

CS4001 – Programming

Syntax and Semantics of Java Programming Language

Week 03: Workshop

Question no: 1

Create "MathOperations.java" with all operator types

Test case

Objective	To compile and execute the program using BlueJ
Action	<ol style="list-style-type: none">1. BlueJ was opened and the project containing the source code the MathOperations.java was loaded.2. The program file MathOperation.java was compiled using the Compile button in BlueJ.3. The compiled 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 the MathOperations without any errors.
Actual result	The program should compile and display all Operators.
Conclusion	Test was successful.

Screenshot of the code compilation

```

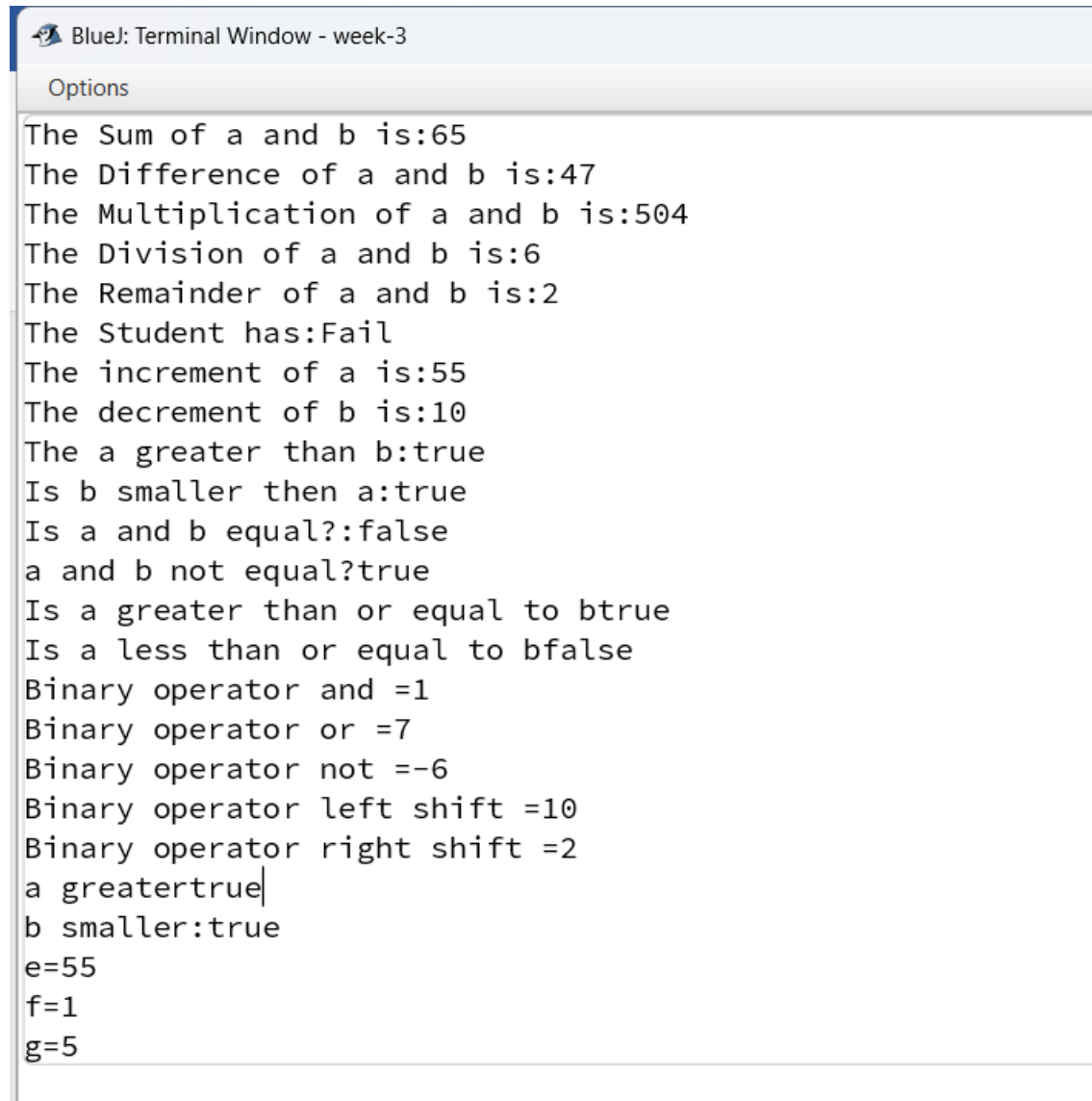
1 public class MathOperations{
2     public static void main(String[] args){
3         int a = 50; //assignment operator
4         int b = 20; //assignment operator
5         int e,f,g; //assignment operator
6         a += 5; //assignment operator
7         b -= 10; //assignment operator
8         int sum = a+b; //Arithmetic operator
9         int c = a++; //Special arithmetic operator
10        int d = b--; //Special arithmetic operator
11        e = c; //assignment operator
12        f = d; //assignment operator
13        g = sum; //assignment operator
14        e *= 1; //assignment operator
15        f /= 10; //assignment operator
16        g %= 10; //assignment operator
17        boolean greater = (a>b); //relation operator
18        boolean smaller = (b<a); //relation operator
19        boolean equal = (a==b); //relation operator
20        boolean not_equal = (a!=b); //relation operator
21        boolean greater_then_or_equal = (a>=b); //relation operator
22        boolean less_then_or_equal = (a<=b); //relation operator
23        boolean n = (a>50 && a>b); //logical operator
24        boolean m = (b<a || b<20); //logical operator
25        String grade = (sum>=100)? "Pass":"Fail"; //Ternary operator
26        int and = 5 & 3; //binary operator
27        int or = 5|3; //binary operator
28        int xor = 5^3; //binary operator
29        int not = ~5; //binary operator
30        int left_shift = 5<<1; //binary operator
31        int right_shift = 5>>1; //binary operator
32
33        System.out.println("The Sum of a and b is:"+sum); //Arithmetic operator
34        System.out.println("The Difference of a and b is:"+a-b); //Arithmetic operator
35        System.out.println("The Multiplication of a and b is:"+a*b); //Arithmetic operator
36        System.out.println("The Division of a and b is:"+a/b); //Arithmetic operator
37        System.out.println("The Remainder of a and b is:"+a%b); //Arithmetic operator
38        System.out.println("The Student has:"+grade); //Ternary operator
39        System.out.println("The increment of a is:"+c); //Arithmetic operator
40        System.out.println("The decrement of b is:"+d); //Arithmetic operator
41        System.out.println("The a greater than b:"+greater); //relation operator
42        System.out.println("Is b smaller then a:"+smaller); //relation operator
43        System.out.println("Is a and b equal?:"+equal); //relation operator
44        System.out.println("a and b not equal?"+"not_equal"); //relation operator
45        System.out.println("Is a greater than or equal to b"+greater_then_or_equal); //relation operator
46        System.out.println("Is a less than or equal to b"+less_then_or_equal); //relation operator
47        System.out.println("Binary operator and ="+"and"); //Binary operator
48        System.out.println("Binary operator or ="+"or"); //Binary operator
49        System.out.println("Binary operator not ="+"not"); //Binary operator
50        System.out.println("Binary operator left shift ="+"left_shift"); //Binary operator
51        System.out.println("Binary operator right shift ="+"right_shift"); //Binary operator
52        System.out.println("a greater"+n); //logical operator
53        System.out.println("b smaller:"+m); //logical operator
54        System.out.println("e="+e); //assignment operator
55        System.out.println("f="+f); //assignment operator
56        System.out.println("g="+g); //assignment operator
57    }
58 }
59

```

Class compiled - no syntax errors

Figure 1 Math Operators

Screenshot of the output



```
BlueJ: Terminal Window - week-3
Options
The Sum of a and b is:65
The Difference of a and b is:47
The Multiplication of a and b is:504
The Division of a and b is:6
The Remainder of a and b is:2
The Student has:Fail
The increment of a is:55
The decrement of b is:10
The a greater than b:true
Is b smaller then a:true
Is a and b equal?:false
a and b not equal:true
Is a greater than or equal to b:true
Is a less than or equal to b:false
Binary operator and =1
Binary operator or =7
Binary operator not =-6
Binary operator left shift =10
Binary operator right shift =2
a greatertrue|
b smaller:true
e=55
f=1
g=5
```

Figure 2 Output of Math Operators

Question no: 2

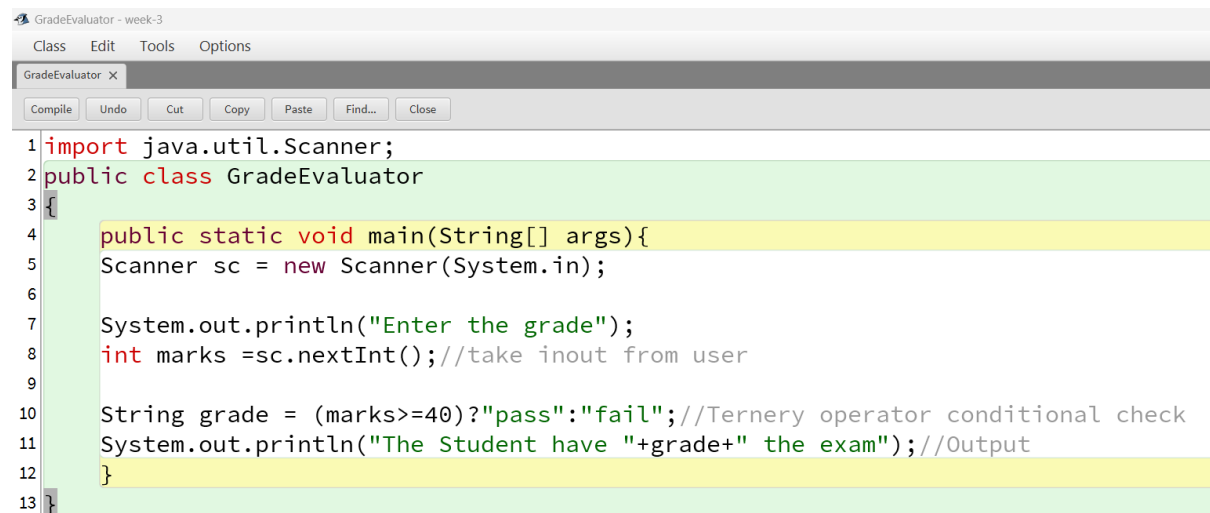
Create a program that:

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

Test case

Objective	To compile and execute the program using BlueJ
Action	<ol style="list-style-type: none">1. BlueJ was opened and the project containing the source code the GradeEvaluator.java was loaded.2. The program file GradeEvaluator.java was compiled using the Compile button in BlueJ.3. The compiled class GradeEvaluator was executed by right-clicking the class and selecting the void main(String[] args) option.
Expected result	The program should compile and display whether or not if the Student has pass or fail without any errors.
Actual result	The program compile and display Pass or Fail according to the user input.
Conclusion	Test was successful.

Screenshot of the code compilation



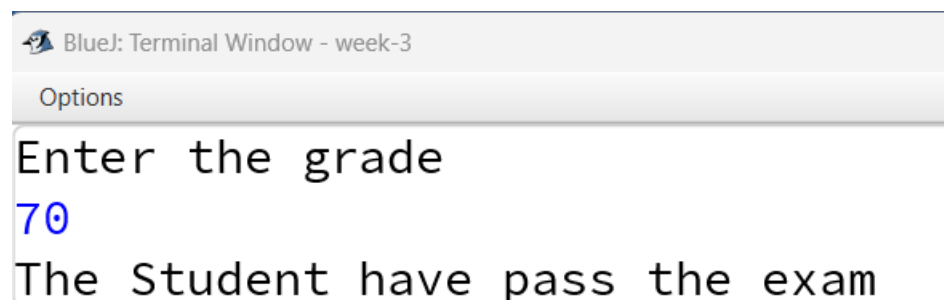
The screenshot shows a Java IDE window titled "GradeEvaluator - week-3". The menu bar includes "Class", "Edit", "Tools", and "Options". Below the menu bar is a toolbar with buttons for "Compile", "Undo", "Cut", "Copy", "Paste", "Find...", and "Close". The main text area displays the following Java code:

```
1 import java.util.Scanner;
2 public class GradeEvaluator
3 {
4     public static void main(String[] args){
5         Scanner sc = new Scanner(System.in);
6
7         System.out.println("Enter the grade");
8         int marks = sc.nextInt(); //take inout from user
9
10        String grade = (marks>=40)? "pass": "fail"; //Ternery operator conditional check
11        System.out.println("The Student have "+grade+" the exam"); //Output
12    }
13 }
```

Class compiled - no syntax errors

Figure 3 Grade Evaluator Code

Screenshot of the output



The screenshot shows a terminal window titled "BlueJ: Terminal Window - week-3". The menu bar includes "Options". The terminal displays the following output:

```
Enter the grade
70
The Student have pass the exam
```

Figure 4 Grade Evaluator Output

Question no: 3

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

Objective	To compile and execute the program using BlueJ
Action	<ol style="list-style-type: none"> 1. BlueJ was opened and the project containing the source code the <code>WaterLevelMonitor.java</code> was loaded. 2. The program file <code>WaterLevelMonitor.java</code> was compiled using the Compile button in BlueJ. 3. The compiled class <code>WaterLevelMonitor</code> was executed by right-clicking the class and selecting the <code>void main(String[] args)</code> option.
Expected result	The program should compile and display the water level either it's 1000L or more without any errors.
Actual result	The program should compile and display whether the water level is more or equal to 1000L or Normal without any errors.
Conclusion	Test was successful.

Screenshot of the code compilation

```

1 public class DataTypeInfoInspector
2 {
3
4     public static void main(String[] args){
5         byte a = 100;
6         short b = 2000;
7         int c = 55000;
8         long d = 10000000L;
9         float e = 10.9f;
10        double f = 100.999d;
11        boolean IsEligible = true;
12        char name = 'A';
13        System.out.println("This is byte datatype:"+a);//byte data type
14        System.out.println("This is short datatype:"+b);//short data type
15        System.out.println("This is int datatype:"+c);//interger data type
16        System.out.println("This is long datatype:"+d);//long data type
17        System.out.println("This is float datatype:"+e);//float datatype
18        System.out.println("This is double datatype:"+f);//double data type
19        System.out.println("This is boolean datatype:"+IsEligible);//boolean data type
20        System.out.println("This is char datatype:"+name);//character data type
21    }
22 }

```

Class compiled - no syntax errors

Figure 5 Primitive data type code

Screenshot of the output

```

This is byte datatype:100
This is short datatype:2000
This is int datatype:55000
This is long datatype:10000000
This is float datatype:10.9
This is double datatype:100.999
This is boolean datatype:true
This is char datatype:A

```

Figure 6 Primitive data type Output

Question no: 4

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

Objective	To compile and execute the program using BlueJ
Action	<ol style="list-style-type: none"> 1. BlueJ was opened and the project containing the source code the Calculate.java was loaded. 2. The program file Calculate.java was compiled using the Compile button in BlueJ. 3. The compiled class Calculate was executed by right-clicking the class and selecting the void main(String[] args) option.
Expected result	The program should compile and display the default value of all datatype without any errors.
Actual result	The program should compile and display default value of all Primitive data types.
Conclusion	Test was successful.

Screenshot of the code compilation

```

1 public class DefaultValues
2 {
3     byte by;
4     short s;
5     int i;
6     long l;
7     float f;
8     double d;
9     boolean b;
10    char c;
11
12    public static void main(String[] args){
13
14        DefaultValues obj = new DefaultValues();
15        System.out.println("--- Default Values of Uninitialized Member Variables ---");
16        System.out.println("byteValue: " + obj.by);
17        System.out.println("shortValue: " + obj.s);
18        System.out.println("intValue: " + obj.i);
19        System.out.println("longValue: " + obj.l);
20        System.out.println("floatValue: " + obj.f);
21        System.out.println("doubleValue: " + obj.d);
22        System.out.println("booleanValue: " + obj.b);
23        System.out.println("charValue: " +obj.c);
24        /*COMMENTARY: Why this won't work for local variable:
25         * If those variable are declared inside the main method (a local variable) and
26         * not initialized, the Java compiler would produce a 'compile time error'
27         * The JVM provides default values (Likes 0,0.0,false) only for instance
28         * variable and static variables. Local variables are stored on the method
29         * stack are not automatically initialized by the JVM. The compiler forces
30         * the programmer assign a value to a local variable before using it to
31         * prevent the program from relying on unknown or garbage values from the stack
32         * memory.
33         */
34    }
35 }
36

```

Class compiled - no syntax errors

Figure 7 Default value code

Screenshot of the output

```

--- Default Values of Uninitialized Member Variables ---
byteValue: 0
shortValue: 0
intValue: 0
longValue: 0
floatValue: 0.0
doubleValue: 0.0
booleanValue: false
charValue: '\0'

```

Figure 8 Default value Output

Question no: 5

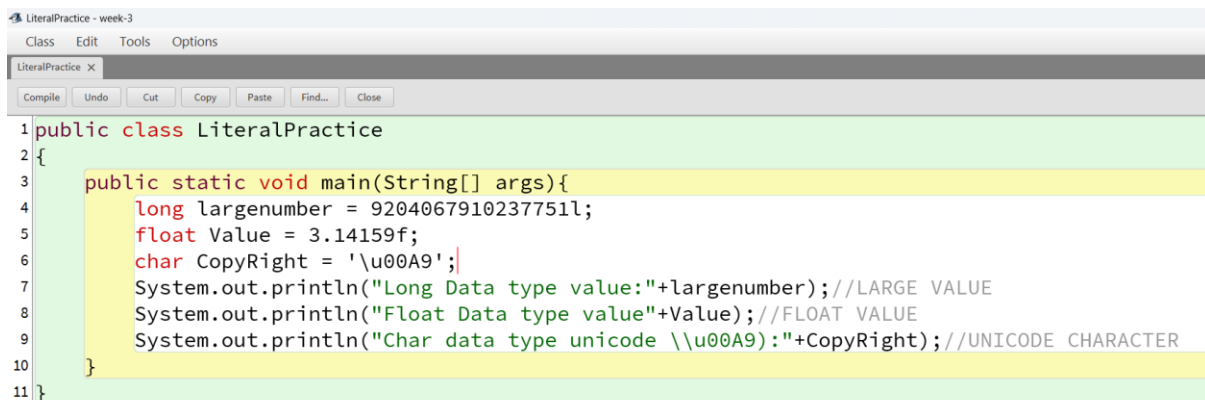
Create a program named LiteralPractice.java that demonstrates the use of specific literals:

- 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

Objective	To compile and execute the program using BlueJ
Action	<ol style="list-style-type: none"> 1. BlueJ was opened and the project containing the source code the LiteralPractice.java was loaded. 2. The program file LiteralPractice.java was compiled using the Compile button in BlueJ. 3. The compiled class LiteralPractice 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 of long, float and Unicode of char datatype without any errors.
Actual result	The program should compile and display long, float data type value and copyright Unicode of char..
Conclusion	Test was successful.

Screenshot of the code compilation

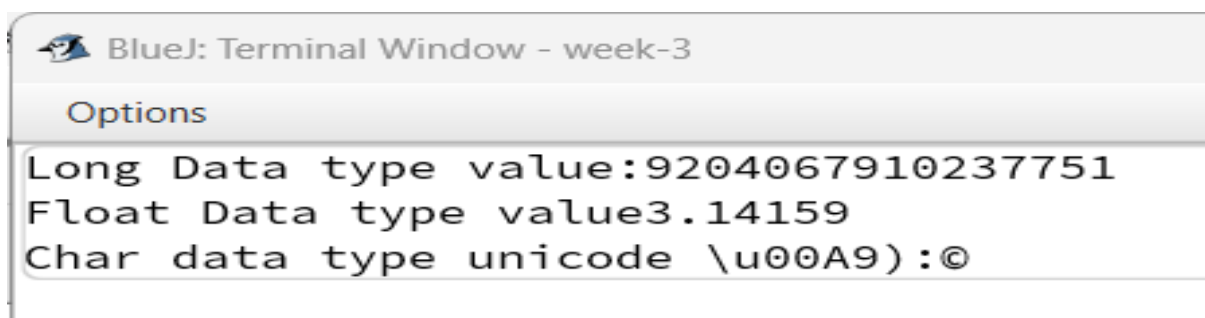


```
1 public class LiteralPractice
2 {
3     public static void main(String[] args){
4         long largenumber = 9204067910237751l;
5         float Value = 3.14159f;
6         char CopyRight = '\u00A9';
7         System.out.println("Long Data type value:"+largenumber);//LARGE VALUE
8         System.out.println("Float Data type value"+Value);//FLOAT VALUE
9         System.out.println("Char data type unicode \\u00A9:"+CopyRight);//UNICODE CHARACTER
10    }
11 }
```

Class compiled - no syntax errors

Figure 9 Literal value practice code

Screenshot of the output



```
BlueJ: Terminal Window - week-3
Options
Long Data type value:9204067910237751
Float Data type value3.14159
Char data type unicode \u00A9):©
```

Figure 10 Literal value practice output

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 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

Objective	To compile and execute the program using BlueJ
Action	<ol style="list-style-type: none"> 1. BlueJ was opened and the project containing the source code the RickshawFare.java was loaded. 2. The program file RickshawFare.java was compiled using the Compile button in BlueJ. 3. The compiled 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 Rickshaw Fare without any errors.
Actual result	The program should compile and display Rickshaw Fare, Discount given to local and Night charges.
Conclusion	Test was successful.

Screenshot of the code compilation

```

1 import java.util.Scanner;
2 public class RickshawFare{
3     public static void main(String[] args) {
4         Scanner sc = new Scanner(System.in);
5
6         final double BASE_FARE = 25.0;
7         final double RATE_PER_KM = 15.0;
8         final double RATE_PER_MINUTE = 2.0;
9         final double LOCAL_DISCOUNT_PERCENT = 0.10;
10        final double NIGHT_CHARGE_PERCENT = 0.25;
11        final double DISCOUNT_THRESHOLD_KM = 5.0;
12
13        System.out.println("=====");
14        System.out.println("    Rickshaw Fare Calculator    ");
15        System.out.println("=====");
16
17        System.out.println("Enter total distance traveled (km): ");
18        double distanceKm = sc.nextDouble();//Input total distance travelled
19
20        System.out.println("Enter total time spent (minutes): ");
21        double timeMinutes = sc.nextDouble();//Input
22
23        System.out.println("Is this a local trip? (true/false): ");
24        boolean isLocal = sc.nextBoolean();
25
26        System.out.println("Is this a night trip? (true/false): ");
27        boolean isNight = sc.nextBoolean();
28
29        double distanceCharge = distanceKm * RATE_PER_KM;
30        double timeCharge = timeMinutes * RATE_PER_MINUTE;
31
32        double baseCalculatedFare = BASE_FARE + distanceCharge + timeCharge;//Total fare
33
34        boolean applyDiscount = isLocal && (distanceKm >= DISCOUNT_THRESHOLD_KM);//check discount
35        double discountAmount = baseCalculatedFare * LOCAL_DISCOUNT_PERCENT;
36
37        double fareAfterDiscount = baseCalculatedFare - (applyDiscount ? discountAmount : 0.0);// Apply discount only if applyDiscount is true
38
39        double nightSurcharge = fareAfterDiscount * NIGHT_CHARGE_PERCENT;
40        double finalFare = fareAfterDiscount + (isNight ? nightSurcharge : 0.0);// Apply night surcharge only if isNight is true
41        System.out.println("-----");
42        System.out.println("Trip Summary:");
43
44        System.out.println("Distance: "+distanceKm+" Time: "+timeMinutes+" min\sec");
45        System.out.println("Discount Applied: "+(applyDiscount ? "Yes" : "No")+" Night Surcharge Applied: "+(isNight ? "Yes" : "No"));
46        System.out.println("-----");
47        System.out.println("TOTAL FINAL FARE: Rs."+ finalFare);//Total fare value
48        System.out.println("=====");
49    }
50 }

```

Class compiled - no syntax errors

Figure 11 Rickshaw Fare Code

Screenshot of the output

```

=====
    Rickshaw Fare Calculator
=====
Enter total distance traveled (km):
10
Enter total time spent (minutes):
20
Is this a local trip? (true/false):
true
Is this a night trip? (true/false):
false

-----
Trip Summary:
Distance: 10.0 Time: 20.0 min ec
Discount Applied:Yes Night Surcharge Applied:No
-----
TOTAL FINAL FARE: Rs.193.5
=====

```

Figure 12 Rickshaw Fare output