Sukkur Institute of Business Administration University

Department of Computer Science

Object Oriented Programming using Java

BS – II (CS/AI/SE)

Spring-2024

Lab # 03: Java Essentials Lab: Exploring Literals, operators, and Arrays

Instructor: Nimra Mughal

Objectives

After performing this lab, students will be able to understand:

- Use of literals
- Use of operators
- Arrays
    a. One Dimensional
    b. Multi-Dimensional
    c. Jagged array
- Java Documentation basics

Literals

In computer programming, a literal refers to the notation used to represent fixed values directly in the source code of a program. Here's a brief explanation of literals for different types:

1. **Integer literals**: These represent whole numbers without any fractional or decimal part. They can be expressed in decimal, binary, octal, or hexadecimal notation.
2. **Floating-point literals**: These represent numbers with a fractional part. They can be written in standard decimal notation or in scientific notation (exponent form).
3. **Character literals**: These represent single characters enclosed within single quotes (' '). They can also represent special characters using escape sequences (e.g., '\n' for newline).
4. **String literals**: These represent sequences of characters enclosed within double quotes (" "). Strings are used to represent text or sequences of characters.
5. **Boolean literals**: These represent the two boolean values: true and false. They are used to represent logical values in boolean expressions.
6. **Null literal**: This represents the absence of a value or a reference to no object.

```
public class LiteralExample {
    public static void main(String[] args) {
        // Integer literals
        int number = 10;
        int binaryNumber = 0b1010;
        int octalNumber = 012;
        int hexNumber = 0xA;

        // Floating-point literals
        double doubleValue = 3.14;
        float floatValue = 3.14f;
        double exponentValue = 6.022e23; // Floating-point literal in exponent form

        // Character literals
        char character = 'A';
        char unicodeChar = '\u0041';

        // String literals
        String str = "Hello, World!";

        // Boolean literals
        boolean boolTrue = true;
        boolean boolFalse = false;

        // Null literal
        Object obj = null;
```

```
        // Printing the values
        System.out.println("Integer literals:");
        System.out.println("number: " + number);
        System.out.println("binaryNumber: " + binaryNumber);
        System.out.println("octalNumber: " + octalNumber);
        System.out.println("hexNumber: " + hexNumber);

        System.out.println("\nFloating-point literals:");
        System.out.println("doubleValue: " + doubleValue);
        System.out.println("floatValue: " + floatValue);
        System.out.println("exponentValue: " + exponentValue);

        System.out.println("\nCharacter literals:");
        System.out.println("character: " + character);
        System.out.println("unicodeChar: " + unicodeChar);

        System.out.println("\nString literals:");
        System.out.println("str: " + str);

        System.out.println("\nBoolean literals:");
        System.out.println("boolTrue: " + boolTrue);
        System.out.println("boolFalse: " + boolFalse);

        System.out.println("\nNull literal:");
        System.out.println("obj: " + obj);
    }
}
```

Arrays in java

Making an array in a Java program involves three distinct steps:

- Declare the array name.
- Create the array.
- Initialize the array values.

We refer to an array element by putting its index in square brackets after the array name.

To use an array in a program, you must declare a variable to reference the array and specify the array's *element type*.

Syntax:

```
elementType[] arrayRefVar;
```

The **`elementType`** can be any data type, and all elements in the array will have the same data type.
Say suppose :
int[] arrayRef;
Unlike declarations for primitive data type variables, the declaration of an array variable does not allocate any space in memory for the array. It creates only a storage location for the reference to an array.
If a variable does not contain a reference to an array, the value of the variable is **null**. You cannot assign elements to an array unless it has already been created. After an array variable is declared, you can create an array by using the **new** operator and assign its reference to the variable with the following **syntax**:

```
arrayRefVar = new elementType[arraySize];
int[] arr; //integer array
arr = new int[10];
```

Java has a shorthand notation, known as the *array initializer*, which combines the declaration, creation, and initialization of an array in one statement using the following **syntax**:

```
elementType[] arrayRefVar = {value0, value1, ..., valuek};
```

Arrays Class

Arrays class which is in `java.util.Arrays` package, is a provision by Java that provides you a number of methods through which arrays can be manipulated. This class also lets you perform sorting and searching operations on an array.

https://www.geeksforgeeks.org/array-class-in-java/

https://docs.oracle.com/javase/8/docs/api/java/util/Arrays.html

1D-Array Example

```java
public class ArrayExample1D {
    public static void main(String[] args) {
        // Declaring and initializing an array of integers
        int[] numbers = {1, 2, 3, 4, 5};

        // Accessing elements of the array
        int thirdNumber = numbers[2]; // Accessing the third element (index 2)
        System.out.println("Third number is: " + thirdNumber);

        // Iterating through the array
        System.out.println("Elements of the array:");
        for (int i = 0; i < numbers.length; i++) {
            System.out.println("Element at index " + i + ": " + numbers[i]);
        }

        // Array of strings
        String[] names = {"Alice", "Bob", "Charlie"};
```

```
        // Accessing elements of the array
        String firstName = names[0];
        System.out.println("First name: " + firstName);
    }
}
```

2D-Array java

```
public class TwoDArrayExample {
    public static void main(String[] args) {
        // Declaring and initializing a 2D array
        int[][] matrix = {
            {1, 2, 3},
            {4, 5, 6},
            {7, 8, 9}
        };
        // Accessing elements of the 2D array
        System.out.println("Element at row 3, column 1: " + matrix[2][0]);

        // Iterating through the 2D array
        System.out.println("Elements of the 2D array:");
        for (int i = 0; i < matrix.length; i++) {
            for (int j = 0; j < matrix[i].length; j++) {
                System.out.print(matrix[i][j] + " ");
            }
            System.out.println(); // Move to the next line after printing each row
        }
    }
}
```

```
public class TwoDArrayExample {
    public static void main(String[] args) {
        // Declaring and initializing a 2D array
        int[][] matrix = {
            {1, 2, 3},
            {4, 5, 6},
            {7, 8, 9}
        };

        // Accessing elements of the 2D array using foreach loop
        System.out.println("Accessing elements of the 2D array:");
        for (int[] row : matrix) {
            for (int element : row) {
                System.out.print(element + " ");
            }
```

```
            System.out.println(); // Move to the next line after printing each row
        }
    }
}
```

Jagged array (UnEven Array)

```
public class JaggedArrayExample {
    public static void main(String[] args) {
        // Declaring and initializing a jagged array
        int[][] jaggedArray = {
            {1, 2, 3},
            {4, 5},
            {6, 7, 8, 9}
        };

        // Accessing elements of the jagged array using foreach loop
        System.out.println("Accessing elements of the jagged array:");
        for (int[] innerArray : jaggedArray) {
            for (int element : innerArray) {
                System.out.print(element + " ");
            }
            System.out.println(); // Move to the next line after printing each row
        }
    }
}
```

Operators in Java

https://www.javatpoint.com/operators-in-java
https://docs.oracle.com/javase/tutorial/java/nutsandbolts/operators.html

```
public class Bitwise {
    public static void main(String[] args) {
        int a = 5;  // binary: 0101
        int b = 3;  // binary: 0011

        // Bitwise AND
        int resultAnd = a & b;  // result: 0001
        System.out.println("Bitwise AND: " + resultAnd);  // Output: 1

        // Bitwise OR
        int resultOr = a | b;  // result: 0111
```

```
        System.out.println("Bitwise OR: " + resultOr);  // Output: 7

        // Bitwise XOR
        int resultXor = a ^ b;  // result: 0110
        System.out.println("Bitwise XOR: " + resultXor);  // Output: 6

        // Bitwise NOT
        int resultNotA = ~a;  // result: 11111111111111111111111111111010 (2's
complement)
        System.out.println("Bitwise NOT of a: " + resultNotA);  // Output: -6

        // Left Shift
        int resultLeftShift = a << 2;  // result: 10100
        System.out.println("Left Shift of a: " + resultLeftShift);  // Output: 20

        // Right Shift
        int resultRightShift = a >> 1;  // result: 0001
        System.out.println("Right Shift of a: " + resultRightShift);  // Output: 2

        // Unsigned Right Shift
        int unsignedRightShift = -8 >>> 2;  // result:
00111111111111111111111111111110
        System.out.println("Unsigned Right Shift of -8: " +
unsignedRightShift);  // Output: 1073741822
    }
}
```

Java Comments

Single line	Multi-ilne	Documentation comment
//This is single line com ment	/* This is Multiline commet */	/** * *We can use various tags *or heading or author name *We can also use HTML tags * */

Java Documentation Comments

```java
import java.io.*;
/**
* Add Two Numbers!
* The AddNum program implements an application that
* simply adds two given integer numbers and Prints
* the output on the screen.
* <p>
* <b>Note:</b> Giving proper comments in your program makes it more
* user friendly and it is assumed as a high quality code.
*
* @author  Zara Ali
* @version 1.0
* @since   2014-03-31
*/
public class Account {

   /**
   * This is the main method which makes use of addNum method.
   * @param args Unused.
   * @exception IOException On input error.
   * @see IOException
   */
   public static void main(String args[]) throws IOException {
      AddNum obj = new AddNum();
      int sum = obj.addNum(10, 20);

      System.out.println("Sum of 10 and 20 is :" + sum);
   }
}
```

```
PS E:\Collaborative Learning\Spring_24_BS-II_OOP\Codes\lab3\doc> javac DocExample.java
PS E:\Collaborative Learning\Spring_24_BS-II_OOP\Codes\lab3\doc> javadoc DocExample.java
Loading source file DocExample.java...
Constructing Javadoc information...
Building index for all the packages and classes...
Standard Doclet version 21.0.2+13-LTS-58
Building tree for all the packages and classes...
Generating .\DocExample.html...
```

The javadoc tool recognizes the following tags:

https://www.tutorialspoint.com/java/java_documentation.htm

https://www.javatpoint.com/java-comments

Exercises

1. Create two multi-dimensional(2d) arrays of size **4 by 4** and perform:

    a. Matrix addition

    b. Matrix Multiplication

    c. Matrix subtraction

2. **You're creating a program to manage student grades for a class**. How would you use a 2D array to store corresponding grades of students for multiple assignments? Explain how you would initialize the array, input grades, calculate averages for each student, and identify students who need extra help.

```
PS E:\Collaborative Learning\Spring_24_BS-II_OOP\Codes\lab3> java StudentGradeManager
Enter grades for student 1:
Assignment 1: 45
Assignment 2: 67
Assignment 3: 23
Enter grades for student 2:
Assignment 1: 78
Assignment 2: 56
Assignment 3: 90
Students needing extra help:
Student 1: Average grade = 45.0
PS E:\Collaborative Learning\Spring_24_BS-II_OOP\Codes\lab3>
```

3. **You're tasked with creating a program to manage employee records** for a company. How would you use a 2D array to store information such as employee IDs, and salaries? (Suppose the salaries are stores in the Integer type)

    Explain how you would create and initialize the array, search for employees by ID, and calculate statistics such as average salary. (please provide a java code)

    a. Use Hexadecimal notation for emploee ids

4. **Bit Manipulation:**Write a Java program to count the number of set bits (bits with value 1) in a given integer.

```
PS E:\Collaborative Learning\Spring_24_BS-II_OOP\Codes\lab3> javac CountSetBits.java
PS E:\Collaborative Learning\Spring_24_BS-II_OOP\Codes\lab3> java CountSetBits
Enter any Number:
34
Number of set bits in "34": 2
PS E:\Collaborative Learning\Spring_24_BS-II_OOP\Codes\lab3>
```

5. **Check Power of Two:** Write a Java program to check if a given integer is a power of two or not. Use bitwise operators for the check.

6. **Detect Odd or Even without Modulo Operator:** Create a Java program to determine whether a given integer is odd or even without using the modulo (%) operator. Instead, use bitwise operators.

7. Write code that creates an array named **odds and stores all odd numbers between 1 and 30** into it using a for loop.

8. **Calculate Hamming Distance:** Write a Java a code to calculate the Hamming distance between two integers. Hamming distance is the number of positions at which the corresponding bits are different. Use bitwise operators to perform the calculation.

```
PS E:\Collaborative Learning\Spring_24_BS-II_OOP\Codes\lab3> java HammingDista
nce
Hamming Distance between 7 and 4 is: 2
PS E:\Collaborative Learning\Spring_24_BS-II_OOP\Codes\lab3>
```

9. **Create a program to calculate the amount of tax owed based on income and tax brackets.**

   In this task, you will develop a program that computes the amount of tax owed by an individual based on their income and a set of predefined tax brackets. program will determine the tax owed by applying the appropriate tax rate to each portion of the income within the corresponding tax bracket and then summing up these amounts.

   For this exercise, we'll use the following standard tax brackets and rates:

   a. Income up to $10,000: Tax rate of 10%

   b. Income from $10,001 to $20,000: Tax rate of 15%

   c. Income above $20,000: Tax rate of 20%

```
PS E:\Collaborative Learning\Spring_24_BS-II_OOP\Codes\lab3> javac .\TaxCalculator.java
PS E:\Collaborative Learning\Spring_24_BS-II_OOP\Codes\lab3> java TaxCalculator
Enter your income: $14000
Total tax owed: $1600.0
PS E:\Collaborative Learning\Spring_24_BS-II_OOP\Codes\lab3>
```

10. **Create a class Person with fields for name, age, and address.**

- Use @author and @version tags in the class comment.
- Use @param and @return tags in method comments.
- Format comments with HTML tags for emphasis and lists (e.g., <b>).