Section 1: Error-Driven Learning in Java

Snippet 1:

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

- **What error do you get when running this code?**

Error:
Main method is not static in class Main, please define the main method as:
public static void main(String[] args)

Explain:
main method defined in class called "Main", but it is missing the "static" keyword.

Correct Code:

```
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!");
       }
 }
```

- **What happens when you compile and run this code?**

Error:
Main method not found in class Main, please define the main method as:
public static void main(String[] args)
or a JavaFX application class must extend javafx.application.Application

Explain:
Main method define in class main is missing public keyword.

Correct Code:

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

<u>Snippet 3:</u>

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

• **What error do you encounter? Why is void used in the main method?**

Error:

Main method must return a value of type void in class Main, please define the main method as: public static void main(String[] args)
Error: incompatible types: unexpected return value
return 0;

Explain:

The main method should have a void return type instead of int. & void means no return value. The method can't return anything so if return 0 is written it leads to compilation error.
The void keyword in Java is used to specify that a method does not return any value. It is a return type that indicates the method performs an action but does not produce a result that can be used elsewhere in the code.

Correct Code:

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

<u>Snippet 4:</u>

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

• **What happens when you compile and run this code? Why is String[] args needed?**

Error:

Main method not found in class Main, please define the main method as:

public static void main(String[] args)

or a JavaFX application class must extend javafx.application.Application

Explain:

String[] args in Java is used to pass command-line arguments to a Java program. It allows external data to be passed into the program when it starts, making it and adaptable to various inputs.

Correct code :

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

<u>Snippet 5:</u>

```
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");
             }
}
```

- **Can you have multiple main methods? What do you observe?**

Explain:

Yes, we can have multiple main methods, but jvm looks for public static void main(String[] args) method as entry point for execution.

<u>Snippet 6:</u>

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

- **What error occurs? Why must variables be declared?**

Error:

```
cannot find symbol
            int x = y + 10; System.out.println(x);
               ^
  symbol:   variable y
  location: class Main
```

Explain: The variable y is not declared it needs to be declared before using it.

Correct code :

```
public class Main {
        public static void main(String[] args) {
     int y = 2;
     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);
         }
 }
```

- **What compilation error do you see? Why does Java enforce type safety?**

Error:incompatible types: String cannot be converted to int

```
      int x = "Hello";
```

Explain:

Type safety is enforced, means that variables must be declared with a specific type, and only values of that type can be assigned to it. int is a primitive data type that can only store integer values. “Hello” is a string.Java enforces type safety : errors occur at Compile Time.

Correct Code :

```
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!"
        }
 }
```

- **What syntax errors are present? How do they affect compilation?**

Error: : ')' expected

```
        System.out.println("Hello, World!"
```

Explain :
The parenthesis is missing & ; is missing the code is unable to run.

Correct code :
```
 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);
                }
 }
```
- **What error occurs? Why can't reserved keywords be used as identifiers?**

Error:
```
    error: not a statement
    int class = 10;
    ^
    main.java:57: error: ';' expected
    int class = 10;
       ^
    main.java:57: error: <identifier> expected
    int class = 10;
             ^
    main.java:58: error: <identifier> expected
    System.out.println(class);
                      ^
    main.java:58: error: illegal start of type
    System.out.println(class);
                       ^
    main.java:58: error: <identifier> expected
    System.out.println(class);
                            ^
    main.java:60: error: reached end of file while parsing
}
```

Explain:
Reserved keywords in programming languages cannot be used as identifiers because they have a predefined meaning within the language, which is necessary for the compiler to understand the code's functionality; using them as variable names would create confusion the compiler will not interpreting the code correctly.

Correct Code:

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

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);
        }
}
```

- **What happens when you compile and run this code? Is method overloading allowed?**

Error:

```
    non-static method display() cannot be referenced from a static context
         display();
         ^
    error: non-static method display(int) cannot be referenced from a static context
         display(5);
```

Explain:

Yes, method overloading is allowed.it allow multiple method with same name but different parameters.

Snippet 11:

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

- **What runtime exception do you encounter? Why does it occur?**

Error:
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 5 out of bounds for length 3 at main.

Explain :
here, the array index[5] is not present the valid index are 0,1,2 only.

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

<u>Snippet 12:</u>
```
public class Main {
        public static void main(String[] args) {
                while (true) {
                System.out.println("Infinite Loop");
                }
         }
 }
```

- **What happens when you run this code? How can you avoid infinite loops?**

Explain:
The program will go in infinite loop to avoid infinite loop in while condition (i<=5) should be in finite set of instructions

Correct Code :
```
public class Main {
        public static void main(String[] args) {
                int i = 1;
                while (i<=5) {
                System.out.println("Infinite Loop");
                i++;
           }
        }
}
```

Snippet 13:

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

• What exception is thrown? Why does it occur?

Error:
Exception in thread "main" java.lang.NullPointerException.

Explain:
Occurs when using a variable that does not point to an object and refers to nothing

Snippet 14:

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

• What compilation error occurs? Why does Java enforce data type constraints?
Error:
```
incompatible types: String cannot be converted to double
     double num = "Hello";
```

Explain:
datatype should be change to string.

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

<u>Snippet 15:</u>

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

• What error occurs when compiling this code? How should you handle different data types in operations?

Error:

```
incompatible types: possible lossy conversion from double to int
      int result = num1 + num2;
```

Explain: we can change the datatype or do explicit typecasting to store in result.

Correct code :

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

<u>Snippet 16:</u>

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

• What is the result of this operation? Is the output what you expected?

Explain:

2.0 is the result of this operation**.** Here, the result is in double so 10/4 leads to 2.0 in double

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);
        }
}
```

- **What compilation error occurs? Why is the ** operator not valid in Java?**

Error:
illegal start of expression
int result = a ** b;

Explain:
Java does not have a built-in exponentiation operator. Instead, it provides the Math.pow() method to perform exponentiation. The correct way to calculate a^b in Java is:

Correct Code:

```
public class Main {
        public static void main(String[] args) {
                int a = 10;
                int b = 5;
                double result = Math.po(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);
   }
 }
```

- **What is the output of this code? How does operator precedence affect the result?**

Explain :
The output of this code is 20.
Here , multiplication has higher precedence than addition. So, multiplication will happen first then addition.

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);
            }
}
```

• What runtime exception is thrown? Why does division by zero cause an issue in Java?

Explian:
Exception in thread "main" java.lang.ArithmeticException: / by zero
In Java, division by zero is undefined for integer types.

Snippet 20:

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

• What syntax error occurs? How does the missing semicolon affect compilation?

Error:

```
';' expected
 System.out.println("Hello, World")
```

Explain:
semicolon (;) is used to mark the end of a statement. Without it, the Java compiler will not be able to properly compile the code.

Snippet 21:

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

• What does the compiler say about mismatched braces?

Error:

```
reached end of file while parsing
}
```

Explain:
The mismatched braces are causing a syntax error**.**

<u>Snippet 22:</u>

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

• **What syntax error occurs? Can a method be declared inside another method?**

Error:

```
illegal start of expression
 static void displayMessage() {
 ^
error: class, interface, or enum expected
}
```

Explain:
methods cannot be declared inside other methods is the syntax error occurred.

<u>Snippet 23:</u>

```
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 to Investigate: Why does the default case print after "Value is 2"? How can you prevent the program from executing the default case?**

Error:

```
Value is 2
Value is 3
Default case
```

Explain:
To prevent the default case from executing we need to add a break statement at the end of each case block.

Correct code:

```
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 to Investigate: When level is 1, why does it print "Level 1", "Level 2", "Level 3", and "Unknown level"? What is the role of the break statement in this situation?**

Error:

```
class MissingBreakCase is public, should be declared in a file named
MissingBreakCase.java
public class MissingBreakCase {
```

Explain:
The switch statement checks the value of level.
When the value matches case 1, it executes the code under case 1, printing "Level 1".There is no break statement after case 1, the program does not stop there and continues to execute.
The break statement is used in switch statements to prevent continue execution.
When the break statement is there it exits the switch block, preventing the execution.

Correct code:
```
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");
        }
    }
 }
```

<u>Snippet 25:</u>

```
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 to Investigate: Why does this code not compile? What does the error tell you about the types allowed in switch expressions? How can you modify the code to make it work?**

Error:
incompatible types: possible lossy conversion from double to int
switch(score) {

Explain:
Java's switch statement does not support double values.Convert double to int.

Correct Code:
```
public class Switch {
        public static void main(String[] args) {
                double score = 85.0;
                int score1 = (int) score;
        switch(score1) {
            case 100:
                System.out.println("Perfect score!");
                break;
             case 85:
                System.out.println("Great job!");
                break;
            default:
                System.out.println("Keep trying!");
                }
        }
 }
```

<u>**Snippet 26:**</u>

```
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 to Investigate: Why does the compiler complain about duplicate case labels? What happens when you have two identical case labels in the same switch block?**

Error:
duplicate case label
case 5:

Explain:
Each case label must be unique within the same switch block. Having two case 5: labels is not allowed because the compiler cannot determine which block of code to execute when the value of number is 5.

Correct code:
```
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 another case 5");
                        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"

Program:

```
import java.util.Scanner;

class Section2{
   public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        int score = sc.nextInt();

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

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.

Program:

```
import java.util.Scanner;

public class Section2 {
   public static void main(String[] args) {
           Scanner sc = new Scanner(System.in);
      System.out.print("Day of the week: ");
      int n = sc.nextInt();
      switch (n) {
         case 1:
            System.out.println("Monday - Weekday");
            break;
         case 2:
            System.out.println("Tuesday - Weekday");
            break;
         case 3:
            System.out.println("Wednesday - Weekday");
            break;
         case 4:
            System.out.println("Thursday - Weekday");
            break;
         case 5:
            System.out.println("Friday - Weekday");
            break;
         case 6:
            System.out.println("Saturday - Weekday");
            break;
         case 7:
            System.out.println("Sunday - Weekend");
            break;
         default:
            System.out.println("Please enter a number");
            break;
      }
   }
}
```

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 ifelse to check if division by zero is attempted and display an error message.

Program:

```
import java.util.Scanner;
public class Section2 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the first number: ");
        double num1 = sc.nextDouble();
        System.out.print("Enter an operator (+, -, *, /): ");
        char operator = sc.next().charAt(0);
        System.out.print("Enter the second number: ");
        double num2 = sc.nextDouble();
        double result;
        switch (operator) {
            case '+':
                result = num1 + num2;
                System.out.println(num1 + " + " + num2 + " = " + result);
                break;
            case '-':
                result = num1 - num2;
                System.out.println(num1 + " - " + num2 + " = " + result);
                break;
            case '*':
                result = num1 * num2;
                System.out.println(num1 + " * " + num2 + " = " + result);
                break;
            case '/':
                if (num2 == 0) {
                    System.out.println("Error: Division by zero is not allowed.");
                } else {
                    result = num1 / num2;
                    System.out.println(num1 + " / " + num2 + " = " + result);
                }
                break;
            default:
                System.out.println("Error: Invalid operator.");
                break;
        }
    }
}
```

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%.

Program:

```
import java.util.Scanner;

public class Section2 {
    public static void main(String[] args) {
         Scanner sc = new Scanner(System.in);

        System.out.print("Enter total amount: ");
        double totalAmount = sc.nextDouble();

        System.out.print("membership (1/2): ");
        int membership = sc.nextInt();

        double discountper = 0;

        if (totalAmount >= 1000) {
            discountper = 20;
        } else if (totalAmount >= 500) {
            discountper = 10;
        } else {
            discountper = 5;
        }
               if (membership==1) {
            discountper = discountper + 5;
        }
        double discountAmount = (discountper / 100) * totalAmount;
        double finalAmount = totalAmount - discountAmount;

        System.out.println("Total Purchase " + totalAmount);
        System.out.println("Discount Applied: " + discountper + "%");
        System.out.println("Discount Amount " + discountAmount);
        System.out.println("Final Amount " + finalAmount);

    }
}
```

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.

Program:

```
import java.util.Scanner;
public class Section2{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("marks1: ");
        int mark1 = sc.nextInt();
        System.out.print("marks2: ");
        int mark2 = sc.nextInt();
        System.out.print("marks3: ");
        int mark3 = sc.nextInt();
        int fail = 0;
        for (int i = 1; i <= 3; i++) {
            int mark = 0;
            switch (i) {
                case 1:
                    mark = mark1;
                    break;
                case 2:
                    mark = mark2;
                    break;
                case 3:
                    mark = mark3;
                    break;
            }
        switch (mark >= 40 ? 1 : 2) {
                case 1:
                    break;
                case 2:
                    fail++;
                    break;
            }
        }
        if (fail == 0) {
            System.out.println("Congratulations.. you are pass.");
        } else {
            System.out.println("You have fail in " + fail + " subject.");
        }
    }
}
```