CDAC Mumbai
Lab Assignment

Section 1: Error-Driven Learning in Java

Objective: This assignment focuses on understanding and fixing common errors encountered in Java programming. By analyzing and correcting the provided code snippets, you will develop a deeper understanding of Java's syntax, data types, and control structures.

Instructions:

1. **Identify the Errors:** Review each code snippet to identify the errors or issues present.
2. **Explain the Error:** Write a brief explanation of the error and its cause.
3. **Fix the Error:** Modify the code to correct the errors. Ensure that the code compiles and runs as expected.
4. **Submit Your Work:** Provide the corrected code along with explanations for each snippet.

Snippet 1:
```
public class Main {
   public void main(String[] args) {
      System.out.println("Hello, World!");
   }
}
```

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

```
public class hello {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
//(. It should be public static void main(String[] args) instead of public
void main(String[] args).)
```

Snippet 2:
```
public class Main {
   static void main(String[] args) {
      System.out.println("Hello, World!");
   }
}
```

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

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

}
//Main method not found in class Main main method as:
//public static void main(String[] args)

Snippet 3:
```java
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?**

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


//Main method must return void in class Main, found int
```

Snippet 4:
```java
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?**

```java
public class Main {
    public static void main(String[] args)
    { System.out.println("Hello, World!");
    }
}
//Main method not found in class Main (String[] args)
```

Snippet 5:
```java
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?**

```java
public class Main {
        public static void main(String[] args) {
            System.out.println("Main method with String[] args");
            // Calling the overloaded main method
            main(new int[]{1, 2, 3});
        }

        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);
   }
}
```

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

```
•   public class Main {
•           public static void main(String[] args) {
•                int y = 5;// Example initialization
•                int x = y + 10;
•                System.out.println(x);
•           }
•        }
•
•    //Error: cannot find symbol
•     //symbol:    variable y
•     //location: class Main
```
•

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?**

```
•   public class Main {
•           public static void main(String[] args) {
•                String x = "Hello";
•                System.out.println(x);
•           }
•        }
```
•
string "Hello", declare x as a String:

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?**

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

- **Missing);**

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?**

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

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?**

```
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(); // Create an instance of Main
        obj.display();          // Call instance method without parameters
        obj.display(5);         // Call instance method with 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?**

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

Snippet 12:

```
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?**

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

-

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?**

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

-

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?**

```
public class Main {
    public static void main(String[] args) {
```

```
            double num = "Hello";
            System.out.println(num); // Compilation error
        }
    }
```

●

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);
   }
}
```

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

```
public class Main {
    public static void main(String[] args) {
        int num1 = 10;
        double num2 = 5.5;
        int result = num1 + num2; // Compilation error
        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);
    }
}
```

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

```
public class Main {
    public static void main(String[] args) {
        int num = 10;
        double result = num / 4;
        System.out.println(result); // Output: 2.5
    }
}
```

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?**

```
public class Main {
    public static void main(String[] args) {
        int a = 10;
        int b = 5;
        int result = a ** b; // Compilation error
        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?**

```
public class Main {
    public static void main(String[] args) {
        int a = 10;
        int b = 5;
        int result = a + b * 2;
        System.out.println(result); // Output: 20
    }
}
```

-

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?**

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

-

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?**

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

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?**

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

Snippet 22:
```
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?**

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

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

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

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?

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

-

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 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?

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

Snippet 26:

```java
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?

```java
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");
        }
    }
}
```

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"

```java
import java.util.Scanner;

public class GradeClassification {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter your score: ");
        int score = scanner.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");
        }
        scanner.close();
    }
}
```

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 aninteger input (1 for Monday, 2 for Tuesday, etc.). Additionally, within each day, print whether itis a weekday or weekend.

```
import java.util.Scanner;

public class DaysOfWeek {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number (1-7) for the day of the week: ");
        int day = scanner.nextInt();

        switch (day) {
            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: Weekend");
                break;
            case 7:
                System.out.println("Sunday: Weekend");
```

```
                    break;
                default:
                    System.out.println("Invalid input! Please enter a number
    between 1 and 7.");
                }
            scanner.close();
        }
    }
```

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.

```java
import java.util.Scanner;

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

        double result;
        switch (operator) {
            case '+':
                result = num1 + num2;
                System.out.println("Result: " + result);
                break;
            case '-':
                result = num1 - num2;
                System.out.println("Result: " + result);
                break;
            case '*':
                result = num1 * num2;
                System.out.println("Result: " + result);
                break;
            case '/':
                if (num2 != 0) {
                    result = num1 / num2;
                    System.out.println("Result: " + result);
                } else {
```

```java
                        System.out.println("Error! Division by zero is not
allowed.");
                    }
                    break;
                default:
                    System.out.println("Invalid operator!");
            }
            scanner.close();
        }
    }
```

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.

```java
import java.util.Scanner;

public class DiscountCalculation {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter total purchase amount: Rs. ");
        double totalAmount = scanner.nextDouble();
        System.out.print("Do you have a membership card? (yes/no): ");
        String membership = scanner.next().toLowerCase();

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

        if (membership.equals("yes")) {
            discount += 5;
        }

        double finalAmount = totalAmount - (totalAmount * discount / 100);
        System.out.println("Discount applied: " + discount + "%");
        System.out.println("Final amount to be paid: Rs. " + finalAmount);
```

```
            scanner.close();
        }
    }
```

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

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.

```java
import java.util.Scanner;

public class StudentPassFail {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int[] marks = new int[3];
        int failCount = 0;

        for (int i = 0; i < 3; i++) {
            System.out.print("Enter marks for Subject " + (i + 1) + ": ");
            marks[i] = scanner.nextInt();
            if (marks[i] < 40) {
                failCount++;
            }
        }

        switch (failCount) {
            case 0:
                System.out.println("Pass: You passed all subjects!");
                break;
            case 1:
                System.out.println("Fail: You failed in 1 subject.");
                break;
            case 2:
                System.out.println("Fail: You failed in 2 subjects.");
                break;
            case 3:
                System.out.println("Fail: You failed in all subjects.");
                break;
            default:
                System.out.println("Invalid input!");
        }

        scanner.close();
    }
}
```

Section 3: Food for Thought: Research and Read More About

1. Evolution of Programming Languages

- **Research Topic:** Explore the different levels of programming languages: Low-level, High-level, and Assembly-level languages.
  - ○ **Questions to Ponder:**
    - ▪ What is a Low-level language? Give examples and explain how they work.
    - ▪ What is a High-level language? How does it differ from a low-level language in terms of abstraction and usage?
    - ▪ What is an Assembly-level language, and what role does it play in programming?
    - ▪ Why do we need different levels of programming languages? What are the trade-offs between simplicity and control over the hardware?

2. Different Programming Languages and Their Usage

- **Research Topic:** Explore different programming languages and understand their use cases.
  - ○ **Questions to Ponder:**
    - ▪ What are the strengths and weaknesses of languages like C, Python, Java, JavaScript, C++, Ruby, Go, etc.?
    - ▪ In which scenarios would you choose a specific language over others? For example, why would you use JavaScript for web development but Python for data science?
    - ▪ Can one programming language be used for all types of software development? Why or why not?

3. Which Programming Language is the Best?

- **Research Topic:** Investigate the debate around the "best" programming language.
  - ○ **Questions to Ponder:**
    - ▪ Is there truly a "best" programming language? If so, which one, and why?
    - ▪ If a language is considered the best, why aren't all organizations using it? What factors influence the choice of a programming language in an organization (e.g., cost, performance, ecosystem, or community support)?
    - ▪ How do trends in programming languages shift over time? What are some emerging languages, and why are they gaining popularity?

4. Features of Java

- **Research Topic:** Dive deep into the features of Java.
  - **Questions to Ponder:**
    - Why is Java considered platform-independent? How does the JVM contribute to this feature?
    - What makes Java robust? Consider features like memory management, exception handling, and type safety. How do these features contribute to its robustness?
    - Why is Java considered secure? Explore features like bytecode verification, automatic garbage collection, and built-in security mechanisms.
    - Analyze other features like multithreading, portability, and simplicity. Why are they important, and how do they impact Java development?

5. Role of public static void main(String[] args) (PSVM)

- **Research Topic:** Analyze the structure and purpose of the main method in Java.
  - **Questions to Ponder:**
    - What is the role of each keyword in public static void main(String[] args)?
    - What would happen if one of these keywords (public, static, or void) were removed or altered? Experiment by modifying the main method and note down the errors.
    - Why is the String[] args parameter used in the main method? What does it do, and what happens if you omit it?

6. Can We Write Multiple main Methods?

- **Research Topic:** Experiment with multiple main methods in Java.
  - **Questions to Ponder:**
    - Can a class have more than one main method? What would happen if you tried to define multiple main methods in a single class?
    - What happens if multiple classes in the same project have their own main methods? How does the Java compiler and JVM handle this situation?
    - Investigate method overloading for the main method. Can you overload the main method with different parameters, and how does this affect program execution?

7. Naming Conventions in Java

- **Research Topic:** Investigate Java's naming conventions.
  - **Questions to Ponder:**
    - Why do some words in Java start with uppercase (e.g., Class names) while others are lowercase (e.g., variable names and method names)?
    - What are the rules for naming variables, classes, and methods in Java, and why is following these conventions important?
    - How do naming conventions improve code readability and maintainability, especially in large projects?

8. Java Object Creation and Memory Management

- **Research Topic:** Understand Java's approach to objects and memory.

- o **Questions to Ponder:**
  - Why are Java objects created on the heap, and what are the implications of this?
  - How does Java manage memory, and what role does the garbage collector play?
  - What are the differences between method overloading and method overriding in Java?
  - What is the role of classes and objects in Java? Explore how they support the principles of object-oriented programming (OOP), such as encapsulation, inheritance, and polymorphism.

9. Purpose of Access Modifiers in Java

- **Research Topic:** Explore the purpose of access modifiers in Java.
  - o **Questions to Ponder:**
    - What is the purpose of access modifiers (e.g., public, private) in controlling access to classes, methods, and variables?
    - How do access modifiers contribute to encapsulation, data protection, and security in object-oriented programming?
    - How do access modifiers influence software design and maintenance?

- Consider potential challenges or limitations of automatic memory management.