



**Module Code & Module Title**

**CS4001NT Programming**

**Assessment Weightage & Type**

**Weekly Assignment**

**Year & Semester**

**Autumn 2025**

**Student Name:** |Sujan kharel

**London Met ID:** 24046104

**College ID:** np05cp4a240412

**Assignment Due Date:** Saturday, November 15, 2025

**Assignment Submission Date:** Saturday, November 22, 2025

*I confirm that I understand my coursework needs to be submitted online via MySecondTeacher under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a marks of zero will be awarded.*

## **1. Introduction**

### **1.1 Overview of Workshop**

The CS4001 Workshop Week 03 is about introducing the students to primitive data types, operators and basics of Java programming. The workshop is also useful in providing the students with hand-on experience in writing programs with these concepts. The workshop also exposes the students to version control via GitHub where they learn to commit Java files to a repository to ensure that they manage their projects appropriately.

## **1.2 Objectives**

- To study and apply Java primitive data types.
- The ability to use various operators such as arithmetic, relational, logical, unary, assignment as well as ternary.
- To generate mini code Java programs by applying real world scenarios.
- To learn about default values of class fields in Java.
- To familiarize myself with literal values in Java.
- To become familiar with GitHub to use it as a version control system by uploading workshop files.

## Question 1:

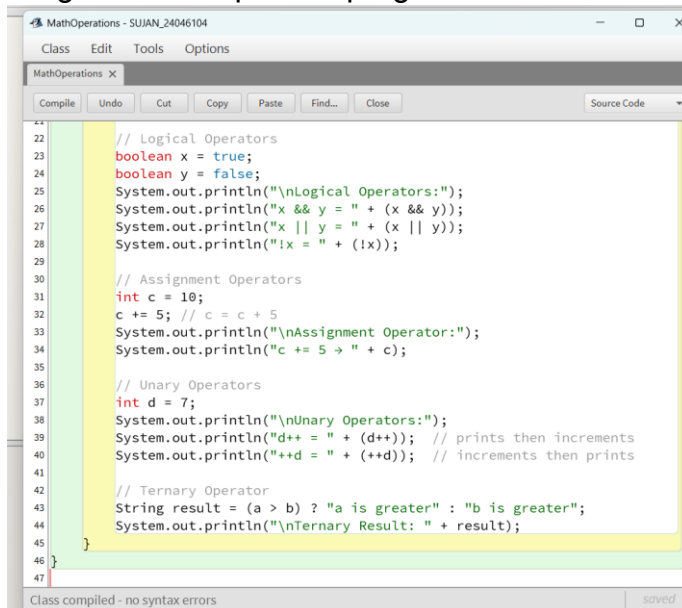
Create "MathOperations.java" with all operator types

### 2.1 Problem Description

Task instructions: to write a Java program that illustrates each of the major categories of operators: arithmetic, relational, logical, unary, assignment and ternary. This is to get to know the behavior of every operator in Java.

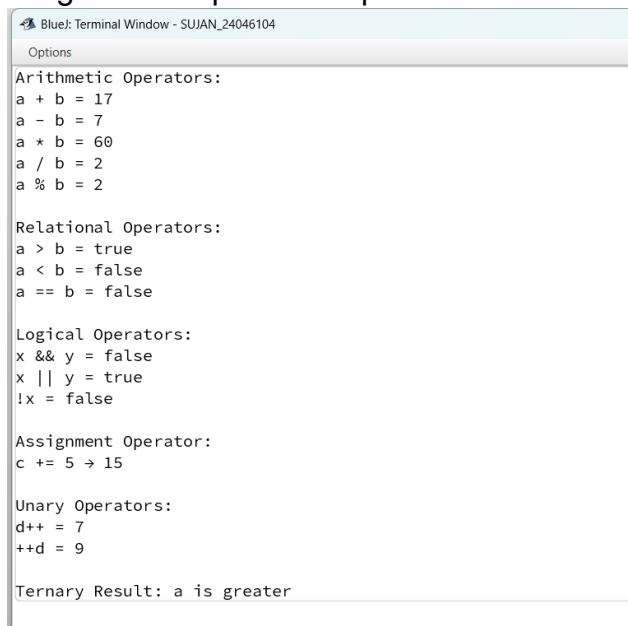
### 2.2 java code

- Program 1: command the prompt
- Program 2: compile the program



```
22 // Logical Operators
23 boolean x = true;
24 boolean y = false;
25 System.out.println("\nLogical Operators:");
26 System.out.println("x && y = " + (x && y));
27 System.out.println("x || y = " + (x || y));
28 System.out.println("!x = " + (!x));
29
30 // Assignment Operators
31 int c = 10;
32 c += 5; // c = c + 5
33 System.out.println("\nAssignment Operator:");
34 System.out.println("c += 5 → " + c);
35
36 // Unary Operators
37 int d = 7;
38 System.out.println("\nUnary Operators:");
39 System.out.println("d++ = " + (d++)); // prints then increments
40 System.out.println("++d = " + (++d)); // increments then prints
41
42 // Ternary Operator
43 String result = (a > b) ? "a is greater" : "b is greater";
44 System.out.println("\nTernary Result: " + result);
45
46 }
47
```

- Program 3: Expected output



```
Blue: Terminal Window - SUJAN_24046104
Options

Arithmetic Operators:
a + b = 17
a - b = 7
a * b = 60
a / b = 2
a % b = 2

Relational Operators:
a > b = true
a < b = false
a == b = false

Logical Operators:
x && y = false
x || y = true
!x = false

Assignment Operator:
c += 5 → 15

Unary Operators:
d++ = 7
++d = 9

Ternary Result: a is greater
```

### 2.3 Test case

Test case no	Operator type	Expression	Expected output	
1	Addition	$a + b = 12 + 5$	17	
2	Subtraction	$a - b = 12 - 5$	7	
3	Multiplication	$a * b = 12 \times 5$	60	
4	Division	$a / b = 12 \div 5$	2	
5	modulus	$a \% b = 12 \% 5$	2	
6	Greater than	$a > b \rightarrow 12 > 5$	true	
7	Less than	$a < b \rightarrow 12 < 5$	false	
8	Equal to	$a == b \rightarrow 12 == 5$	false	

<b>9</b>	Logical And	x && y → true && false	false	
<b>10</b>	Logical or	x		
<b>11</b>	Logical not	!x→!True	false	
<b>12</b>	Assignment	c += 5 → 10 + 5	15	
<b>13</b>	Unary post increment	d++ (prints old value: 7)	7	
<b>14</b>	Unary pre- increment	++d (after post- increment d=8 → ++d=9)	9	
<b>15</b>	Ternary	(a > b)? "a is greater": "b is greater"	A is greater	

## Question 2: GradeEvaluator.java

Create a program that:

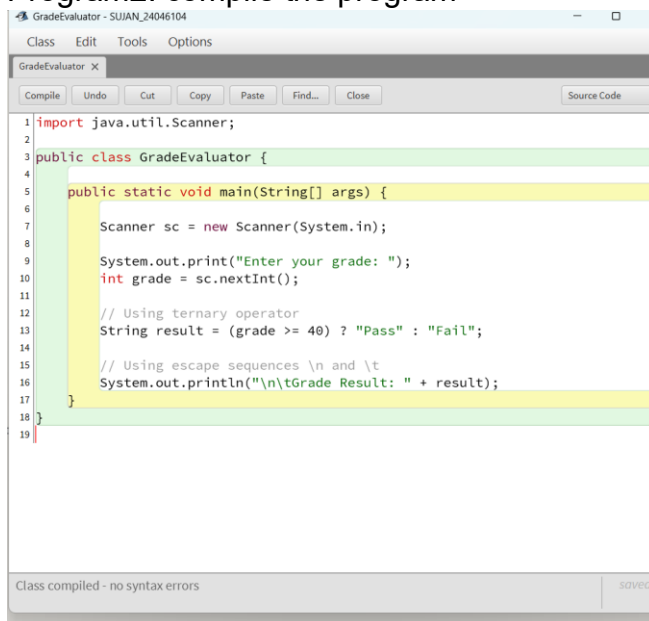
- Takes a numeric grade as input
- Uses the ternary operator to assign:
- "Pass" if grade  $\geq 40$
- "Fail" if grade  $< 40$

### 3.1 Problem Description

Create a program which receives a numeric mark and shows the student Pass or Fail with the help of ternary operator. Presence of escape sequences in the output is required.

### 3.2 java code

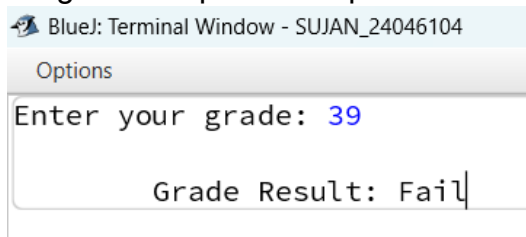
- Program1: command the prompt
- Program2: compile the program



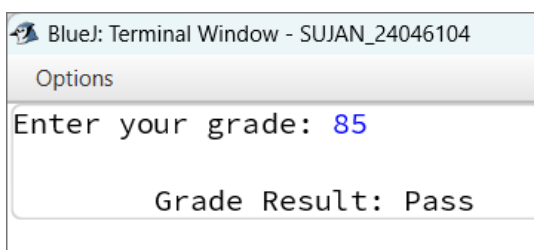
```
1 import java.util.Scanner;
2
3 public class GradeEvaluator {
4
5     public static void main(String[] args) {
6
7         Scanner sc = new Scanner(System.in);
8
9         System.out.print("Enter your grade: ");
10        int grade = sc.nextInt();
11
12        // Using ternary operator
13        String result = (grade >= 40) ? "Pass" : "Fail";
14
15        // Using escape sequences \n and \t
16        System.out.println("\n\tGrade Result: " + result);
17    }
18 }
19
```

Class compiled - no syntax errors

### Program3: expected output



```
BlueJ: Terminal Window - SUJAN_24046104
Options
Enter your grade: 39
Grade Result: Fail
```



```
BlueJ: Terminal Window - SUJAN_24046104
Options
Enter your grade: 85
Grade Result: Pass
```

### 3.2 Test case0

Test case no	input	Condition evaluated	Expected output
1	85	$85 \geq 40 \rightarrow \text{Pass}$	pass
2	40	$40 \geq 40 \rightarrow \text{Pass}$	pass
3	39	$39 < 40 \rightarrow \text{Fail}$	Fail



### 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.

#### 4.1 Problem Description

Java has eight primitive types of data. These are the simplest types of data which are constructed into the language. The primitives are fixed in size and have a certain type of value contained in them.

This program:

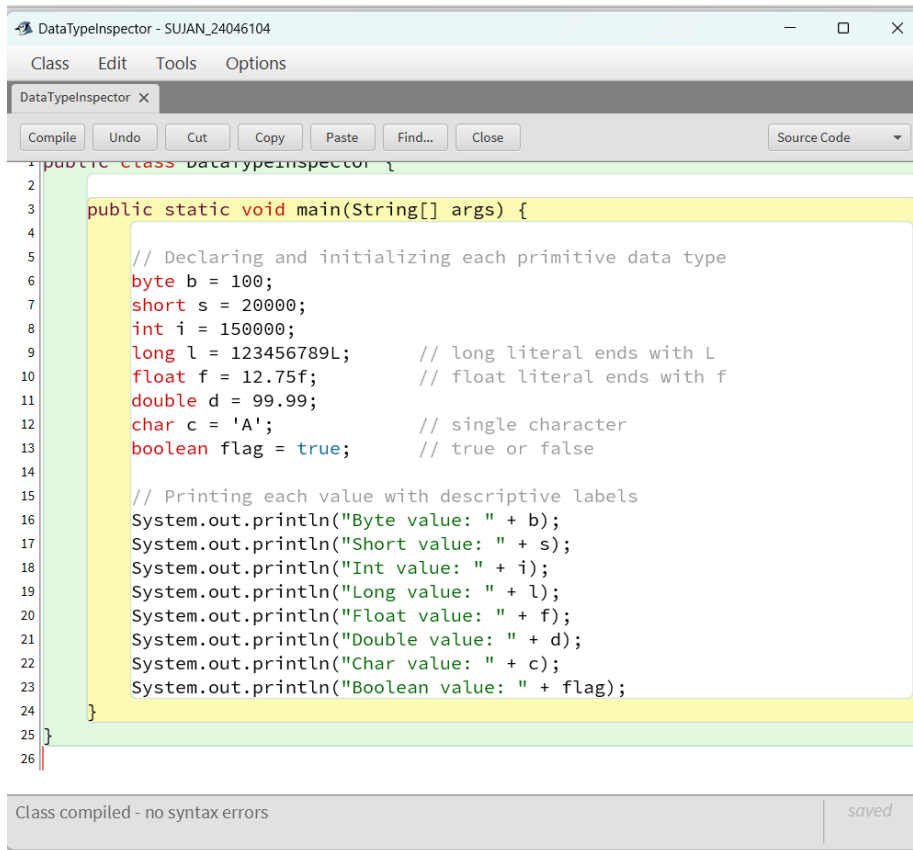
- Declares an individual variable to each primitive data type.
- Creates them with proper literal values.
- Print out each value on the console with a description.

Its eight primitive data type are as follows:

- Byte
- Short
- Long
- Float
- Int
- Double
- Char
- Boolean

#### 4.2 Java code

- Program1: command the prompt
- Program2: compile the program

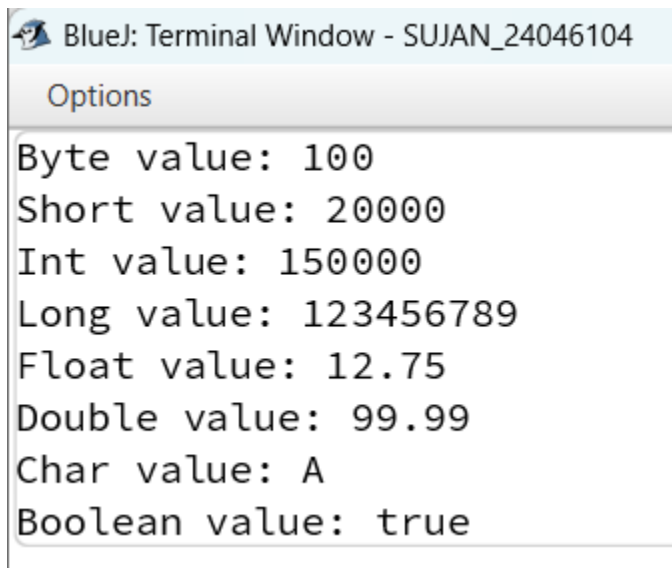


The screenshot shows a window titled "DataInspector - SUJAN\_24046104". The menu bar includes "Class", "Edit", "Tools", and "Options". Below the menu bar is a toolbar with buttons for "Compile", "Undo", "Cut", "Copy", "Paste", "Find...", and "Close". A "Source Code" dropdown menu is also present. The main text area contains the following Java code:

```
1 public class DataInspector {  
2  
3     public static void main(String[] args) {  
4  
5         // Declaring and initializing each primitive data type  
6         byte b = 100;  
7         short s = 20000;  
8         int i = 150000;  
9         long l = 123456789L;    // long literal ends with L  
10        float f = 12.75f;        // float literal ends with f  
11        double d = 99.99;  
12        char c = 'A';            // single character  
13        boolean flag = true;     // true or false  
14  
15        // Printing each value with descriptive labels  
16        System.out.println("Byte value: " + b);  
17        System.out.println("Short value: " + s);  
18        System.out.println("Int value: " + i);  
19        System.out.println("Long value: " + l);  
20        System.out.println("Float value: " + f);  
21        System.out.println("Double value: " + d);  
22        System.out.println("Char value: " + c);  
23        System.out.println("Boolean value: " + flag);  
24    }  
25 }  
26
```

At the bottom of the window, a status bar indicates "Class compiled - no syntax errors" and a "saved" indicator.

- Expected output



The screenshot shows a terminal window titled "BlueJ: Terminal Window - SUJAN\_24046104". The terminal has an "Options" menu at the top. The output of the program is displayed as follows:

```
Byte value: 100  
Short value: 20000  
Int value: 150000  
Long value: 123456789  
Float value: 12.75  
Double value: 99.99  
Char value: A  
Boolean value: true
```

### 4.3 Test case

Test case no	Data type	Input	Expected output
1	byte	100	Byte value: 100
2	short	20000	Short value: 20000
3	Int	150000	Int value: 150000
4	long	123456789L	Long value: 123456789
5	float	12.75f	Float value: 12.75
6	double	99.99	Double value: 99.99
7	char	'A'	Char value: A
8	Boolean	true	Boolean value: true

## Question 4: Default Value Checker

Create a Java class named `DefaultValues.java`.

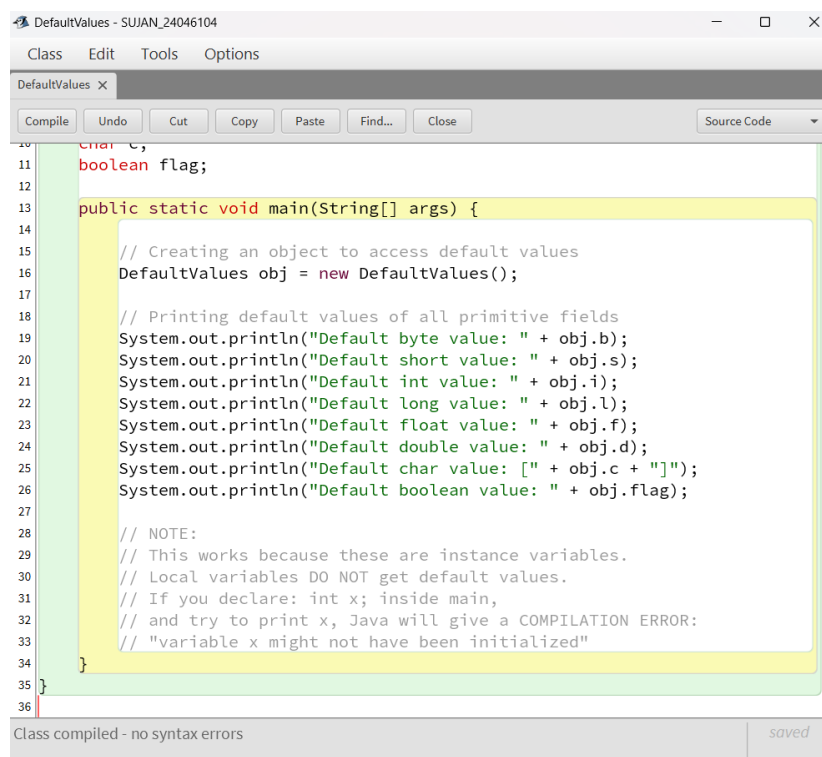
- Declare member variables (fields) for all 8 primitive types without initializing them.
- In the main method, create an instance of the class and print the value of each field.
- Add a comment explaining why this wouldn't work for local variables.

### 5.1 Problem description

This assignment is to develop a Java application entitled `DefaultValues.java` which illustrates the default values that are automatically given to instance variables (fields) of primitive data types in Java.

### 5.2 Java code

- Program1: command the prompt
- Compile the program



```
10  char c;  
11  boolean flag;  
12  
13  public static void main(String[] args) {  
14  
15      // Creating an object to access default values  
16      DefaultValues obj = new DefaultValues();  
17  
18      // Printing default values of all primitive fields  
19      System.out.println("Default byte value: " + obj.b);  
20      System.out.println("Default short value: " + obj.s);  
21      System.out.println("Default int value: " + obj.i);  
22      System.out.println("Default long value: " + obj.l);  
23      System.out.println("Default float value: " + obj.f);  
24      System.out.println("Default double value: " + obj.d);  
25      System.out.println("Default char value: [" + obj.c + "]");  
26      System.out.println("Default boolean value: " + obj.flag);  
27  
28      // NOTE:  
29      // This works because these are instance variables.  
30      // Local variables DO NOT get default values.  
31      // If you declare: int x; inside main,  
32      // and try to print x, Java will give a COMPILATION ERROR:  
33      // "variable x might not have been initialized"  
34  }  
35  }  
36
```

Class compiled - no syntax errors

- Programm3: Expected results

```
BlueJ: Terminal Window - SUJAN_24046104

Options

Default byte value: 0
Default short value: 0
Default int value: 0
Default long value: 0
Default float value: 0.0
Default double value: 0.0
Default char value: [\]
Default boolean value: false
```

### 5.3 Test case

Test case no	Data type	Variable	Expected output
1	byte	b	Default byte value: 0
2	short	s	Default short value: 0
3	int	i	Default int value: 0
4	long	l	Default long value: 0
5	float	f	Default float value: 0.0
6	double	d	Default double value: 0.0
7	char	c	Default char value: [] (blank)
8	Boolean	flag	Default Boolean value: false

### 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.

### 6.1 Problem Description

The task is to create a Java program named `LiteralPractice.java` that demonstrates the use of *literal values* in Java. In this question, you must:

- Declare a long variable and initialize it with a value that requires the L suffix because long values may exceed the int range.
- Declare a float variable and initialize it using the f suffix required for float literals
- Declare a char variable and initialize it using a Unicode escape sequence, such as the copyright symbol © (`\u00A9`).
- Print the value of each variable with a descriptive label.

This exercise helps you understand how Java handles specific literal formats and why certain suffixes or escape sequences are required.

### 6.2 Java code

- Program1: command the prompt
- Program2: compile the program

```
1 public class LiteralPractice {
2
3     public static void main(String[] args) {
4
5         // Long literal (must use 'L' suffix)
6         long population = 9876543210L;
7
8         // Float literal (must use 'f' suffix)
9         float price = 199.99f;
10
11        // Unicode literal for copyright symbol ©
12        char copyrightSymbol = '\u00A9';
13
14        // Printing the values
15        System.out.println("Long literal value: " + population);
16        System.out.println("Float literal value: " + price);
17        System.out.println("Unicode character (©): " + copyrightSymbol);
18    }
19 }
20
```

Class compiled - no syntax errors | saved

- Program3: expected result

```
BlueJ: Terminal Window - SUJAN_24046104
Options
Long literal value: 9876543210
Float literal value: 199.99
Unicode character (©): ©
```

## 6.2 Test case

Test case no	Data type	Literal used	Expected output
1	long	9876543210L	long literal value: 9876543210
2	float	199.99f	Float literal value: 199.99
3	char	'\u00A9'	Unicode character (©): ©

## **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").

Normal Questions Scenario Questions GitHub Task: Version Control Deliverables

Scenario Question (cont.)

## **Solution**

The RickshawFare.java program will be a console-based Java application. It will prompt the user for the necessary inputs and then calculate and display the total fare. This will ensure consistent and transparent pricing for all customers.

## **Execution**

- The program is executed from the command line (e.g., java RickshawFare).
- The program will interactively ask for inputs.
- The output will be the final calculated fare, displayed in the console.



## 7.1 Problem Description

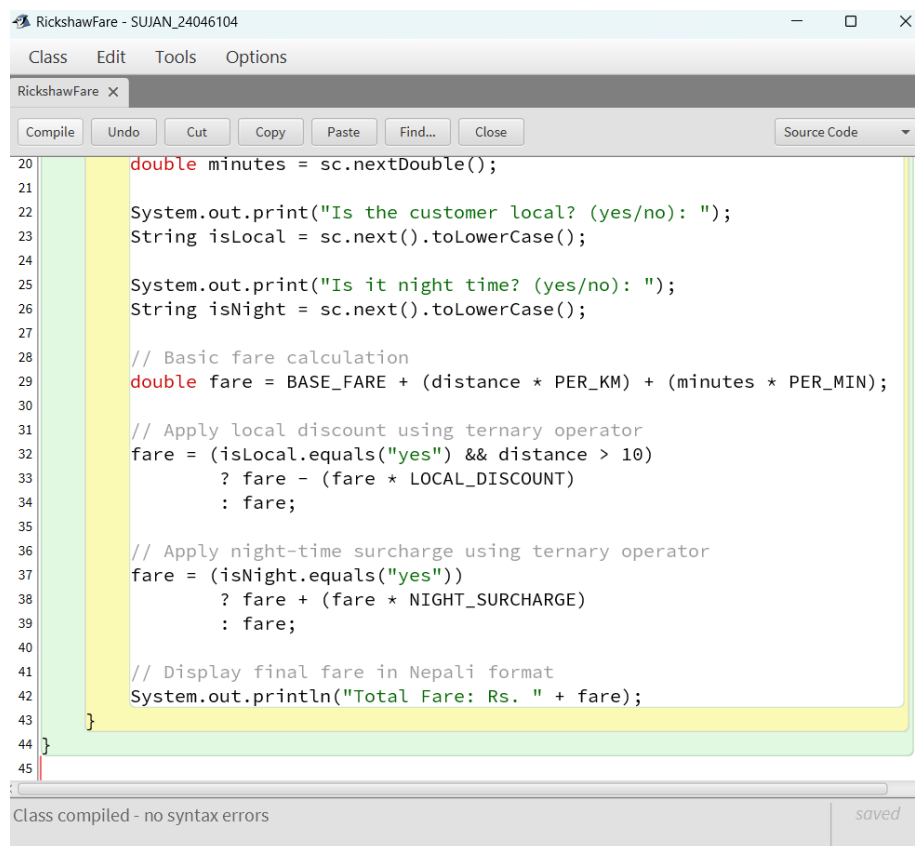
A local rickshaw service in Biratnagar wants a simple software tool to calculate fares for their passengers. Currently, drivers calculate fares manually, which may lead to mistakes and inconsistencies. To ensure fair, quick, and accurate pricing, they want a small Java program that automatically calculates the fare based on distance, time, and additional conditions like local discounts and night-time charges.

The fare calculation includes:

- A base fare
- A per-kilometer charge
- A per-minute charge
- A local discount for customers travelling long distances
- A night-time surcharge for trips during night hours

## 7.2 Java code

- Program1: command the prompt
- Program2: compile the program



```
RickshawFare - SUJAN_24046104
Class Edit Tools Options
RickshawFare X
Compile Undo Cut Copy Paste Find... Close Source Code
20 double minutes = sc.nextDouble();
21
22 System.out.print("Is the customer local? (yes/no): ");
23 String isLocal = sc.next().toLowerCase();
24
25 System.out.print("Is it night time? (yes/no): ");
26 String isNight = sc.next().toLowerCase();
27
28 // Basic fare calculation
29 double fare = BASE_FARE + (distance * PER_KM) + (minutes * PER_MIN);
30
31 // Apply local discount using ternary operator
32 fare = (isLocal.equals("yes") && distance > 10)
33       ? fare - (fare * LOCAL_DISCOUNT)
34       : fare;
35
36 // Apply night-time surcharge using ternary operator
37 fare = (isNight.equals("yes"))
38       ? fare + (fare * NIGHT_SURCHARGE)
39       : fare;
40
41 // Display final fare in Nepali format
42 System.out.println("Total Fare: Rs. " + fare);
43 }
44 }
45
Class compiled - no syntax errors saved
```

- Program3: Expected results

```

BlueJ: Terminal Window - SUJAN_24046104
Options
Long literal value: 9876543210
Float literal value: 199.99
Unicode character (©): ©
Enter distance travelled (in km): 5
Enter time taken (in minutes): 10
Is the customer local? (yes/no): no
Is it night time? (yes/no): no
Total Fare: Rs. 145.0

```

### 7.3 Test case

Test case no	Distance(km)	Time(min)	Local	Night	Expected Fare	Notes
1	5	10	no	no	Rs145	No discount, no surcharge
2	12	15	yes	no	Rs369	Local + long-distance discount
3	8	10	no	yes	Rs174	Night surcharge only
4	15	20	yes	yes	Rs540	Both discount and night surcharge
5	3	5	no	no	Rs85	Only base + per km/mi