Snippet 1:

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

Q]. What error do you get when running this code?

A]. In the given snippet the main method is declare in a wrong way.Its method signature is different.

In the give snippet’s main method, void keyword is missing. Keyword void is the return type.

Keyword clarifies that method is not going to return anything , method will return NOTHING.

Corrected Snippet 1:

```
public class Main1 {
    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!");
 }
}
```

Q]. What happens when you compile and run this code?

A]. When we compile this code snippet we get error stating as: “Main method not found in class Main, please define the main method”. In the snippet access modifier-public is missing. Due to which compile is not able to find the correct main method.

Corrected Snippet 2:

```
              public class Main2 {
              public static void main(String[] args) {
              System.out.println("Hello, World!");
    }}
```

Snippet 3:

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

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

A] When we compile this code snippet we encounter an error stating-

Error: Main method must return a value of type void in class Main”

This is because in the snippet the main method’s return type is int and its retuning 0.

Returning 0 and void are total different methods. As the main method is entry point of program,so it must be only executed and not to return any value.

Corrected Snippet 3:

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


   }
}
```

……

Snippet 4:

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

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

A].When we compile this code we get error stating as:

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

```
   public static void main(String[] args)
```

Here we get error as the parameter of main method-String[] args is missing

String[] args allows the program to accept command-Line arguments. This helps to pass the argument while compiling the code.

Corrected Snippet 4:

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

…….

Snippet 5:

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

Q]. Can you have multiple main methods? What do you observe?

A]. Yes we can have multiple main methods. But there is one codition->this main methods must be having different parameter i.e they must be overloaded.

When we compile this code we get o/p as Main method with String[] args. As the compiler looks for the main method with String[] parameter.

There is no error in snippet.

………

Snippet 6:

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

Q]. What error occurs? Why must variables be declared?

A].We get error as-> Cannot find symbol ‘y’. Before using or accessing the variable we must declare the variable.Declaring the variable means getting the memory location for the symbol.So before using the variable it must have the memory location.Here the y variable must be declare and initialized.

Corrected Snippet 7:

```
public class Main {
    public static void main(String[] args) {
    int y=0;
    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);
 }
}
```

Q]. What compilation error do you see? Why does Java enforce type safety?

A].When we compile this snippet we get error stating that-> error: incompatible types: String cannot be converted to int

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

This error occurs as we are trying to to covert String into the intger.

Java does not allow us to this type of conversions since this are safety measure for not losing the data.As the conversion might lead to lose some data and the output might get affected and therefore we get error.

Corrected snippet 7:

```
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!"

 }

}
```

Q]. What syntax errors are present? How do they affect compilation?

A]. Here the closing bracket of print method is missing. Java is strictly typed checked language. Therefore it does not get compiled until and unless all syntax is not checked and corrected by programmer.

Corrected Snippet 8:

```
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);

 }

}
```

Q]. What error occurs? Why can't reserved keywords be used as identifiers?

A]. There is a syntactical error in snippet. Here keyword ‘class’ is used as identifier.We cannot use keyword as identifier.We cannot use it because if we do it, java will not able to differentiate the keyword and identifier it will get confused.To avoid this confusion its is stricticly prohibited to use it as identifier.

……

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);
 }
}
```

Q]. What happens when you compile and run this code? Is method overloading allowed?

A].When we compile this snippet we get error.In the snippet ,display() is overloaded which is allowed.There's a error regarding static and non static context.We cannot reference from static method.Also the overloaded methods are instance method they must be called by creating instance,

Corrected 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) {
        Main obj = new Main();
        obj.display();

        obj.display(5);

    }

}
```

……….

Snippet 11:

```
public class Main {

 public static void main(String[] args) {

 int[] arr = {1, 2, 3};

 System.out.println(arr[5]);

 }

}
```

Q]. What runtime exception do you encounter? Why does it occur?

A]. Here the array is declare of size three,And we are accessing the 5th element ,which is not in the bound of array,.There java gives error->”ArrayIndexOutOfBound”.

It means we cannot access the memory which is not specified by the user.We can access only till the index 2 as array starts from 0.

Corrected Snippet 11:

```
public class Main {

 public static void main(String[] args) {

 int[] arr = {1, 2, 3};

 System.out.println(arr[2]);

 }

}
```

………

Snippet 12:

```
public class Main {

 public static void main(String[] args) {

 while (true) {

 System.out.println("Infinite Loop");

 }

 }
```

Q]. What happens when you run this code? How can you avoid infinite loops?

A].When we run this code it runs into infinite loop. This happens because while loop gets executed every time because its condition is always true.To avoid infinite loop there are many ways :

->Check the Loop Condition

->Update Variables Inside the Loop

->Use a Break Statement

->Set a Maximum Iteration Limit

…….

Snippet 13:

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

Q]. What exception is thrown? Why does it occur?

A].When we run this it will throw a NullPointerException.It occurs because we are trying to print it length null value. String it initialize with null value. We cannot get length of null. Therfore it throws null exception.

…….

Snippet 14:

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

Q].What compilation error occurs? Why does Java enforce data type constraints?

A].When we run this snippet it will give error.It will give error: incompatible types: String cannot be converted to double

```
        double num = "Hello";
```

This error occurs dues to type safety.Java is strictly typed language.It check each and every syntax.

It doesnot allows this because we are trying to convert string into double value.This might cause loss in data,which lead to different output.Therefore java does not allows it.

Correct Snippet 14:

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

……….

Snippet 15:

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

Q].What error occurs when compiling this code? How should you handle different data types

in operations?

A].This snippet will give error: incompatible types: possible lossy conversion from double to int

```
        int result = num1 + num2;
```

Java does not allows to convert the data types into lower datatypes.Here we are trying to convert int into double.It will cause into data loss.We can handle this types conversion :

-> Use the Appropriate Type:Use the data of same datatype.

-> Or we can perform explicit typecast:First typecast the data and then use it for operation.

Corrected Snippet 15:

```
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);

    }

}
```

Snippet 16:

```
public class Main {

 public static void main(String[] args) {

 int num = 10;

 double result = num / 4;

 System.out.println(result);

 }

}
```

Q].What is the result of this operation? Is the output what you expected?

A]. This snippet will give output of 2.0. Yes we get expected output as ‘/’ operator retrun the Quotient part only.

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);

 }

}
```

Q].What compilation error occurs? Why is the ** operator not valid in Java?

A].This snippet gives error: illegal start of expression

```
        int result = a ** b;
```

The error occurs because ** is not a valid operator in Java. Java uses specific operators for arithmeticoperations, but it does not include ** as one of them.Instead we can use Math.pow().

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);
 }
}
```

Q]. What is the output of this code? How does operator precedence affect the result.

A].This snippet will give output as 20.When there are multiple operstors they are evaluated according To the precedence assign to them.They are evaluated according to BODMAS rule.

……

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);
 }
}
```

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

A].This snippet will throw the runtime exception:

```
Exception in thread "main" java.lang.ArithmeticException: / by zero
```

In mathematics, division by zero is undefined because there is no number that you can multiply by 0. Java specifically handles this scenario by throwing an ArithmeticException when an integer division by zero occurs.

…….

Snippet 20:

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

Q]. What syntax error occurs? How does the missing semicolon affect compilation?

A].There is syntax error.Semicolon is missing after the println statement.

In Java, each statement must end with a semicolon (;). The semicolon is used by the compiler to recognize the end of a statement.Due to the missing semicolon, the Java compiler cannot correctly.

It expects a semicolon to mark the end of the System.out.println("Hello, World") statement.

As a result, the compiler generates a syntax error and fails to compile the code. The program will not run until this error is corrected.

Corrected Snippet 20:

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

……

Snippet 21:

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

Q].What does the compiler say about mismatched braces?

A]. error: reached end of file while parsing

The compiler expects a closing brace } to match the opening brace { for the main method.

Since it doesn't find this closing brace, it reports that it reached the end of the file.

Corrected snippet 21:

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

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

……

Snippet 22:

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

Q]. What syntax error occurs? Can a method be declared inside another method?

A].It will give error :

```
error: illegal start of expression
        static void displayMessage() {
               ^
```

In Java, you cannot declare a method inside another method. Methods in Java must be declared a the class level, not inside other methods. The displayMessage() method is incorrectly placed inside the main method, which is not allowed.

Corrected Snippet 22.

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

    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");
 }
 }
}
```

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

A]. It will give output as

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

Since there is no break statement after case 2, execution will go to the next case, which is case 3, and then to the default case.To avoid this we must use break statement after every case,so that it execute the case and will break the switch case.

Corrected Snippet 23:

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

Q].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?

A].It will give output as:

```
Level 1

Level 2

Level 3

Unknown level
```

The break statement is used to terminate the execution of a switch block once a case has been executed. Without a break statement, all subsequent cases and the default case are executed after the matched case.

Corrected Snippet 24:

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

……

Snippet 25:

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

Q]. 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?

A].It will not get compiled ,it gives error: switch expression must be of type char, byte, short, int, Character, Byte, Short, Integer, Enum, String, or a corresponding wrapper class

```
        switch(score) {
              ^
```

double is not a valid type for a switch expression. This is why the compiler is throwing an error.

Corrected snippet 25:

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

……

Snippet 26:

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

Q]. 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?

A].It will not get compile ,it gives error:

```
error: duplicate case label
            case 5:
                 ^
```

When there are duplicate case labels, the compiler does not know which block of code to execute if the case label matches the switch expression. This leads to confusion and therefore it gives error.

Corrected 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 6:

                System.out.println("This is case 6");

                break;

            default:

                System.out.println("This is the default case");

        }

    }

}
```

………..