LONDON METROPOLITAN UNIVERSITY

ITAHARI INTERNATIONAL COLLEGE

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.

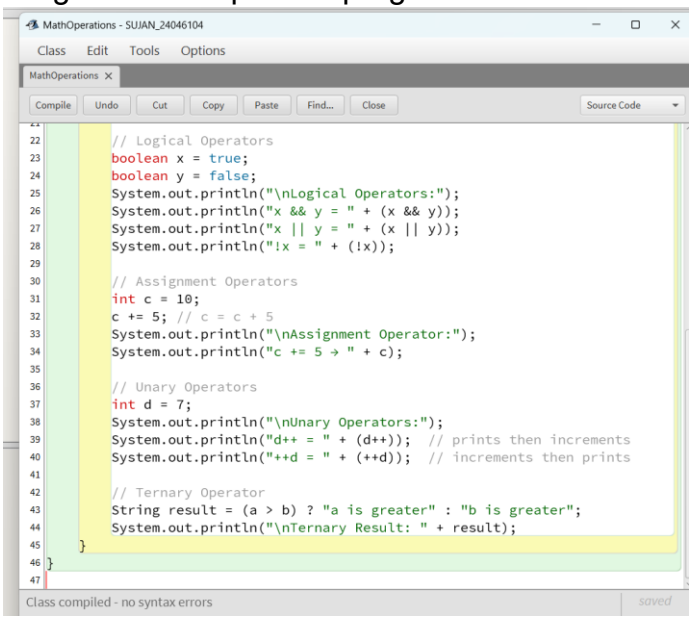
Question 1:
Create "MathOperations.java" with all operator types

2.1 Problem Description

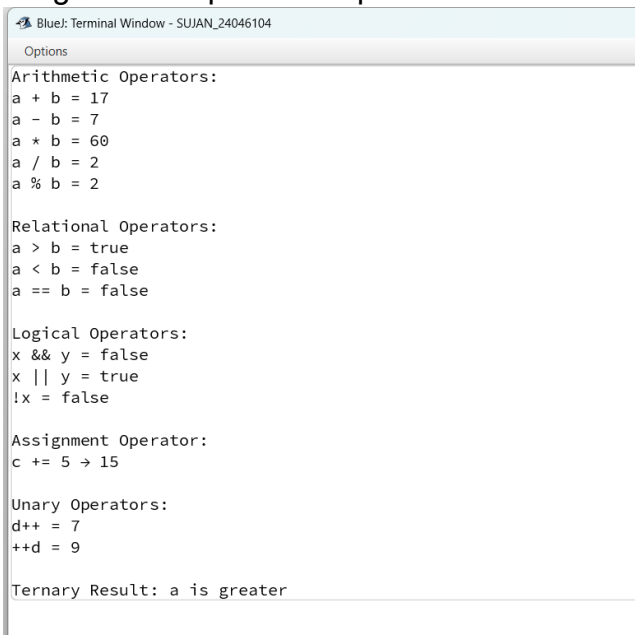
Task instructions: to write a Java program that illustrates each of the major categories of operators: arithmetic, relational, logical, unary, assignment and ternary. This is to get to know the behavior of every operator in Java.

2.2 java code

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



- Program 3: Expected output



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 × 5	60	
4	Division	a / b = 12 ÷ 5	2	
5	modulus	a % b = 12 % 5	2	
6	Greater than	a > b → 12 > 5	true	
7	Less than	a < b → 12 < 5	false	
8	Equal to	a == b → 12 == 5	false	

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

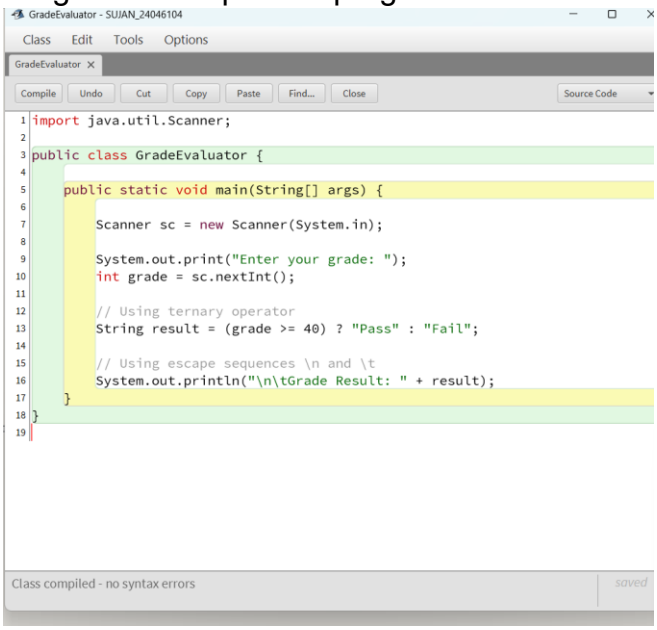
Question 2: GradeEvaluator.java
Create a program that:
• Takes a numeric grade as input
• Uses the ternary operator to assign:
• "Pass" if grade ≥ 40
• "Fail" if grade < 40

3.1 Problem Description

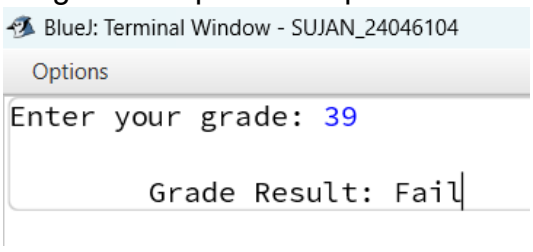
Create a program which receives a numeric mark and shows the student Pass or Fail with the help of ternary operator. Presence of escape sequences in the output is required.

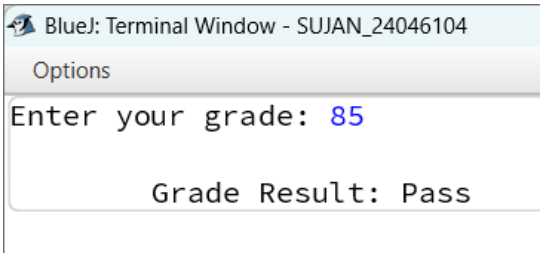
3.2 java code

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



Program3: expected output





3.2 Test case0

Test case no	input	Condition evaluated	Expected output
1	85	85 ≥ 40 → Pass	pass
2	40	40 ≥ 40 → Pass	pass
3	39	39 < 40 → 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:

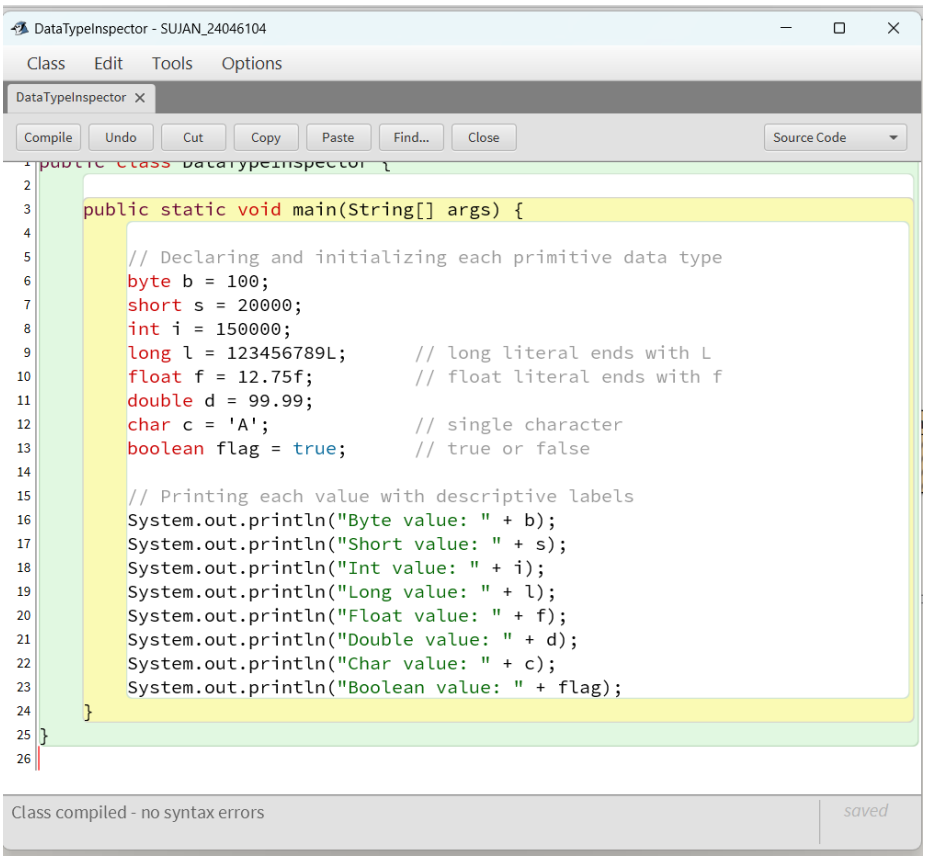
- Declares an individual variable to each primitive data type.
- Creates them with proper literal values.
- Print out each value on the console with a description.

  Its eight primitive data type are as follows:

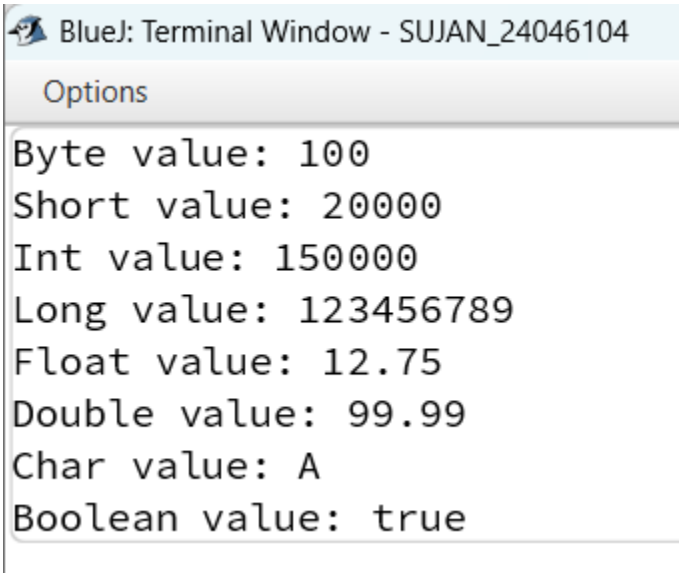
- Byte
- Short
- Long
- Float
- Int
- Double
- Char
- Boolean

4.2 Java code

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



- Expected output



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	B**oolean**	**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.

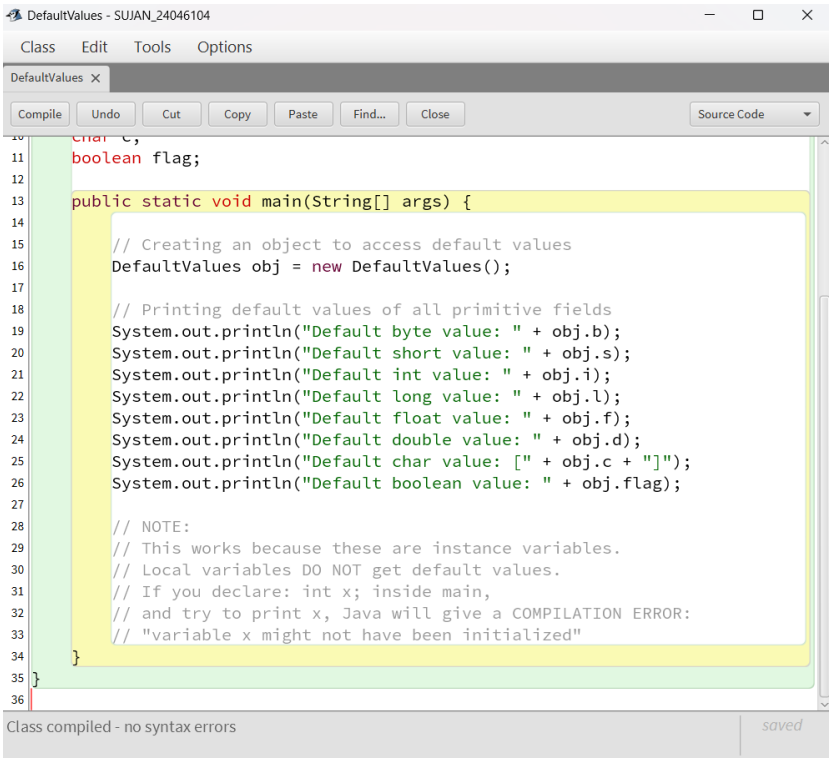
• Add a comment explaining why this wouldn't work for local variables.

5.1 Problem description

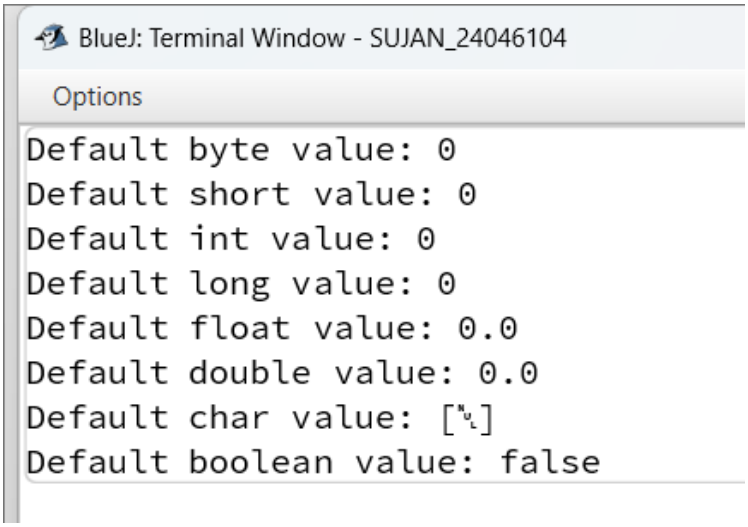
This assignment is to develop a Java application entitled DefaultValues.java which illustrates the default values that are automatically given to instance variables (fields) of primitive data types in Java.

5.2 Java code

- Program1: command the prompt
- Compile the program



- Programm3: Expected results



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

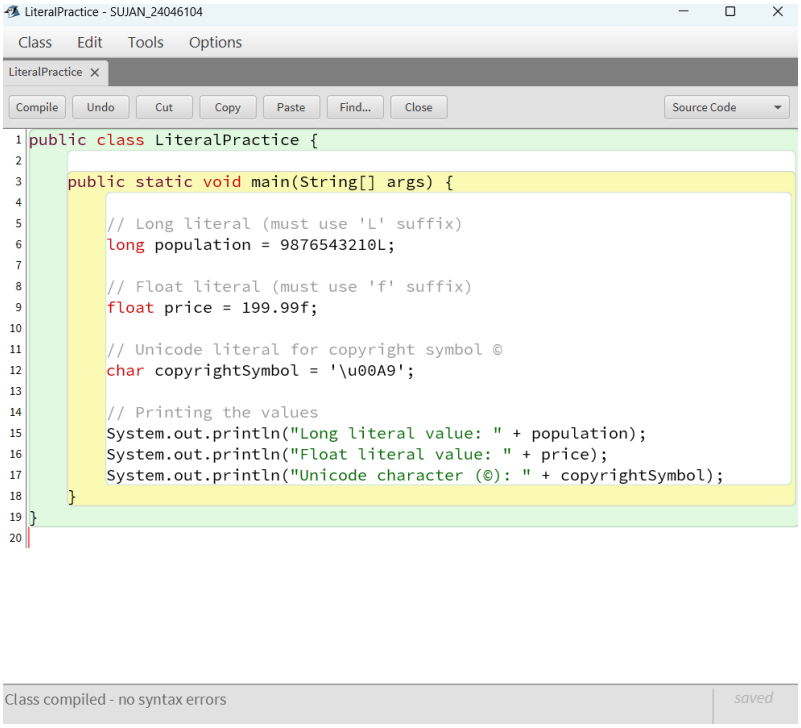
The task is to create a Java program named LiteralPractice.java that demonstrates the use of *literal values* in Java. In this question, you must:

- 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.

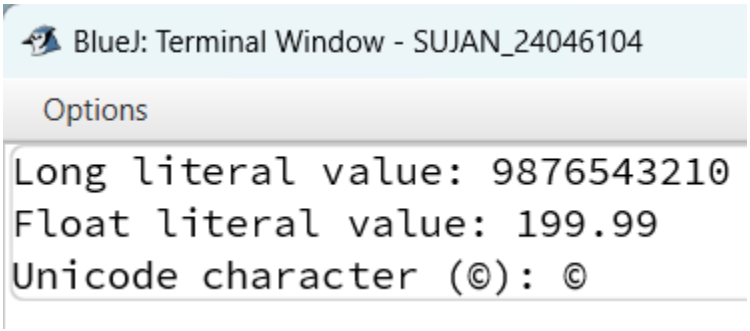
  This exercise helps you understand how Java handles specific literal formats and why certain suffixes or escape sequences are required.

6.2 Java code

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



- Program3: expected result



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 terinary 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

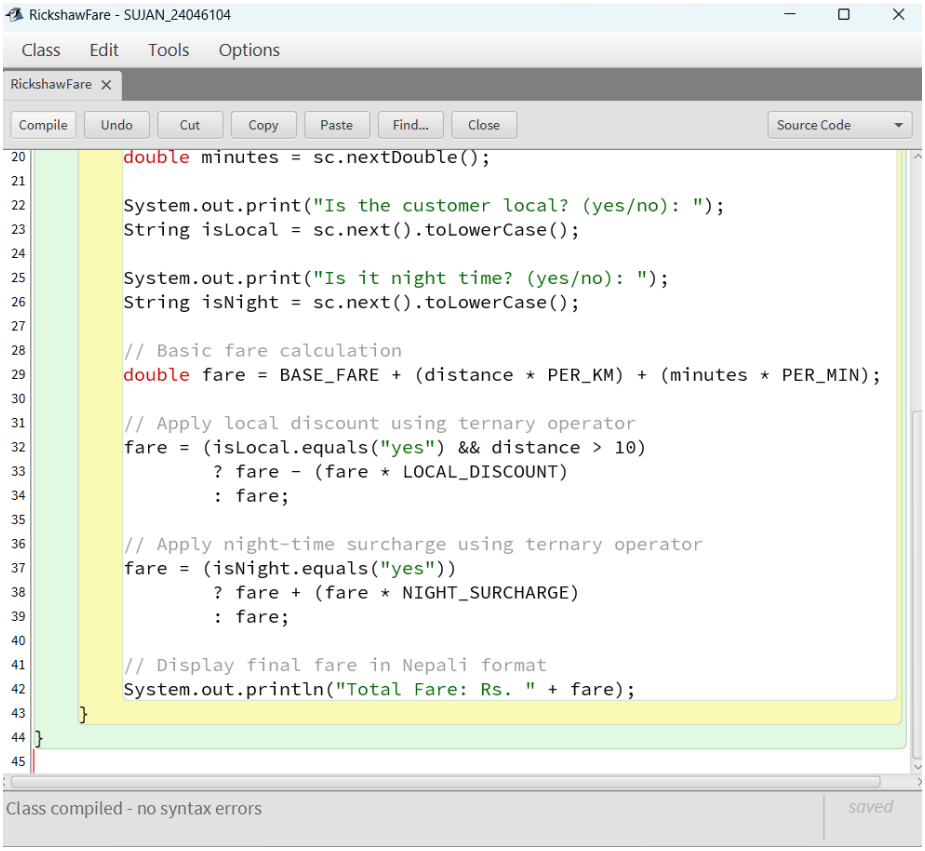
A local rickshaw service in Biratnagar wants a simple software tool to calculate fares for their passengers. Currently, drivers calculate fares manually, which may lead to mistakes and inconsistencies. To ensure fair, quick, and accurate pricing, they want a small Java program that automatically calculates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

The fare calculation includes:

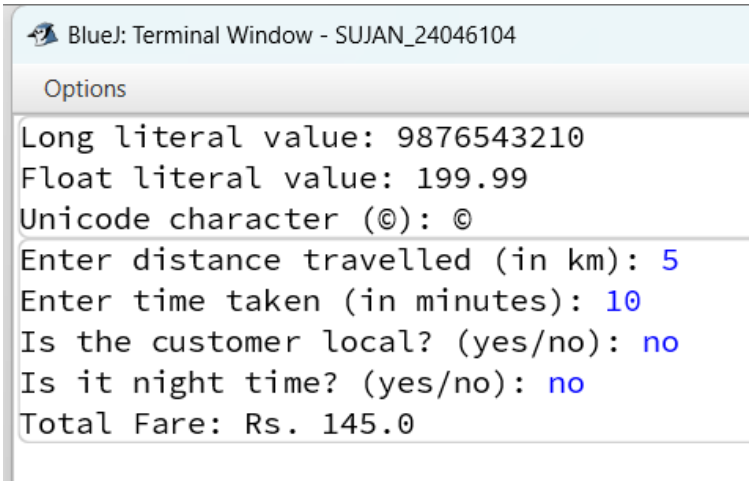
- A base fare
- A per-kilometer charge
- A per-minute charge
- A local discount for customers travelling long distances
- A night-time surcharge for trips during night hours

7.2 Java code

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



- Program3: Expected results



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