

Section 1: Error-Driven Learning in Java

##Snippet 1

```
public class Main {  
    public void main(String[] args) {  
        System.out.println("Hello, World!");  
    }  
}
```

Error: The main method is missing the `static` keyword.

Solution :

```
public class Main {  
    public static void main(String[] args) {  
        System.out.println("Hello, World!");  
    }  
}
```

Snippet 2

```
public class Main {  
    static void main(String[] args) {  
        System.out.println("Hello, World!");  
    }  
}
```

Error: The `main` method is missing the `public` access modifier.

Solution :

```
public class Main {  
    public static void main(String[] args) {  
        System.out.println("Hello, World!");  
    }  
}
```

Snippet 3

```
public class Main {  
    public static int main(String[] args) {  
        System.out.println("Hello, World!");  
        return 0;  
    }  
}
```

Error: The `main` method cannot have a return type other than `void`.

```
public class Main {  
    public static void main(String[] args) {  
        System.out.println("Hello, World!");  
    }  
}
```

Snippet 4

```
public class Main {  
    public static void main() {  
        System.out.println("Hello, World!");  
    }  
}
```

Error: The `main` method must include a `String[] args` parameter.

```
java  
public class Main {  
    public static void main(String[] args) {  
        System.out.println("Hello, World!");  
    }  
}
```

Snippet 5

```
public class Main {  
    public static void main(String[] args) {  
        System.out.println("Main method with String[] args");  
    }  
    public static void main(int[] args) {  
        System.out.println("Overloaded main method with int[] args");  
    }  
}
```

Error : Java allows method overloading, including overloading `main`.

```
public class Main {  
    public static void main(String[] args) {  
        System.out.println("Main method with String[] args");  
    }  
    public static void main(int[] args) {  
        System.out.println("Overloaded main method with int[] args");  
    }  
}
```

Snippet 6

```
public class Main {  
    public static void main(String[] args) {  
        int x = y + 10;  
        System.out.println(x);  
    }  
}
```

Error: Variable `y` is not declared before usage.

```
public class Main {  
    public static void main(String[] args) {  
        int y = 5;  
        int x = y + 10;  
        System.out.println(x);  
    }  
}
```

Snippet 7

```
public class Main {  
    public static void main(String[] args) {  
        int x = "Hello";  
        System.out.println(x);  
    }  
}
```

Error: Incompatible types: `int` variable cannot store a `String` value.

```
public class Main {  
    public static void main(String[] args) {  
        String x = "Hello";  
        System.out.println(x);  
    }  
}
```

Snippet 8

```
public class Main {  
    public static void main(String[] args) {  
        System.out.println("Hello, World!"  
    }  
}
```

Error: Missing closing parenthesis in `System.out.println` statement.

```
public class Main {  
    public static void main(String[] args) {  
        System.out.println("Hello, World!");  
    }  
}
```

Snippet 9

```
public class Main {  
    public static void main(String[] args) {  
        int class = 10;  
        System.out.println(class);  
    }  
}
```

Error: `class` is a reserved keyword and cannot be used as a variable name.

```
java  
public class Main {  
    public static void main(String[] args) {  
        int myClass = 10;  
        System.out.println(myClass);  
    }  
}
```

Snippet 10

```
public class Main {
    public void display() {
        System.out.println("No parameters");
    }
    public void display(int num) {
        System.out.println("With parameter: " + num);
    }
    public static void main(String[] args) {
        display();
        display(5);
    }
}
```

Error: Cannot call non-static methods from `main` without an object.

```
public class Main {
    public void display() {
        System.out.println("No parameters");
    }
    public void display(int num) {
        System.out.println("With parameter: " + num);
    }
    public static void main(String[] args) {
        Main obj = new Main();
        obj.display();
        obj.display(5);
    }
}
```

Snippet 11

```
public class Main {
    public static void main(String[] args) {
        int[] arr = {1, 2, 3};
        System.out.println(arr[5]);
    }
}
```

Error: This code throws an `ArrayIndexOutOfBoundsException` because the array only has indices 0, 1, and 2, but index 5 is accessed.

```
public class Main {
    public static void main(String[] args) {
        int[] arr = {1, 2, 3};
        if (arr.length > 5) {
            System.out.println(arr[5]);
        } else {
            System.out.println("Index out of bounds");
        }
    }
}
```

Snippet 12

```
public class Main {  
    public static void main(String[] args) {  
        while (true) {  
            System.out.println("Infinite Loop");  
        }  
    }  
}
```

Error: This code results in an infinite loop, continuously printing "Infinite Loop" without termination.

```
public class Main {  
    public static void main(String[] args) {  
        int count = 0;  
        while (count < 5) {  
            System.out.println("Loop iteration: " + count);  
            count++;  
        }  
    }  
}
```

Snippet 13

```
public class Main {  
    public static void main(String[] args) {  
        String str = null;  
        System.out.println(str.length());  
    }  
}
```

Error: This code throws a `NullPointerException` because `str` is `null`, and calling `length()` on `null` is invalid.

```
public class Main {  
    public static void main(String[] args) {  
        String str = "Hello";  
        System.out.println(str.length());  
    }  
}
```

Snippet 14

```
public class Main {  
    public static void main(String[] args) {  
        double num = "Hello";  
        System.out.println(num);  
    }  
}
```

Error: Compilation error due to assigning a `String` value to a `double` variable.

```
public class Main {  
    public static void main(String[] args) {  
        String num = "Hello";  
        System.out.println(num);  
    }  
}
```

Snippet 15

```
public class Main {  
    public static void main(String[] args) {  
        int num1 = 10;  
        double num2 = 5.5;  
        int result = num1 + num2;  
        System.out.println(result);  
    }  
}
```

Error: Compilation error due to assigning a `double` result to an `int` variable.

```
public class Main {  
    public static void main(String[] args) {  
        int num1 = 10;  
        double num2 = 5.5;  
        double result = num1 + num2;  
        System.out.println(result);  
    }  
}
```

Snippet 16

```
public class Main {  
    public static void main(String[] args) {  
        int num = 10;  
        double result = num / 4;  
        System.out.println(result);  
    }  
}
```

Error: Since both `num` and `4` are integers, integer division occurs, resulting in `2.0` instead of `2.5`.

```
public class Main {  
    public static void main(String[] args) {  
        int num = 10;  
        double result = num / 4.0;  
        System.out.println(result);  
    }  
}
```

Snippet 17

```
public class Main {  
    public static void main(String[] args) {  
        int a = 10;  
        int b = 5;  
        int result = a ** b;  
        System.out.println(result);  
    }  
}
```

Error: The `**` operator does not exist in Java for exponentiation.

```
import java.lang.Math;  
public class Main {  
    public static void main(String[] args) {  
        int a = 10;  
        int b = 5;  
        double result = Math.pow(a, b);  
        System.out.println(result);  
    }  
}
```

Snippet 18

```
public class Main {  
    public static void main(String[] args) {  
        int a = 10;  
        int b = 5;  
        int result = a + b * 2;  
        System.out.println(result);  
    }  
}
```

Error: Multiplication has a higher precedence than addition, so `b * 2` is evaluated first, resulting in `10 + 10 = 20`.

```
public class Main {  
    public static void main(String[] args) {  
        int a = 10;  
        int b = 5;  
        int result = (a + b) * 2;  
        System.out.println(result);  
    }  
}
```

Snippet 19

```
public class Main {  
    public static void main(String[] args) {  
        int a = 10;  
        int b = 0;  
        int result = a / b;  
        System.out.println(result);  
    }  
}
```

Error: Division by zero throws an `ArithmeticException`.

```
public class Main {  
    public static void main(String[] args) {  
        int a = 10;  
        int b = 0;  
        if (b != 0) {  
            int result = a / b;  
            System.out.println(result);  
        } else {  
            System.out.println("Division by zero is not allowed");  
        }  
    }  
}
```

Snippet 20

```
public class Main {  
    public static void main(String[] args) {  
        System.out.println("Hello, World")  
    }  
}
```

Error: Missing semicolon at the end of the `System.out.println` statement.

```
public class Main {  
    public static void main(String[] args) {  
        System.out.println("Hello, World");  
    }  
}
```


Snippet 21

```
public class Main {  
    public static void main(String[] args) {  
        System.out.println("Hello, World!");  
        // Missing closing brace here  
    }  
}
```

Error: The closing brace for the `main` method is missing, causing a compilation error.

```
public class Main {  
    public static void main(String[] args) {  
        System.out.println("Hello, World!");  
    }  
}
```

Snippet 22

```
public class Main {  
    public static void main(String[] args) {  
        static void displayMessage() {  
            System.out.println("Message");  
        }  
    }  
}
```

Error: A method cannot be declared inside another method.

```
public class Main {  
    public static void displayMessage() {  
        System.out.println("Message");  
    }  
    public static void main(String[] args) {  
        displayMessage();  
    }  
}
```

Snippet 23

```
public class Confusion {  
    public static void main(String[] args) {  
        int value = 2;  
        switch(value) {  
            case 1:  
                System.out.println("Value is 1");  
            case 2:  
                System.out.println("Value is 2");  
            case 3:  
                System.out.println("Value is 3");  
            default:  
                System.out.println("Default case");  
        }  
    }  
}
```

Error: Missing `break` statements cause fall-through behavior, executing all cases after a match.

```
public class Confusion {  
    public static void main(String[] args) {  
        int value = 2;  
        switch(value) {  
            case 1:  
                System.out.println("Value is 1");  
                break;  
            case 2:  
                System.out.println("Value is 2");  
                break;  
            case 3:  
                System.out.println("Value is 3");  
                break;  
            default:  
                System.out.println("Default case");  
        }  
    }  
}
```

Snippet 24

```
public class MissingBreakCase {  
    public static void main(String[] args) {  
        int level = 1;  
        switch(level) {  
            case 1:  
                System.out.println("Level 1");  
            case 2:  
                System.out.println("Level 2");  
            case 3:  
                System.out.println("Level 3");  
            default:  
                System.out.println("Unknown level");  
        }  
    }  
}
```

Error: Missing `break` statements cause fall-through, printing all subsequent cases.

```
public class MissingBreakCase {  
    public static void main(String[] args) {  
        int level = 1;  
        switch(level) {  
            case 1:  
                System.out.println("Level 1");  
                break;  
            case 2:  
                System.out.println("Level 2");  
                break;  
            case 3:  
                System.out.println("Level 3");  
                break;  
            default:  
                System.out.println("Unknown level");  
        }  
    }  
}
```

Snippet 25

```
public class Switch {
    public static void main(String[] args) {
        double score = 85.0;
        switch(score) {
            case 100:
                System.out.println("Perfect score!");
                break;
            case 85:
                System.out.println("Great job!");
                break;
            default:
                System.out.println("Keep trying!");
        }
    }
}
```

Error: `switch` does not support `double` types.

```
public class Switch {
    public static void main(String[] args) {
        int score = 85;
        switch(score) {
            case 100:
                System.out.println("Perfect score!");
                break;
            case 85:
                System.out.println("Great job!");
                break;
            default:
                System.out.println("Keep trying!");
        }
    }
}
```

Snippet 26

```
public class Switch {  
    public static void main(String[] args) {  
        int number = 5;  
        switch(number) {  
            case 5:  
                System.out.println("Number is 5");  
                break;  
            case 5:  
                System.out.println("This is another case 5");  
                break;  
            default:  
                System.out.println("This is the default case");  
        }  
    }  
}
```

Error: Duplicate `case 5` labels cause a compilation error.

```
public class Switch {  
    public static void main(String[] args) {  
        int number = 5;  
        switch(number) {  
            case 5:  
                System.out.println("Number is 5");  
                break;  
            case 6:  
                System.out.println("This is case 6");  
                break;  
            default:  
                System.out.println("This is the default case");  
        }  
    }  
}
```

Section 2: Java Programming with Conditional Statements

Question 1: Grade Classification

Write a program to classify student grades based on the following criteria:

If the score is greater than or equal to 90, print "A"

If the score is between 80 and 89, print "B"

If the score is between 70 and 79, print "C"

If the score is between 60 and 69, print "D"

? If the score is less than 60, print "F"

Solution :

```
import java.util.Scanner;

public class GradeClassification {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the student's score: ");
        int score = scanner.nextInt();
        scanner.close();

        if (score >= 90) {
            System.out.println("A");
        } else if (score >= 80 && score <= 89) {
            System.out.println("B");
        } else if (score >= 70 && score <= 79) {
            System.out.println("C");
        } else if (score >= 60 && score <= 69) {
            System.out.println("D");
        } else {
            System.out.println("F");
        }
    }
}
```

Output Clear

```
Enter the student's score: 78
C

=== Code Execution Successful ===
```

Question 2: Days of the Week

Write a program that uses a nested switch statement to print out the day of the week based on an integer input (1 for Monday, 2 for Tuesday, etc.). Additionally, within each day, print whether it is a weekday or weekend.

Solution :

```
public class DaysOfWeek {  
    public static void main(String[] args) {  
        int day = 3; // Example input  
  
        switch (day) {  
            case 1, 2, 3, 4, 5:  
                System.out.println("Weekday");  
                break;  
            case 6, 7:  
                System.out.println("Weekend");  
                break;  
            default:  
                System.out.println("Invalid day");  
        }  
    }  
}
```

Output Clear

Weekday

=== Code Execution Successful ===

Question 3: Calculator

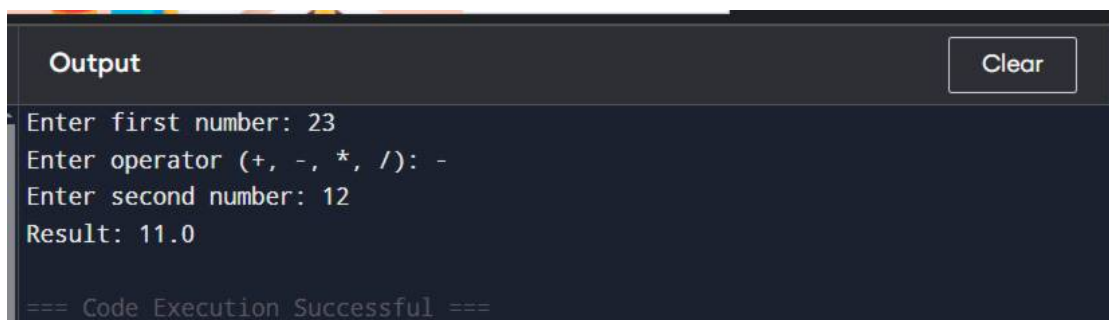
Write a program that acts as a simple calculator. It should accept two numbers and an operator (+, -, *, /) as input. Use a switch statement to perform the appropriate operation. Use nested if else to check if division by zero is attempted and display an error message.

Solution :

```
import java.util.Scanner;

public class Calculator {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter first number: ");
        double num1 = scanner.nextDouble();
        System.out.print("Enter operator (+, -, *, /): ");
        char operator = scanner.next().charAt(0);
        System.out.print("Enter second number: ");
        double num2 = scanner.nextDouble();
        scanner.close();

        switch (operator) {
            case '+':
                System.out.println("Result: " + (num1 + num2));
                break;
            case '-':
                System.out.println("Result: " + (num1 - num2));
                break;
            case '*':
                System.out.println("Result: " + (num1 * num2));
                break;
            case '/':
                if (num2 == 0) {
                    System.out.println("Error: Division by zero is not allowed.");
                } else {
                    System.out.println("Result: " + (num1 / num2));
                }
                break;
            default:
                System.out.println("Invalid operator");
        }
    }
}
```



The screenshot shows a dark-themed IDE output window. At the top, there is a tab labeled 'Output' and a 'Clear' button. The output text is as follows:

```
Enter first number: 23
Enter operator (+, -, *, /): -
Enter second number: 12
Result: 11.0

=== Code Execution Successful ===
```


Question 4: Discount Calculation

Write a program to calculate the discount based on the total purchase amount. Use the following criteria:

If the total purchase is greater than or equal to Rs.1000, apply a 20% discount.

If the total purchase is between Rs.500 and Rs.999, apply a 10% discount.

If the total purchase is less than Rs.500, apply a 5% discount.

Additionally, if the user has a membership card, increase the discount by 5%.

Solution :

```
public class DiscountCalculator {
    public static void main(String[] args) {
        double totalPurchase = 1200;
        boolean hasMembership = true;
        double discount;

        if (totalPurchase >= 1000) {
            discount = 20;
        } else if (totalPurchase >= 500) {
            discount = 10;
        } else {
            discount = 5;
        }

        if (hasMembership) {
            discount += 5;
        }

        double finalPrice = totalPurchase - (totalPurchase * discount / 100);
        System.out.println("Final price: Rs." + finalPrice);
    }
}
```

Output Clear

Final price: Rs.900.0

=== Code Execution Successful ===

Question 5: Student Pass/Fail Status with Nested Switch

Write a program that determines whether a student passes or fails based on their grades in three subjects. If the student scores more than 40 in all subjects, they pass. If the student fails in one or more subjects, print the number of subjects they failed in.

Solution:

```
public class StudentPassFail {  
    public static void main(String[] args) {  
        int subject1 = 35, subject2 = 45, subject3 = 30;  
        int failCount = 0;  
  
        if (subject1 < 40) failCount++;  
        if (subject2 < 40) failCount++;  
        if (subject3 < 40) failCount++;  
  
        switch (failCount) {  
            case 0:  
                System.out.println("Pass");  
                break;  
            default:  
                System.out.println("Fail in " + failCount + " subjects");  
        }  
    }  
}
```



The screenshot shows a dark-themed output window. At the top, the word "Output" is on the left and a "Clear" button is on the right. Below this, the text "Fail in 2 subjects" is displayed. At the bottom, a status bar shows "=== Code Execution Successful ===" with a cursor at the end.