

CS4001 – Programming

Primitive Data Types in Java

Week 03: Workshop

Itahari International College

Academic Year 2025/26

Question 1:

Create "MathOperations.java" with all operator types

Test case

Objectives	To compile and execute the program using BlueJ
Action	<ul style="list-style-type: none">BlueJ 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 BlueJ.The complied class MathOperations was executed by right clicking the class and selecting the void main(String[] args) option.
Expected Result	The program should compile and display all operator types as programmed without any errors.
Actual Result	The program successfully compiles, and display operator types as programmed without any errors.
Conclusion	Test was successful

CS4001 – Programming

Screenshot of the code compilation

```
1 public class MathOperations {  
2     public static void main(String[] args) {  
3         int a = 30;  
4         int b = 10;  
5  
6         // 1. Arithmetic Operators  
7         int sum = a + b;  
8         int diff = a - b;  
9         int prod = a * b;  
10        int div = a / b;  
11        int mod = a % b;  
12        System.out.println("1. Arithmetic Operators :-");  
13        System.out.println("+ : " + sum); //Addition  
14        System.out.println("- : " + diff); //Subtraction  
15        System.out.println("* : " + prod); //Multiplication  
16        System.out.println("/ : " + div); //Division  
17        System.out.println("% : " + mod); //Modulus  
18  
19        // Increment and Decrement (Unary Operators)  
20        int x = 5;  
21        System.out.println("Increment and Decrement Operators :-");  
22        System.out.println("x before increment: " + x);  
23  
24        x++;  
25        System.out.println("x after x++ : " + x); // post-increment  
26  
27        x--;  
28        System.out.println("x after x-- : " + x); // post-decrement  
29  
30        ++x;  
31        System.out.println("x after ++x : " + x); // pre-increment  
32  
33        --x;  
34  
35        System.out.println("x after --x : " + x); // pre-decrement  
36  
37        System.out.println(""); //Just to give space  
38  
39        // 2. Relational Operators  
40        System.out.println("2. Relational Operators :-");  
41        System.out.println("a > b : " + (a > b)); // Greaterthan  
42        System.out.println("a < b : " + (a < b)); // Lessthan  
43        System.out.println("a == b : " + (a == b)); //Equal to  
44        System.out.println("a != b : " + (a != b)); //Notequal  
45        System.out.println("a >= b : " + (a >= b)); // Greaterorequal  
46        System.out.println("a <= b : " + (a <= b)); // Lessorequal  
47  
48        System.out.println(""); //Just to give space  
49  
50        // 3. Logical Operators  
51        boolean p = true;  
52        boolean q = false;  
53        System.out.println("3. Logical Operators :-");  
54        System.out.println("p && q : " + (p && q)); //AND  
55        System.out.println("p || q : " + (p || q)); //OR  
56        System.out.println("!p : " + (!p)); //NOT  
57  
58        System.out.println(""); //Just to give space  
59  
60        // 4. Assignment Operators  
61        int y = 20;  
62        System.out.println("4. Assignment Operators :-");  
63        System.out.println("y = " + y);  
64        y += 5;  
65        System.out.println("y += 5 : " + y);  
66        y -= 3;  
67        System.out.println("y -= 3 : " + y);  
68  
69  
70        Class compiled - no syntax errors
```

Figure 1 QN.1 Code Compilation (1)

```
34 System.out.println("x after --x : " + x); // pre-decrement  
35  
36 System.out.println(""); //Just to give space  
37  
38 // 2. Relational Operators  
39 System.out.println("2. Relational Operators :-");  
40 System.out.println("a > b : " + (a > b)); // Greaterthan  
41 System.out.println("a < b : " + (a < b)); // Lessthan  
42 System.out.println("a == b : " + (a == b)); //Equal to  
43 System.out.println("a != b : " + (a != b)); //Notequal  
44 System.out.println("a >= b : " + (a >= b)); // Greaterorequal  
45 System.out.println("a <= b : " + (a <= b)); // Lessorequal  
46  
47 System.out.println(""); //Just to give space  
48  
49 // 3. Logical Operators  
50 boolean p = true;  
51 boolean q = false;  
52 System.out.println("3. Logical Operators :-");  
53 System.out.println("p && q : " + (p && q)); //AND  
54 System.out.println("p || q : " + (p || q)); //OR  
55 System.out.println("!p : " + (!p)); //NOT  
56  
57 System.out.println(""); //Just to give space  
58  
59 // 4. Assignment Operators  
60 int y = 20;  
61 System.out.println("4. Assignment Operators :-");  
62 System.out.println("y = " + y);  
63 y += 5;  
64 System.out.println("y += 5 : " + y);  
65 y -= 3;  
66 System.out.println("y -= 3 : " + y);  
67  
68  
69  
70 Class compiled - no syntax errors
```

Figure 2 QN.1 Code Compilation (2)

CS4001 – Programming

The screenshot shows a Java code editor window titled "MathOperations x". The code is a series of println statements illustrating different operators:

```
65 y -= 3;
66 System.out.println("y -= 3 : " + y);
67 y *= 2;
68 System.out.println("y *= 2 : " + y);
69 y /= 4;
70 System.out.println("y /= 4 : " + y);
71 y %= 3;
72 System.out.println("y %= 3 : " + y);

73 System.out.println(""); //Just to give space
74

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

79 System.out.println(""); //Just to give space
80

81 // 6. Bitwise Operators
82 int m = 6; // 0110 in binary
83 int n = 4; // 0100 in binary
84 System.out.println("6. Bitwise Operators :-");
85 System.out.println("m & n : " + (m & n)); // AND
86 System.out.println("m | n : " + (m | n)); // OR
87 System.out.println("m ^ n : " + (m ^ n)); // XOR
88 System.out.println("~m : " + (~m)); // NOT
89 System.out.println("m << 1 : " + (m << 1)); // Left shift
90 System.out.println("m >> 1 : " + (m >> 1)); // Right shift
91
92 }

93 }

94 }
```

At the bottom left, it says "Class compiled - no syntax errors". At the bottom right, there is a "saved" indicator.

Figure 3 QN.1 Code Compilation (3)

Screenshot of the Output

The screenshot shows the terminal output of the compiled Java program. It displays the results of each operator category:

```
Options
1. Arithmetic Operators :-
+ : 40
- : 20
* : 300
/ : 3
% : 0
Increment and Decrement Operators :-
x before increment: 5
x after x++ : 6
x after ++x : 5
x after --x : 6
x after --x : 5

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 = 20
y += 5 : 25
y -= 3 : 22
y *= 2 : 44
y /= 4 : 11
y %= 3 : 2

5. Ternary Operator :-
Max of a and b : 30

6. Bitwise Operators :-
m & n : 4
m | n : 6
m ^ n : 2
~m : -7
m << 1 : 12
m >> 1 : 3
```

At the bottom, there is a message: "Can only enter input while your program is running".

Figure 4 QN.1 Output Display

CS4001 – Programming

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 BlueJ
Action	<ul style="list-style-type: none">• BlueJ 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 BlueJ.• The complied class GradeEvaluator was executed by right clicking the class and selecting the void main(String[] args) option.
Expected Result	The program should compile and ask the user a grade and display the result (i.e. Pass/Fail) without any errors.
Actual Result	The program Successfully compile and ask user a grade and display the result (i.e. Pass/Fail) without any errors.
Conclusion	Test was successful

CS4001 – Programming

Screenshot of the code compilation

```
GradeEvaluator X
Compile Undo Cut Copy Paste Find... Close Source Code
1 import java.util.Scanner;
2
3 public class GradeEvaluator {
4     public static void main(String[] args) {
5         Scanner sc = new Scanner(System.in);
6
7         System.out.print("Enter your grade: ");
8         double grade = sc.nextDouble();
9
10        // Ternary operator: condition ? value_if_true : value_if_false
11        String result = (grade >= 40) ? "Pass" : "Fail";
12
13        // Output with escape sequences
14        System.out.println("Grade Evaluation Result");
15        System.out.println("-----");
16        System.out.println("Your grade: " + grade);
17        System.out.println("Status: " + result);
18
19        sc.close();
20    }
21}
```

Class compiled - no syntax errors saved

Figure 5 QN.2 Code Compilation

Screenshot of the Output

```
Enter your grade: 50
Grade Evaluation Result
-----
Your grade: 50.0
Status: Pass
```

Can only enter input while your program is running

Figure 6 QN.2 Output Display

CS4001 – Programming

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 BlueJ
Action	<ul style="list-style-type: none">• BlueJ was opened and the project containing the source code file <code>DataTypeInspector.java</code> was loaded.• The program file <code>DataTypeInspector.java</code> was compiled using the Compile button in BlueJ.• The compiled class <code>DataTypeInspector</code> was executed by right clicking the class and selecting the void <code>main(String[] args)</code> option.
Expected Result	The program should compile and display the value of each variable as instructed without any errors.
Actual Result	The program successfully compiles and displays the value of each variable as instructed without any errors.
Conclusion	Test was successful

CS4001 – Programming

Screenshot of the code compilation

The screenshot shows a Java code editor window titled "DataTypeInspector X". The code is a simple application that prints various primitive data types to the console. The code includes comments explaining the data types and their bit widths. The editor has standard toolbar buttons for Compile, Undo, Cut, Copy, Paste, Find..., and Close. A "Source Code" dropdown menu is visible in the top right. Below the code editor is a status bar with the message "Class compiled - no syntax errors" and a "saved" indicator.

```
1 public class DataTypeInspector {  
2     public static void main(String[] args) {  
3         // 1. Integer types  
4         byte byteVar = 100;           // 8-bit  
5         short shortVar = 20000;        // 16-bit  
6         int intVar = 100000;          // 32-bit  
7         long longVar = 1000000000L;    // 64-bit (note the 'L' suffix)  
8  
9         // 2. Floating point types  
10        float floatVar = 3.14f;       // 32-bit (note the 'f' suffix)  
11        double doubleVar = 3.14159265359; // 64-bit  
12  
13        // 3. Boolean type  
14        boolean boolVar = true;  
15  
16        // 4. Character type  
17        char charVar = 'A';  
18  
19         // Printing values with descriptive labels  
20        System.out.println("Data Type Inspector");  
21        System.out.println("-----");  
22        System.out.println("byte value      : " + byteVar);  
23        System.out.println("short value    : " + shortVar);  
24        System.out.println("int value      : " + intVar);  
25        System.out.println("long value     : " + longVar);  
26        System.out.println("float value    : " + floatVar);  
27        System.out.println("double value   : " + doubleVar);  
28        System.out.println("boolean value  : " + boolVar);  
29        System.out.println("char value     : " + charVar);  
30    }  
31 }
```

Class compiled - no syntax errors

saved

Figure 7 QN.3 Code Compilation

Screenshot of the Output

The screenshot shows the output window of a Java application. The title bar says "Data Type Inspector". The window displays a list of primitive data types and their corresponding values. At the bottom, there is a message indicating that input can only be entered while the program is running.

```
byte value      : 100  
short value    : 20000  
int value      : 100000  
long value     : 1000000000  
float value    : 3.14  
double value   : 3.14159265359  
boolean value  : true  
char value     : A
```

Can only enter input while your program is running

Figure 8 QN.3 Output Display

CS4001 – Programming

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 BlueJ
Action	<ul style="list-style-type: none">• BlueJ 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 BlueJ.• The complied class DefaultValues was executed by right clicking the class and selecting the void main(String[] args) option.
Expected Result	The program should compile and display the value as instructed without any errors.
Actual Result	The program successfully compiles and displays the value as instructed without any errors.
Conclusion	Test was successful

CS4001 – Programming

Screenshot of the code compilation

The screenshot shows a Java code editor window titled "DefaultValues X". The code is as follows:

```
1 public class DefaultValues {
2     // Member variables (Fields) for all 8 primitive types
3     byte byteVar;           // default: 0
4     short shortVar;         // default: 0
5     int intVar;             // default: 0
6     long longVar;           // default: 0L
7     float floatVar;         // default: 0.0f
8     double doubleVar;       // default: 0.0d
9     boolean boolVar;        // default: false
10    char charVar;           // default: '\u0000' (null character)
11
12   public static void main(String[] args) {
13       // Create an instance of the class
14       DefaultValues obj = new DefaultValues();
15
16       // Print the default values of each field
17       System.out.println("Default Values of Primitive Types");
18       System.out.println("-----");
19       System.out.println("Value of byte   : " + obj.byteVar);
20       System.out.println("Value of short : " + obj.shortVar);
21       System.out.println("Value of int   : " + obj.intVar);
22       System.out.println("Value of long  : " + obj.longVar);
23       System.out.println("Value of float  : " + obj.floatVar);
24       System.out.println("Value of double : " + obj.doubleVar);
25       System.out.println("Value of boolean: " + obj.boolVar);
26       System.out.println("Value of char   : [" + obj.charVar + "]");
27
28   /*
29    * Why this wouldn't work for local variables:
30    * - Local variables (declared inside methods) do NOT have default values.
31    * - The compiler requires that local variables be explicitly initialized
32    * before they are used; otherwise it's a compile-time error:
33    *   "variable <name> might not have been initialized".
34
35    * For Example (you can uncomment to see the error):
36    * int x;           // local variable
37    * System.out.println(x); // X Compile error: x might not have been initialized
38   */
39
40 }
41 }
```

At the bottom of the code editor, there is a status bar with the message "Class compiled - no syntax errors" and a "saved" indicator.

Figure 9 QN.4 Code Compilation

Screenshot of the Output

The screenshot shows the output of the Java program. The console window displays the following text:

```
Options
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   : []
```

At the bottom of the console window, there is a message: "Can only enter input while your program is running".

Figure 10 QN.4 Output Display

CS4001 – Programming

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 BlueJ
Action	<ul style="list-style-type: none">• BlueJ 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 BlueJ.• The complied class LiteralPractice was executed by right clicking the class and selecting the void main(String[] args) option.
Expected Result	The program should compile and print the value as instructed without any errors.
Actual Result	The program successfully compiles and prints the value as instructed without any errors.
Conclusion	Test was successful

CS4001 – Programming

Screenshot of the code compilation

The screenshot shows a Java code editor window titled "LiteralPractice". The code defines a class "LiteralPractice" with a main method. It demonstrates various literal types: a long literal (1234567890123L), a float literal (9.81f), and a char literal using a Unicode escape sequence ('\u00A9'). The code uses System.out.println to print these literals. A status bar at the bottom indicates "Class compiled - no syntax errors" and "saved".

```
1 public class LiteralPractice {
2     public static void main(String[] args) {
3         // Long literal (requires 'L' suffix)
4         long longVar = 1234567890123L;
5
6         // Float literal (requires 'f' suffix)
7         float floatVar = 9.81f;
8
9         // Char literal using Unicode escape sequence (© symbol is \u00A9)
10        char charVar = '\u00A9';
11
12        // Printing values
13        System.out.println("Literal Practice");
14        System.out.println("-----");
15        System.out.println("Long literal value : " + longVar);
16        System.out.println("Float literal value : " + floatVar);
17        System.out.println("Char literal value : " + charVar);
18    }
19 }
```

Class compiled - no syntax errors | saved

Figure 11 QN.5 Code Compilation

Screenshot of the Output

The screenshot shows the output of the Java program. It displays the printed values from the main method: "Long literal value : 1234567890123", "Float literal value : 9.81", and "Char literal value : ©". Below the output, a message says "Can only enter input while your program is running".

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

Can only enter input while your program is running

Figure 12 QN.5 Output Display

CS4001 – Programming

Scenario Question

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 ternary 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 BlueJ
Action	<ul style="list-style-type: none">• BlueJ 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 BlueJ.• The complied class RickshawFare was executed by right clicking the class and selecting the void main(String[] args) option.
Expected Result	The program should compile and display the Final Rickshaw fare without any errors.
Actual Result	The program successfully compiles and displays the Final Rickshaw fare without any errors.
Conclusion	Test was successful

CS4001 – Programming

Screenshot of the code compilation

```
1 import java.util.Scanner;
2
3 public class RickshawFare {
4     public static void main(String[] args) {
5         // 1. Declare constants for fare calculation
6         double BASE_FARE = 25.0;           // base fare
7         double PER_KM = 15.0;             // per kilometer charge
8         double PER_MIN = 2.0;             // per minute charge
9         double LOCAL_DISCOUNT = 0.10;    // 10% discount for locals (long distance)
10        double NIGHT_SURCHARGE = 0.20;   // 20% surcharge for night travel
11
12        // 2. Use Scanner to get inputs
13        Scanner sc = new Scanner(System.in);
14
15        System.out.print("Enter distance (in km): ");
16        double distance = sc.nextDouble();
17
18        System.out.print("Enter time (in minutes): ");
19        double time = sc.nextDouble();
20
21        System.out.print("Is the customer local? (true/false): ");
22        boolean islocal = sc.nextBoolean();
23
24        System.out.print("Is the travel during night? (true/false): ");
25        boolean isnight = sc.nextBoolean();
26
27        // 3. Calculate fare
28        double fare = BASE_FARE + (distance * PER_KM) + (time * PER_MIN);
29
30        // 4. Apply discount if applicable (locals on long distance, say ≥ 10 km)
31        fare = (islocal && distance >= 10) ? fare - (fare * LOCAL_DISCOUNT) : fare;
32
33        // 5. Apply night surcharge if applicable
34        fare = isnight ? fare + (fare * NIGHT_SURCHARGE) : fare;
35
36        // 6. Display the final fare in Nepali format
37        System.out.println("Final Rickshaw Fare");
38        System.out.println("-----");
39        System.out.println("Rs. " + fare);
40
41        sc.close();
42    }
43}
```

Class compiled - no syntax errors

Figure 13 QN.6 Code Compilation

Screenshot of the Output

```
Enter distance (in km): 10
Enter time (in minutes): 30
Is the customer local? (true/false): true
Is the travel during night? (true/false): false
Final Rickshaw Fare
-----
Rs. 211.5
```

Can only enter input while your program is running

Figure 14 QN.6 Output Display