Snippet 1

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

What error do you get when running this code?

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

Correct Code:

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

Snippet 2

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

What happens when you compile and run this code?

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

Correct Code:

```
public class Main {
   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;
   }
}
```

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

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

Ans: As main is the entry point function ,the program execution starts from main method and its return type must be void as it do not return any value.

Correct Code: 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!");
    }
}
```

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

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

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

String[] args is needed as it takes command line arguments

Correct Code

```
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");
    }
}
```

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

Yes There can be multiple main methods but not the same as string[] args it can be different like int[] args. Jvm recognize for string[] args to start the execution and if compile and run the problem we get,

```
 output: 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? Why must variables be declared?

```
Main.java:3: error: cannot find symbol
        int x = y + 10;
                ^
  symbol:   variable y
  location: class Main
```

Correct Code

```
public class Main {
    public static void main(String[] args) {
        int y = 2;// Declared y here
        int x = y + 10;
        System.out.println(x);
    }
}
```

Snippet 7

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

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

```
Main.java:3: error: incompatible types: String cannot be converted to int
        int x = "Hello";
```

Correct code

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

Snippet 8

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

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

```
Main.java:3: error: ')' expected
        System.out.println("Hello, World!"
                                           ^
```

Missing Closing parenthesis ‘)’

Correct Code

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

Snippet 9

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

}

What error occurs? Why can't reserve keywords be used as identifiers?

```
Errors: Main.java:3: error: not a statement
        int class = 10;
        ^
Main.java:3: error: ';' expected
        int class = 10;
           ^
Main.java:3: error: <identifier> expected
        int class = 10;
                 ^
Main.java:4: error: <identifier> expected
        System.out.println(class);
                          ^
Main.java:4: error: illegal start of type
        System.out.println(class);
                           ^
Main.java:4: error: <identifier> expected
        System.out.println(class);
                                ^
Main.java:6: error: reached end of file while parsing
}
```

No, reserved Words can’t be used as identifiers as they already have there predefined meaning and then also using them can cause ambiguity.

Correct Code

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

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) {
displa ();
display(5);
}
 }
```

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

```
Error:   Main.java:9: error: non-static method display() cannot be referenced from a static context
        display();
        ^
Main.java:10: error: non-static method display(int) cannot be referenced from a static context
        display(5);
```

Ans: As display() methods are not static they cannot be called from an static context.

Yes. Method overloading is allowed.

Correct Code

```
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 cs = new Main() // Create an instance of Main
 Cs.display();
 Cs.display(5);
 }
 }
```

Snippet 11

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

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

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

Correct Code

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


      System.out.println(arr[5]);
   }else{
System.out.println("Index Out of bound");
}
}
}
```

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?

The code runs Infinite Loops.

We can avoid infinite loops by using a false condition

Correct Code

```
public class Main {
    public static void main(String[] args) {
        int i = 0;
        while (i < 10){
            System.out.println("Loop iteration: " + i);
            i++;
        }
    }
}
```

Snippet 13

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

What exception is thrown? Why does it occur?

```
Exception in thread "main" java.lang.NullPointerException
        at Main.main(Main.java:4)
```

Ans: NullPointerException .It occurred because str is assigned null.

Correct Code:

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

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?

```
Main.java:3: error: incompatible types: String cannot be converted to double
        double num = "Hello";
```

Data type is important as to prevent errors

Correct Code:

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

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?

Error : Main.java:5: error: incompatible types: possible lossy conversion from double to int

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

firstly, data types are very important to perform operations and to get correct output. Choosing right data type and proper understanding of type conversion,Implicit casting and explicitly casting

Correct Code

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

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

Error: Type mismatch

Correct Code

```
public class Main {
    public static void main(String[] args) {
        int num = 10;
        double result = num / 4.0;
        System.out.println(result);
    }
}
```

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?

```
Main.java:6: error: illegal start of expression
        int result = a ** b;
```

** not an valid operator

Correct Code

```
public class Main {
 public static void main(String[] args) {
 int a = 10;
 int b = 5;
 int result = (int)Math.pow(a,b);
System.out.println(result);
}
 }
```

Snippet 18

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

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

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

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

```
Exception in thread "main" java.lang.ArithmeticException: / by zero
	at Main.main(Main.java:5)
```

Ans : Div is not allowed in java

Correct Code

```
public class Main {
   public static void main(String[] args) {
      int a = 10;
      int b = 0;
      if(b != 0){
      int result = a / b;
      System.out.println(result);
   }else{
     System.out.println("Div by zero not allowed");
}
}
}
```

Snippet 20

```
public class Main {
public static void main(String[] args) {
System.out.println("Hello, World") // line 3
}
}
```

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

Ans Syntax Error missing ; at line 3

As every Sentence must end with ; it is a must otherwise it give syntax Error

Correct Code

```
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!");
}
```

What does the compiler say about mismatched braces?

Ans : Error Main.java:5: error: reached end of file while parsing

}

Correct Code

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

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

```
Main.java:3: error: illegal start of expression
        static void displayMessage() {
        ^
Main.java:7: error: class, interface, or enum expected
}
```

No method cannot be declared inside another method

Correct Code

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

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?

```
Main.java:1: error: class Confusion is public, should be declared in a file named Confusion.java
public class Confusion {
```

Correct Code

```
public class Confusion {
   public static void main(String[] args) {
      int value = 2;
      switch(value) {
         case 1:
            System.out.println("Value is 1");
            break;
         case 2:
            System.out.println("Value is 2");
            break;
         case 3:
            System.out.println("Value is 3");
            break;
            default:
                System.out.println("Default case");
        }
    }
}
```

Snippet 24

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

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

Error : Main.java:1: error: class MissingBreakCase is public, should be declared in a file named MissingBreakCase.java

```
public class MissingBreakCase {
```

Correct Code

```
public class MissingBreakCase {
```

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

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?

```
        case 100:
        ^
Main.java:8: error: orphaned case
        case 85:
        ^
Main.java:12: error: orphaned default
        default:
        ^
Main.java:17: error: illegal character: '\u201a'
∩é╖
 ^
Main.java:17: error: illegal character: '\u00b7'
∩é╖
```

Correct Code

```
public class Switch {
   public static void main(String[] args) {
        double score = 85.0;
      if(score == 100) {
            System.out.println("Perfect score!");

          } else if(score == 85) {

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

```
Error: Switch.java:10: error: duplicate case label
         case 5:
```

Correct Code

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