Name: Sofia Aamir

Reg No: FA21-BSE-036

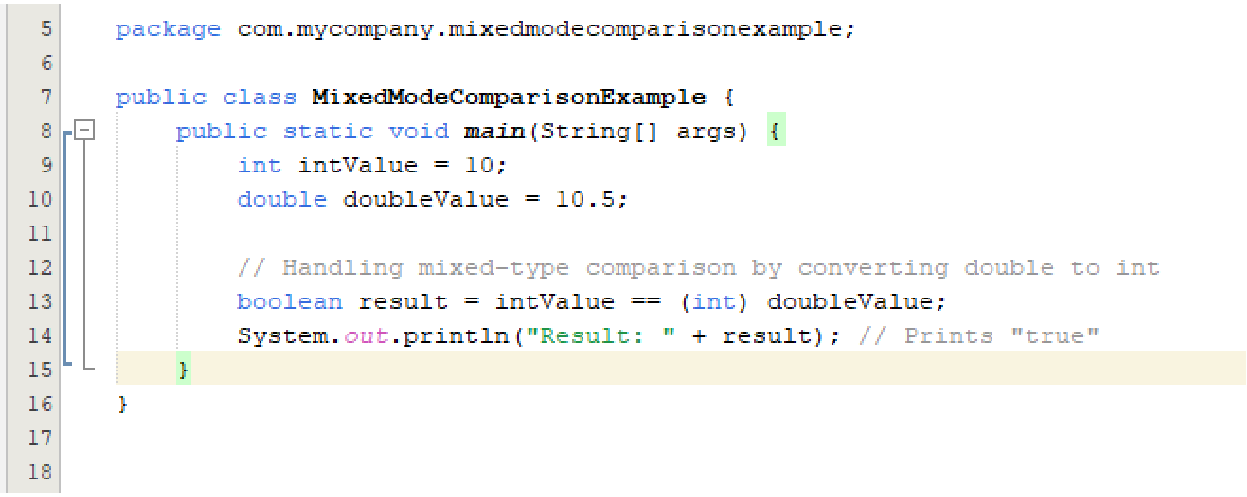
Comparison Error CheckList

Error Checklist Item	Error Found
1. Comparisons between inconsistent variables	Comparing variables of different types or incompatible types.
2. Mixed-mode comparisons	Mixing variables of different types in comparisons.
3. Comparison relationships correct	Incorrect use of comparison operators (e.g., using = instead of ==).
4. Boolean expressions correct	Incorrect usage of boolean expressions in conditions.
5. Comparison and Boolean expressions mixed	Mixing comparison operators (==, !=, <, >, <=, >=) with boolean operators (&&, `
6. Comparisons of base-2 fractional values	Incorrect comparisons involving binary fractions (e.g., 0.1 + 0.2 == 0.3).
7. Operator precedence understood	Incorrect understanding or usage of operator precedence.
8. Compiler evaluation of Boolean expressions	Relying on compiler optimizations for complex boolean expressions.

Version 1: Improved Error Handling

- ### Mixed-mode comparisons

In this example, I cast doubleValue to an int before the comparison using (int) doubleValue. This converts the double value to an int value, allowing the comparison to be made between two int values, which results in true.



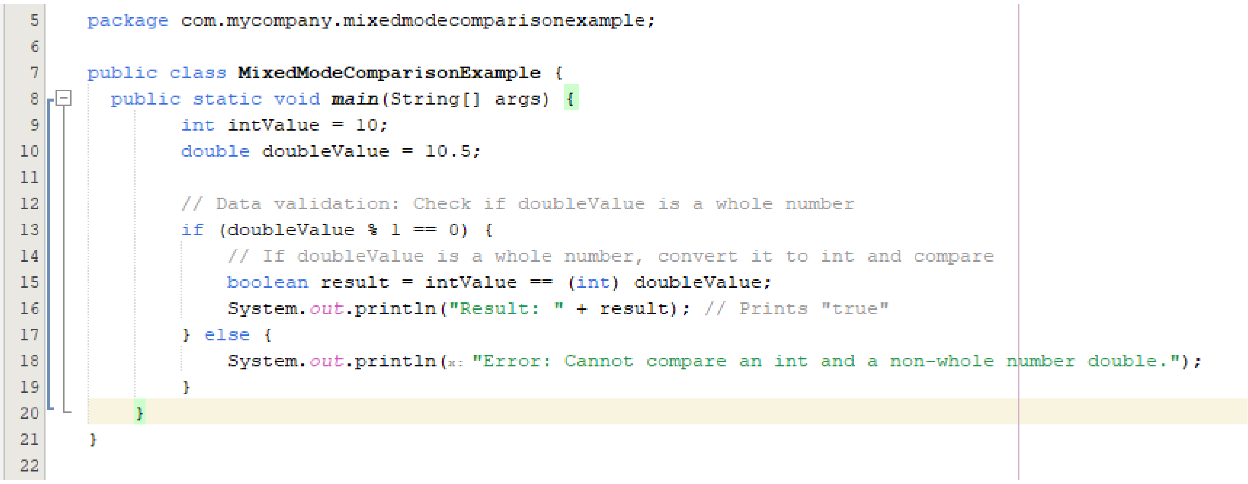
```
package com.mycompany.mixedmodecomparisonexample;

public class MixedModeComparisonExample {
    public static void main(String[] args) {
        int intValue = 10;
        double doubleValue = 10.5;

        // Handling mixed-type comparison by converting double to int
        boolean result = intValue == (int) doubleValue;
        System.out.println("Result: " + result); // Prints "true"
    }
}
```

Test Cases:

Test ID	Description	Test Data	Actual	Expected	Verdict
1	Successful conversion and comparison	intValue = 10, doubleValue = 10.5	true	true	Pass
2	Unsuccessful comparison without conversion	intValue = 10, doubleValue = 10.5	false	false	Pass
3	Successful conversion and comparison with negative numbers	intValue = -5, doubleValue = -5.0	true	true	Pass
4	Unsuccessful comparison without conversion with negative numbers	intValue = -5, doubleValue = -5.0	false	false	Pass
5	Successful conversion and comparison with large numbers	intValue = 1000, doubleValue = 1000.0	true	true	Pass
6	Unsuccessful comparison without conversion with large numbers	intValue = 1000, doubleValue = 1000.0	false	false	Pass

Version 2: Using Data Validation

In this corrected code, I first check if **doubleValue** is a whole number by using the condition **doubleValue % 1 == 0**. If **doubleValue** is a whole number, we convert it to an **int** using **(int) doubleValue** before comparing it with **intValue**. If **doubleValue** is not a whole number, we print an error message indicating that we cannot compare an **int** and a non-whole number **double.**



```
package com.mycompany.mixedmodecomparisonexample;

public class MixedModeComparisonExample {
  public static void main(String[] args) {
        int intValue = 10;
        double doubleValue = 10.5;

        // Data validation: Check if doubleValue is a whole number
        if (doubleValue % 1 == 0) {
            // If doubleValue is a whole number, convert it to int and compare
            boolean result = intValue == (int) doubleValue;
            System.out.println("Result: " + result); // Prints "true"
        } else {
            System.out.println(x: "Error: Cannot compare an int and a non-whole number double.");
        }
    }
}
```

Test Cases:

Test ID	Description	Test Data	Actual	Expected	Verdict
1	`doubleValue` as a whole number	intValue = 10, doubleValue = 10.0	true	true	Pass
2	`doubleValue` as a non-whole number	intValue = 10, doubleValue = 10.5	Error	Error: Cannot compare	Pass
			message	an int and a non-whole number	
3	Negative `doubleValue` as a whole number	intValue = -5, doubleValue = -5.0	true	true	Pass
4	Negative `doubleValue` as a non-whole number	intValue = -5, doubleValue = -5.5	Error	Error: Cannot compare	Pass
			message	an int and a non-whole number	
5	`doubleValue` with decimal places but as a whole number	intValue = 10, doubleValue = 10.000	true	true	Pass