CS4001 – Programming

Syntax and Semantics of Java Programming Language

Week 03: Workshop

Question 1:
Create "MathOperations.java" with all operator types

Test Case

Objectives	To compile and execute the program using Blue J.
Action	• Blue J was opened and the project containing the source code file MathOperations.java was loaded. • The program file MathOperations.java was compiled using the Compile button in blue J. • The compiled class MathOperations was executed by-right clicking the class and selecting the void main(String[] args) options.
Expected result	The program should compile and display the all operators types without any errors.
Actual result	The program successfully compile and display the all operators types without any errors.
Conclusion	Test was successful.

Screenshot of the code compilation

```java
public class MathOperations {
    public static void main(String[] args) {
        int a = 35;
        int b = 15;

        // 1. Arithmetic Operators
        int sum = a + b;
        int diff = a - b;
        int prod = a * b;
        int div = a / b;
        int mod = a % b;
        System.out.println("1. Arithmetic Operators");
        System.out.println("+ : " + sum); //Addition
        System.out.println("- : " + diff); //Subtraction
        System.out.println("* : " + prod); //Multiplication
        System.out.println("/ : " + div); //Division
        System.out.println("% : " + mod); //Modulus

        // Increment and Decrement (Unary Operators)
        int x = 7;
        System.out.println("Increment and Decrement Operators");
        System.out.println("x before increment: " + x);
        x++;
        System.out.println("x after x++ : " + x); //Increment
        x--;
        System.out.println("x after x-- : " + x); //Decremen

        // 2. Relational Operators
        System.out.println("2. Relational Operators");
        System.out.println("a > b : " + (a > b)); //Greater than
        System.out.println("a < b : " + (a < b)); //Less than
        System.out.println("a == b : " + (a == b)); //Equal to
        System.out.println("a != b : " + (a != b)); //Not equa
        System.out.println("a >= b : " + (a >= b)); //Greater or equal
        System.out.println("a <= b : " + (a <= b)); //Less or equal

        // 3. Logical Operators
        boolean p = true;
        boolean q = false;
        System.out.println("3. Logical Operators");
        System.out.println("p && q : " + (p && q)); //AND
        System.out.println("p || q : " + (p || q)); //OR
        System.out.println("!p : " + (!p)); //NOT

        // 4. Assignment Operators
        int y = 10;
        System.out.println("4. Assignment Operators");
        System.out.println("y = " + y);
        y += 5;
        System.out.println("y += 5 : " + y);
        y -= 3;
        System.out.println("y -= 3 : " + y);
        y *= 2;
        System.out.println("y *= 2 : " + y);
        y /= 4;
        System.out.println("y /= 4 : " + y);
        y %= 3;
        System.out.println("y %= 3 : " + y);

        // 5. Ternary Operator
        int max = (a > b) ? a : b;
        System.out.println("5. Ternary Operator");
        System.out.println("Max of a and b : " + max);

        // 6. Bitwise Operators
        int m = 8;   // 0110 in binary
        int n = 6;   // 0100 in binary
        System.out.println("6. Bitwise Operators");
        System.out.println("m & n : " + (m & n)); // AND
        System.out.println("m | n : " + (m | n)); // OR
        System.out.println("m ^ n : " + (m ^ n)); // XOR
        System.out.println("~m : " + (~m));         // NOT
        System.out.println("m << 1 : " + (m << 1)); // Left shift
        System.out.println("m >> 1 : " + (m >> 1)); // Right shift
    }
}
```

Class compiled - no syntax errors

Figure 1 QN:1 Code Compilation

Screenshot of the output

```
1. Arithmetic Operators
+ : 50
- : 20
* : 525
/ : 2
% : 5
Increment and Decrement Operators
x before increment: 7
x after x++ : 8
x after x-- : 7
2. Relational Operators
a > b : true
a < b : false
a == b : false
a != b : true
a >= b : true
a <= b : false
3. Logical Operators
p && q : false
p || q : true
!p : false
4. Assignment Operators
y = 10
y += 5 : 15
y -= 3 : 12
y *= 2 : 24
y /= 4 : 6
y %= 3 : 0
5. Ternary Operator
Max of a and b : 35
6. Bitwise Operators
m & n : 0
m | n : 14
m ^ n : 14
~m : -9
m << 1 : 16
m >> 1 : 4
```

```
Can only enter input while your program is running
```

Figure 2 QN:1 Output Display

Question 2: GradeEvaluator.java
Create a program that:
• Takes a numeric grade as input
• Uses the ternary operator to assign:
• "Pass" if grade ≥ 40
• "Fail" if grade < 40
Format output using escape sequences

Test Case

Objectives	To compile and execute the program using Blue J.
Action	• Blue J was opened and the project containing the source code file GradeEvaluator.java was loaded. • The program file GradeEvaluator.java was compiled using the Compile button in blue J. • The compiled class GradeEvaluator was executed by right clicking the class and selecting the void main (String[] args) options.
Expected result	The program should compile and ask one number to user and display the result.
Actual result	The program sucessfully compile and ask one number to user and display the result.
Conclusion	Test was successful.

Screenshot of the code compilation

```java
import java.util.Scanner;

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

        System.out.print("Enter your grade: ");
        double grade = sc.nextDouble();

        // Ternary operator: condition ? value_if_true : value_if_false
        String result = (grade >= 40) ? "Pass" : "Fail";

        // Output with escape sequences
        System.out.println("Grade Evaluation Result");
        System.out.println("-----------------------");
        System.out.println("Your grade: " + grade);
        System.out.println("Status: " + result);

        sc.close();
    }
}
```

```
Class compiled - no syntax errors
```

Figure 3 QN:2 Code Compilating

Screenshot of the output

```
Enter your grade: 99
Grade Evaluation Result
-----------------------
Your grade: 99.0
Status: Pass
```

```
Can only enter input while your program is running
```

Figure 4 QN:2 Output Display

Question 3: Data Type Inspector
Create a Java program named DataTypeInspector.java that:
• Declares and initializes a variable for each of Java's 8 primitive data types.
• Uses appropriate literal values for initialization.
• Prints the value of each variable to the console, each with a descriptive label.

Test Case

Objectives	To compile and execute the program using Blue J.
Action	• Blue J was opened and the project containing the source code file DataTypeInspector.java was loaded. • The program file DataTypeInspector.java was compiled using the Compile button in blue J. • The compiled class DataTypeInspector was executed by right clicking the class and selecting the void main(String[] args) options.
Expected result	The program should compile and display the 8 primitive datatypes without any errors.
Actual result	The program sucessfully compile and display the 8 primitive datatypes without any errors.
Conclusion	Test was successful.

Screenshot of the code compilation

```java
public class DataTypeInspector {
    public static void main(String[] args) {
        // 1. Integer types
        byte byteVar = 100;             // 8-bit
        short shortVar = 20000;         // 16-bit
        int intVar = 100000;            // 32-bit
        long longVar = 10000000000L; // 64-bit (note the 'L' suffix)

        // 2. Floating point types
        float floatVar = 3.14f;         // 32-bit (note the 'f' suffix)
        double doubleVar = 3.14159265359; // 64-bit

        // 3. Boolean type
        boolean boolVar = true;

        // 4. Character type
        char charVar = 'A';

        // Printing values with descriptive labels
        System.out.println("Data Type Inspector");
        System.out.println("--------------------");
        System.out.println("byte value      : " + byteVar);
        System.out.println("short value     : " + shortVar);
        System.out.println("int value       : " + intVar);
        System.out.println("long value      : " + longVar);
        System.out.println("float value     : " + floatVar);
        System.out.println("double value    : " + doubleVar);
        System.out.println("boolean value   : " + boolVar);
        System.out.println("char value      : " + charVar);
    }
}
```

```
Class compiled - no syntax errors
```

Figure 5 QN:3 Code Compilating

Screenshot of the output

```
Data Type Inspector
--------------------
byte value      : 100
short value     : 20000
int value       : 100000
long value      : 10000000000
float value     : 3.14
double value    : 3.14159265359
boolean value   : true
char value      : A
```

Figure 6 QN:3 Output Display

Question 4: Default Value Checker

Create a Java class named DefaultValues.java.
• Declare member variables (fields) for all 8 primitive types without initializing them.
• In the main method, create an instance of the class and print the value of each field.
• Add a comment explaining why this wouldn't work for local variables.

Test Case

Objectives	To compile and execute the program using Blue J.
Action	• Blue J was opened and the project containing the source code file DefaultValues.java was loaded. • The program file DefaultValues.java was compiled using the Compile button in blue J. • The compiled class DefaultValues was executed by right clicking the class and selecting the void main(String[] args) options.
Expected result	The program should compile and display all the values as programmed without any errors.
Actual result	The program sucessfully compile and display all the values as programmed without any errors.
Conclusion	Test was successful.

Screenshot of the code compilation

```java
public class DefaultValues {
    // Member variables (fields) for all 8 primitive types
    byte byteVar;         // default: 0
    short shortVar;       // default: 0
    int intVar;           // default: 0
    long longVar;         // default: 0L
    float floatVar;       // default: 0.0f
    double doubleVar;     // default: 0.0d
    boolean boolVar;      // default: false
    char charVar;         // default: '\u0000' (null character)

    public static void main(String[] args) {
        // Create an instance of the class
        DefaultValues obj = new DefaultValues();

        // Print the default values of each field
        System.out.println("Default Values of Primitive Types");
        System.out.println("---------------------------------");
        System.out.println("Value of byte    : " + obj.byteVar);
        System.out.println("Value of short   : " + obj.shortVar);
        System.out.println("Value of int     : " + obj.intVar);
        System.out.println("Value of long    : " + obj.longVar);
        System.out.println("Value of float   : " + obj.floatVar);
        System.out.println("Value of double  : " + obj.doubleVar);
        System.out.println("Value of boolean : " + obj.boolVar);
        System.out.println("Value of char    : [" + obj.charVar + "]");

        /*
         Why this wouldn't work for local variables:
         - Local variables (declared inside methods) do not have default values.
         - The compiler requires that local variables be explicitly initialized
           before they are used; otherwise it is a compile-time error:
           "variable <name> might not have been initialized".

         For Example (you can uncomment to see the error):
         int x;                  // local variable
         System.out.println(x); // ✕ Compile error: x might not have been initialized
        */
    }
}
```

Class compiled - no syntax errors

Figure 7 QN:4 Code Compilating

Screenshot of the output

```
Default Values of Primitive Types
---------------------------------
Value of byte    : 0
Value of short   : 0
Value of int     : 0
Value of long    : 0
Value of float   : 0.0
Value of double  : 0.0
Value of boolean : false
Value of char    : [ ]
```

Can only enter input while your program is running

Figure 8 QN:4 Output Display

Question 5: Literal Practice

Create a program named LiteralPractice.java that demonstrates the use of specific literals:
• A long variable initialized with a value requiring the ’L’ suffix.
• A float variable initialized with a value requiring the ’f’ suffix.
• A char variable initialized using a Unicode escape sequence (e.g., for the copyright symbol ©).
• Print the value of each variable.
Test Case

Objectives	To compile and execute the program using Blue J.
Action	• Blue J was opened and the project containing the source code file LiteralPractice.java was loaded. • The program file LiteralPractice.java was compiled using the Compile button in blue J. • The compiled class LiteralPractice was executed by clicking the class right and selecting the void main(String[] args) options.
Expected result	The program should compile and display the value without any errors.
Actual result	The program sucessfully compile and display the value without any errors.
Conclusion	Test was successful.

Screenshot of the code compilation

```java
public class LiteralPractice {
    public static void main(String[] args) {
        // Long literal (requires 'L' suffix)
        long longVar = 1234567890123L;

        // Float literal (requires 'f' suffix)
        float floatVar = 9.81f;

        // Char literal using Unicode escape sequence (© symbol is \u00A9)
        char charVar = '\u00A9';

        // Printing values
        System.out.println("Literal Practice");
        System.out.println("----------------");
        System.out.println("Long literal value    : " + longVar);
        System.out.println("Float literal value   : " + floatVar);
        System.out.println("Char literal value    : " + charVar);
    }
}
```

```
Class compiled - no syntax errors
```

Figure 9 QN:5 Code Compilating

Screenshot of the output

```
Literal Practice
----------------
Long literal value    : 1234567890123
Float literal value   : 9.81
Char literal value    : ©
```

```
Can only enter input while your program is running
```

Figure 10 QN:5 Output Display

Question no: 6

Context

A local rickshaw service in Biratnagar needs a simple tool to calculate fares for their customers. The fare calculation has a few components: a base fare, a per-kilometer charge, and a per-minute charge. They also offer discounts for locals on long distances and have a surcharge for night-time travel.

Problem

The rickshaw drivers need a program that can:

• Take distance (in km) and time (in minutes) as input.

• Ask if the customer is a local and if the travel is during the night. (Hint: use terinary operator)

• Calculate the total fare based on the rules.

• Display the final fare in a clear, Nepali format (e.g., "Rs. 550").

Test Case

Objectives	To compile and execute the program using Blue J.
Action	• Blue J was opened and the project containing the source code file RickshawFare.java was loaded. • The program file RickshawFare.java was compiled using the Compile button in blue J. • The compiled class RickshawFare was executed by right clicking the class and selecting the void main(String[] args) options.
Expected result	The program should compile and display the total fare without any errors.
Actual result	The program sucessfully compile and display the total fare without any errors.
Conclusion	Test was successful.

Screenshot of the code compilation

```java
import java.util.Scanner;

public class RickshawFare {
    public static void main(String[] args) {
        // 1. Declare constants for fare calculation
        double BASE_FARE = 25.0;         // base fare
        double PER_KM = 25.0;            // per kilometer charge
        double PER_MIN = 8.0;            // per minute charge
        double LOCAL_DISCOUNT = 0.10;    // 10% discount for locals (long distance)
        double NIGHT_SURCHARGE = 0.20;  // 20% surcharge for night travel

        // 2. Use Scanner to get inputs
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter distance (in km): ");
        double distance = sc.nextDouble();

        System.out.print("Enter time (in minutes): ");
        double time = sc.nextDouble();

        System.out.print("Is the customer local? (true/false): ");
        boolean isLocal = sc.nextBoolean();

        System.out.print("Is the travel during night? (true/false): ");
        boolean isNight = sc.nextBoolean();

        // 3. Calculate fare
        double fare = BASE_FARE + (distance * PER_KM) + (time * PER_MIN);

        // 4. Apply discount if applicable (locals on long distance, say ≥ 10 km)
        fare = (isLocal && distance >= 10) ? fare - (fare * LOCAL_DISCOUNT) : fare;

        // 5. Apply night surcharge if applicable
        fare = isNight ? fare + (fare * NIGHT_SURCHARGE) : fare;

        // 6. Display the final fare in Nepali format
        System.out.println("Final Rickshaw Fare");
        System.out.println("--------------------");
        System.out.println("Rs. " + fare);

        sc.close();
    }
}
```

Class compiled - no syntax errors

Figure 11 QN:6 Code Compiling

Screenshot of the output

```
Enter distance (in km): 7
Enter time (in minutes): 30
Is the customer local? (true/false): true
Is the travel during night? (true/false): false
Final Rickshaw Fare
--------------------
Rs. 440.0
```

Can only enter input while your program is running

Figure 12 QN:6 Output Display