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.

Snippet 1:

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

ANS: In the program, the main method is incorrectly defined. It should be public static void main(String[] args), but the static keyword is missing.

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

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

Snippet 2:

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

ANS: Missing public keyword in 2 line

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

What happens when you compile and run this code?

Snippet 3:

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

ANS: In main method we have to put void in upper int will be there also in 2nd last line return 0; is invalid because void is not return value.

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

        System.out.println("Hello, World!");
        }
```

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

Snippet 4:

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

ANS: In mainmethod string [] args is missing
String []args needed because it used to pass command line argument to program

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

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

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");
   }
}
```

ANS: No Error
Yes, we overload the main method defining multiple method with same name but different parameter types

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

Snippet 6:

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

ANS: In Upper code y is not declared
In java every variable must be declared before it is used.if we not declare so compiler doesn't recognize it.

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

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

Snippet 7:

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

ANS: error: incompatible types: String cannot be converted to int int x = "Hello";
In java every variable must be assigned a value of the correct data type and in code hello is a string but x is declare as int

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

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

Snippet 8:

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

ANS: In System.out.println("Hello, World!") the parenthesis is missing it error form because java expects the method call to complete.

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

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

Snippet 9:

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

ANS: In java class is not used as variable name used for define a class. java didn’t use keywords as variable name because it would create confusion for compilier

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

- **What error occurs? Why can't reserved keywords be used as identifiers?**

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);
   }
}
```

ANS: It not run because display() is being call inside the main and main() is a static method & display()is non static.

Method overloading is 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();
        obj.display();
        obj.display(5);
    }
}
```

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

Snippet 11:

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

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

at Main.main(Main.java:3)

It is occur because array contain only 3 element in java we calculate index as 0,1,2 and in sop statement we try to access 5 index which does not exist so it give arrayindexoutof boundsexcepiton.

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

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

Snippet 12:

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

ANS:

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

It run in infinite loop because the condition in while is true and it continues print infinite loop without stop we have to terminate it by control c.

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

Snippet 13:

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

ANS: It throw NullPointerException at runtime.it occur because string is to be null and we call str.leanght() to access method on null which not allow in java.

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

- **What exception is thrown? Why does it occur?**

Snippet 14:

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

ANS: Exception in thread "main" java.lang.NullPointerException: Cannot invoke "String.length()" because "<local1>" is null
at Main.main(Main.java:2)
1 error

In java variables have the right kind of value to avoid mistakes. This helps to run program fast we found many errors while write the code for this make it easy to fix before the program

```
public class Main {
    public static void main(String[] args) {
        double num = 10.5;
        System.out.println(num);
    }
}
```

- **What compilation error occurs? Why does Java enforce data type constraints?**

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);
   }
}
```

Main.java:4: error: incompatible types: possible lossy conversion from double to int
int result = num1 + num2; System.out.println(result);

in this we have to convert double to int for remove decimal.

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

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

Snippet 16:

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

ANS:

After Running code we get 2.0 output but we have to 2.5 the problem is that two whole no remover decimal before save the result

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

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

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);
   }
}
```

ANS: Main.java:4: error: illegal start of expression
int result = a ** b; System.out.println(result);
^
1 error
error: compilation failed

Java is not support to finding power of no.this symbol is not valid in java it in java only support +,-,*,/,%

```
public class Main {
   public static void main (String[] args) {
      int a = 10;
      int b = 5;
      int result = 1;

      for (int i = 0; i < b; i++) {
         result *= a;
      }
      System.out.println(result);
   }
}
```

o/p: 100000

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

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);
   }
}
```

ANS: o/p: 20
In Java, *, /, and % are done before + and -.
So, b * 2 is calculated first, then added to a.

- **What is the output of this code? How does operator precedence affect the 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);
   }
}
```

ANS: java not allow integer divide by zero so it give arithmeticException to incorrect behavior this error occur in runtime because compiler not check for divide by zero

```
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at Main.main(Main.java:4)
```

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

Snippet 20:

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

ANS: error: ';' expected this error is occured

```
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?**

Snippet 21:

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

ANS:

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

}

}
```

In mismatched brace the compiler will throw error because it expect {to closing }

- **What does the compiler say about mismatched braces?**

Snippet 22:

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

ANS: illegal start of expression is error form.
The method is not declared inside another method because it not allow method declaration inside another mehtod

```
public class Main {

  static void displayMessage() {
    System.out.println("Message");
  }

  public static void main(String[] args) {
    displayMessage();
  }
}
```

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

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");
    }
  }
}
```

ANS:
Program is not have break statement if condition match it continue running the next case including default for stop the code we add break statement so program exit the switch after running the match 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?

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("Leve
      l 3"); default:
        System.out.println("Unknown level");
    }
  }
}
```

ANS:

In level 1 we go to case 1:and execute System.out.println("Level 1"); this statement after this there is no break statement so it go to case 2: and then case 3: and finally to default all cases are run until break is found or the switch ends.

Break stop the execution of switch block if case is matched

- **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?

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!");
      }
   }
}
```

```
public class Switch {
   public static void main(String[] args) {
      double score = 85.0;

      if (score == 100.0) {
         System.out.println("Perfect score!");
      } else if (score == 85.0) {
         System.out.println("Great job!");
      } else {
         System.out.println("Keep trying!");
      }
   }
}
```

ANS: Because the switch is not support double and switch only work with char,int,byte
Modification of code: we convert double to int and use if else instead of switch

- **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?

Snippet 26:

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

ANS:

In switch case case must be unique. When we have two identical case labels compiler does not allow same case label in same switch block if 2 case have same value compiler is confusing which have to run and cause error

```
public class Switch {
   public static void main(String[] args) {
      int number = 5;

      switch (number) {
         case 5:
            System.out.println("Number is 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?

.

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"

```
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();

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

Output: Enter the student's score: 85
Grade: B

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.

```
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");

                switchType("Weekday");

                break;

            case 2:

                System.out.println("Tuesday");

                switchType("Weekday");

                break;

            case 3:

                System.out.println("Wednesday");

                switchType("Weekday");

                break;

            case 4:

                System.out.println("Thursday");

                switchType("Weekday");

                break;
```

```
            case 5:
                System.out.println("Friday");
                switchType("Weekday");
                break;
            case 6:
                System.out.println("Saturday");
                switchType("Weekend");
                break;
            case 7:
                System.out.println("Sunday");
                switchType("Weekend");
                break;
            default:
                System.out.println("Invalid input! Please enter a number between 1 and 7.");  }
        scanner.close(); }
    public static void switchType(String type) {
        switch (type) {
            case "Weekday":
                System.out.println("It is a weekday.");
                break;
            case "Weekend":
                System.out.println("It is a weekend.");
                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 if-else to check if division by zero is attempted and display an error message.

```
import java.util.Scanner;
public class SimpleCalculator {
   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 an operator (+, -, *, /): ");
      char operator = scanner.next().charAt(0);
      System.out.print("Enter second number: ");
      double num2 = scanner.nextDouble();
      double result = 0;
      boolean validOperation = true;

  switch (operator) {
         case '+':
            result = num1 + num2;
            break;
         case '-':
            result = num1 - num2;
            break;
         case '*':
            result = num1 * num2;
            break;
         case '/':

    if (num2 == 0) {
                System.out.println("Error! Division by zero is not allowed.");
                validOperation = false;
            } else {
                result = num1 / num2;      }
            break;
         default:
            System.out.println("Invalid operator! Please enter +, -, *, or /.");
            validOperation = false; }
      if (validOperation) {
         System.out.println("Result: " + result); }
      scanner.close(); }}
```

OUTPUT:

```
Enter first number: 10
Enter an operator (+, -, *, /): +
Enter second number: 5
Result: 15.0
```

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.

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

```
import java.util.Scanner;
public class DiscountCalculator {

   public static double calculateDiscount(double totalAmount, boolean hasMembership) {
      double discount = 0;

      if (totalAmount >= 1000) {
         discount = 20;
      } else if (totalAmount >= 500) {
         discount = 10;
      } else {
         discount = 5;
      }
      if (hasMembership) {
         discount += 5;
      }
      double discountAmount = (totalAmount * discount) / 100;
      double finalPrice = totalAmount - discountAmount;

      System.out.println("Total Purchase Amount: Rs." + totalAmount);
      System.out.println("Applied Discount: " + discount + "%");
      System.out.println("Discount Amount: Rs." + discountAmount);
      System.out.println("Final Price After Discount: Rs." + finalPrice);
      return finalPrice;
   }
   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? (true/false): ");
      boolean hasMembership = scanner.nextBoolean();

      calculateDiscount(totalAmount, hasMembership);

      scanner.close();
   }
}
```

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.

```
import java.util.Scanner;
public class StudentPassFail {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter marks for Subject 1: ");
        int subject1 = scanner.nextInt();
        System.out.print("Enter marks for Subject 2: ");
        int subject2 = scanner.nextInt();
        System.out.print("Enter marks for Subject 3: ");
        int subject3 = scanner.nextInt();
        int failedSubjects = 0;

        if (subject1 < 40) failedSubjects++;
        if (subject2 < 40) failedSubjects++;
        if (subject3 < 40) failedSubjects++;

        switch (failedSubjects) {
            case 0:
                System.out.println("Congratulations! You have passed.");
                break;
            default:
                System.out.println("You have failed in " + failedSubjects + " subject(s).");
                switch (failedSubjects) {
                    case 1:
                        System.out.println("You need to improve in one subject.");
                        break;
                    case 2:
                        System.out.println("You need to improve in two subjects.");
                        break;
                    case 3:
                        System.out.println("You need to improve in all three subjects.");
                        break; }
                break; }

        scanner.close();}}
```

OUTPUT: Enter marks for Subject 1: 35
Enter marks for Subject 2: 39
Enter marks for Subject 3: 50
You have failed in 2 subject(s).
You need to improve in two subjects.

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.

    - **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.
    - **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.
-