for Light 5: 50  Do you want to remove any light's brightness level? (yes/no): yes  Enter the light number to remove: 5  Light 5 brightness level removed.  Dynamic Array - Updated Brightness Levels:  Light 1: 85%  Light 2: 100%  Light 3: 60%  Light 4: 95%  Smart Light Mode: Energy Saving Mode  Length of mode: 18  Mode in uppercase: ENERGY SAVING MODE  Mode in lowercase: energy saving mode  First word of mode: Energy  Updated Mode: Efficiency Saving Mode  Final Configuration of Smart Light Controller:  Brightness Levels (Final):  Light 1: 85%  Light 2: 100%  Light 3: 60%  Light 4: 95%  Current Mode: Efficiency Saving Mode  Smart Light Controller successfully configured! |