ASSIGNMENT NO: 02

Snippet 1:

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

- What error do you get when running this code?

```
C:\Users\91951\OneDrive\Desktop\dac\Assignment2>javac Snippet.java

C:\Users\91951\OneDrive\Desktop\dac\Assignment2>java Snippet
Error: Main method not found in class Snippet, please define the main method as:
   public static void main(String[] args)
or a JavaFX application class must extend javafx.application.Application
```

Correct Code:

```
public class Snippet{

      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?

```
C:\Users\91951\OneDrive\Desktop\dac\Assignment2>javac Snippet.java

C:\Users\91951\OneDrive\Desktop\dac\Assignment2>java Snippet
Error: Main method not found in class Snippet, please define the main method as:
   public static void main(String[] args)
or a JavaFX application class must extend javafx.application.Application
```

Snippet 3:

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

- What error do you encounter?

```
C:\Users\91951\OneDrive\Desktop\dac\Assignment2>javac Snippet.java

C:\Users\91951\OneDrive\Desktop\dac\Assignment2>java Snippet
Error: Main method not found in class Snippet, please define the main method as:
   public static void main(String[] args)
or a JavaFX application class must extend javafx.application.Application
```

- Why is void used in the main method?
  It signifies that the method does not return any value. This is because once the main method finishes execution, the program ends, and there is no further action that can be taken with a return value.

Snippet 4:

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

- What happens when you compile and run this code?

```
C:\Users\91951\OneDrive\Desktop\dac\Assignment2>javac Snippet.java

C:\Users\91951\OneDrive\Desktop\dac\Assignment2>java Snippet
Error: Main method not found in class Snippet, please define the main method as:
   public static void main(String[] args)
or a JavaFX application class must extend javafx.application.Application
```

- Why is String[] args needed?

  This is an array of String objects, which stores the command-line arguments passed to the program. The name args is a convention, and you can use any other name.

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");
   }
}
```

- Can you have multiple main methods?

  Yes, Java allows multiple main methods in a single class but main methods must be overloaded versions of the original main i.e. methods have the same name but different parameter lists.

- What do you observe?

```
C:\Users\91951\OneDrive\Desktop\dac\Assignment2>javac Snippet.java

C:\Users\91951\OneDrive\Desktop\dac\Assignment2>java Snippet
Main method with String[] args
```

Snippet 6:

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

- What error occurs?

```
C:\Users\91951\OneDrive\Desktop\dac\Assignment2>javac Snippet.java
Snippet.java:28: error: cannot find symbol
 int x = y + 10;
         ^
  symbol:   variable y
  location: class Snippet
1 error
```

- Why must variables be declared?

  Variables must be declared because variables in a program helps ensure that the code is robust, safe, and efficient. It enables the compiler to enforce type safety, manage memory effectively, and improve the overall maintainability and clarity of the code.

Snippet 7:

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

- What compilation error do you see?

```
C:\Users\91951\OneDrive\Desktop\dac\Assignment2>javac Snippet.java
Snippet.java:34: error: incompatible types: String cannot be converted to int
     int x = "Hello";
             ^
1 error
```

- Why does Java enforce type safety?

  Java enforces type safety to ensure that operations and data manipulations are performed only on compatible data types, which helps prevent many common programming errors and enhances the robustness, reliability, and security of Java programs.

Snippet 8:

```
public class Main {

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

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

```
C:\Users\91951\OneDrive\Desktop\dac\Assignment2>javac Snippet.java
Snippet.java:40: error: ')' or ',' expected
    System.out.println("Hello, World!"
                                      ^
1 error
```

Snippet 9:

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

- What error occurs?

```
C:\Users\91951\OneDrive\Desktop\dac\Assignment2>javac Snippet.java
Snippet.java:44: error: not a statement
    int class = 10;
    ^
Snippet.java:44: error: ';' expected
    int class = 10;
       ^
Snippet.java:44: error: <identifier> expected
    int class = 10;
              ^
Snippet.java:45: error: illegal start of expression
    System.out.println(class);
                       ^
Snippet.java:45: error: <identifier> expected
    System.out.println(class);
                            ^
5 errors
```

- Why can't reserved keywords be used as identifiers?

  Reserved keywords cannot be used as identifiers such as variable names, method names, class names, etc because these keywords have predefined meanings and roles.

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?

```
C:\Users\91951\OneDrive\Desktop\dac\Assignment2>javac Snippet.java
Snippet.java:55: error: non-static method display() cannot be referenced from a static context
 display();
 ^
Snippet.java:56: error: non-static method display(int) cannot be referenced from a static context
 display(5);
 ^
2 errors
```

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?

```
C:\Users\91951\OneDrive\Desktop\dac\Assignment2>javac Snippet.java

C:\Users\91951\OneDrive\Desktop\dac\Assignment2>java Snippet
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 5 out of bounds for length 3
        at Snippet.main(Snippet.java:61)
```

- Why does it occur?

  Because it is out of the length of assign array length.

Snippet 12:

```
public class Main {

      while (true) {
         System.out.println("Infinite Loop");
      }
   }
}
```

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

  WE can avoid infinite loops by ensuring that your program runs as intended and doesn't get stuck in a loop that never ends .

Snippet 13:

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

- What exception is thrown?

```
C:\Users\91951\OneDrive\Desktop\dac\Assignment2>javac Snippet.java

C:\Users\91951\OneDrive\Desktop\dac\Assignment2>java Snippet
Exception in thread "main" java.lang.NullPointerException: Cannot invoke "String.length()" because "<local1>" is null
        at Snippet.main(Snippet.java:73)
```

- Why does it occur?

  This occurs in the code because the variable str is declared as a String but is initialized to null

Snippet 14:

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

- What compilation error occurs?

```
C:\Users\91951\OneDrive\Desktop\dac\Assignment2>javac Snippet.java
Snippet.java:77: error: incompatible types: String cannot be converted to double
 double num = "Hello";
              ^
1 error
```

- Why does Java enforce data type constraints?

  To ensure type safety, which is a critical aspect of the language's design.

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?

```
C:\Users\91951\OneDrive\Desktop\dac\Assignment2>javac Snippet.java
Snippet.java:84: error: incompatible types: possible lossy conversion from double to int
 int result = num1 + num2;
                   ^
1 error
```

- How should you handle different data types in operations?
  Handling different data types in operations requires careful consideration to ensure that operations are performed correctly and that type-related errors are avoided like type casting , handling different datatype, string concatenation etc.

Snippet 16:

```
public class Main {

      int num = 10;
double result = num / 4;
      System.out.println(result);
   }
}
```

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

```
C:\Users\91951\OneDrive\Desktop\dac\Assignment2>javac Snippet.java

C:\Users\91951\OneDrive\Desktop\dac\Assignment2>java Snippet
2.0
```

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?

```
C:\Users\91951\OneDrive\Desktop\dac\Assignment2>javac Snippet.java
Snippet.java:97: error: illegal start of expression
 int result = a ** b;
                  ^
1 error
```

- Why is the ** operator not valid in Java?

  The ** operator is not valid in Java because Java does not have an exponentiation operator

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?

```
C:\Users\91951\OneDrive\Desktop\dac\Assignment2>java Snippet
20
```

- How does operator precedence affect the result?

  Operator precedence determines the order in which operators are evaluated in an expression. When multiple operators are used in a single expression, precedence rules dictate which operations are performed first.

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?

```
C:\Users\91951\OneDrive\Desktop\dac\Assignment2>java Snippet
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at Snippet.main(Snippet.java:111)
```

- Why does division by zero cause an issue in Java?

  Because in Java division by zero causes an issue because it's mathematically undefined

Snippet 20:

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

- What syntax error occurs?

```
C:\Users\91951\OneDrive\Desktop\dac\Assignment2>javac Snippet.java
Snippet.java:116: error: ';' expected
 System.out.println("Hello, World")
                                   ^
1 error
```

- How does the missing semicolon affect compilation?

  A missing semicolon is a common syntax error that affects compilation because the semicolon (;) is used to terminate statements.

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?

```
C:\Users\91951\OneDrive\Desktop\dac\Assignment2>javac Snippet.java
Snippet.java:121: error: reached end of file while parsing
}
 ^
1 error
```

Snippet 22:

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

- What syntax error occurs?

```
C:\Users\91951\OneDrive\Desktop\dac\Assignment2>javac Snippet.java
Snippet.java:123: error: illegal start of expression
 static void displayMessage() {
 ^
Snippet.java:127: error: class, interface, enum, or record expected
}
^
2 errors
```

- Can a method be declared inside another method?

  Method cannot declare directly inside another method. Methods must be declared within the body of a class or an interface, not within other methods. However, we can define methods within methods indirectly

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"?

  Because in the switch statement, the default case prints after "Value is 2" because if how the condition failure works in switch-case statements.

  How can you prevent the program from executing the default case?

To prevent the default case from executing in a switch statement, we need to control the flow of execution by ensuring that once a match is found and the corresponding case block is executed, we avoid falling through to the default case. This can be achieved using the break statement.

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"?

The reason "Level 1", "Level 2", "Level 3", and "Unknown level" are all printed is due to the absence of break statements. Without break statements, once a case matches, the execution continues through all subsequent cases and the default case, resulting in a fall-through effect.

What is the role of the break statement in this situation?

The role of break statement in this situation is to ensure that after the matching case is executed the program exit the switch block.

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?

The code does not compile because switch statements do not support double types for the switch expression.

What does the error tell you about the types allowed in switch expressions?

The error tell that the switch statement requires the switch expression to be of a type that can be evaluated to a specific case label.

How can you modify the code to make it work?

```
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!");

      }

   }

}
```

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 to Investigate:

Why does the compiler complain about duplicate case labels?

The compiler complains about duplicate case labels in a switch statement because each case label within a switch block must be unique.

What happens when you have two identical case labels in the same switch block?

When we have two identical case labels in the same switch block, the compiler will not compile the code and will issue an error.

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"

CODE:

```
class Grade{

    public static void main(String args[]){


        int marks = 85;
```

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

OUTPUT:

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.

```
class Days{

public static void main(String args[]){



        int day = 4;



        switch(day){

                case 1 :

                    System.out.println("Today is Monday");

                            break;

           case 2 :

                    System.out.println("Today is Tuesday");

                            break;

           case 3 :

                    System.out.println("Today is Wednesday");

                            break;

                case 4 :

                    System.out.println("Today is Thursday");
```

```
                    break;
                case 5 :
                    System.out.println("Today is Friday");
                    break;
            case 6 :
                    System.out.println("Today is Saturday");
                    break;
            case 7 :
                    System.out.println("Today is Sunday");
                    break;
            case 8 :
                    System.out.println("Invalid Day");
                    break;

          }
}
}
```

OUTPUT:

Today is Thursday

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.

CODE:

```
import java.util.Scanner;

class Calculator{

public static void main(String args[]){

        Scanner sc = new Scanner(System.in);

        double num1 , num2 , result;

        char operator;

        System.out.println("Enter first number : ");

        num1 = sc.nextDouble();

        System.out.println("Enter Operator : ");

        operator = sc.next().charAt(0);

        System.out.println("Enter second number : ");

        num2 = sc.nextDouble();

        if(operator == '+'){

                result = num1 + num2;

        }else if(operator == '-'){

                result = num1 - num2;

        }else if(operator == '*'){
```

```
                result = num1 * num2;
        }else if(operator == '/'){
                if(num2 != 0){
                        result = num1 / num2;
                }else{
                        System.out.println("Error: Division by zero not allowed");
                        return;
                }
        }else{
                System.out.println("Invalid Operator");
                return;
        }
        System.out.println("Result : " + result);
}
}
```

OUTPUT:

```
Enter first number :
12
Enter Operator :
/
Enter second number :
3
```

Result : 4.0

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.

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

CODE:

```
class Discount{

public static void main(String args[]){



        int price = 5000;

        double total;

        boolean membership = true;



       if(price>=1000){

              if(membership == true){

                     total = price * 0.25;

                     System.out.println("Total : " + total);

              }else{

                     total = price * 0.2;

                     System.out.println("Total : " + total);
```

```
            }
        }else if(price>=500 && price<=999){
            if(membership == true){
                total = price * 0.15;
                System.out.println("Total : " + total);
            }else{
                total = price * 0.1;
                System.out.println("Total : " + total);
            }
        }else if(price<500){
            if(membership == true){
                total = price * 0.1;
                System.out.println("Total : " + total);
            }else{
                total = price * 0.05;
                System.out.println("Total : " + total);
            }
        }
}
}
```

OUTPUT:

```
Total : 1250.0
```

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.

CODE:

```
class PassFail{

public static void main(String args[]){

        int sub1 = 45, sub2 = 21, sub3 = 25, failcount = 0;

        String result;

        if (sub1>=40 && sub2>=40 && sub3>=40){

                result = "pass";

        }

        else

        {

                result = "fail";

        }


        switch(result){

                case "pass": System.out.println("Student has passed all 3 subjects.");

                break;
```

```
case "fail":
if(sub1<40){
        failcount++;
}
if(sub2<40){
        failcount++;
}
if(sub3<40){
        failcount++;
}
switch(failcount){
                case 1: System.out.println("Student has failed in 1 subject.");
                break;
                case 2: System.out.println("Student has failed in 2 subjects.");
                break;
                case 3: System.out.println("Student has failed in 3 subjects.");
                break;
}
break;
```

```
        }

}

}
```

OUTPUT:

Student has failed in 2 subjects.