Snippet 1:

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

🕐 What error do you get when running this code?

Error: The main method lacks the static keyword

Explain: Java requires the main method to be declared as static since it serves as the program's entry point and must be callable without creating an object

Fixed 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: The main method lacks the public access modifier

Explain: The JVM requires the main method to be declared as public to be accessible as the program's entry point

Fixed 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: The main method has incorrect return type int instead of void

Explain: Java's main method must be declared with void return type as JVM expects no return value from the program's entry point

Fixed Code:

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

Why is void used in the main method?

Having void makes it clear that the main method's purpose is to execute the program rather than produce a value . Also It's part of Java's standard specification

}

🕐 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!");

}

}
```

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

Error: The main method is missing the required String[] args parameter

Explain: Java's main method must accept a String array parameter to receive command-line arguments. Java Specification

Fixed Code:

```
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!");

}

}
```

🕐 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");

}

}
```

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

Error: While this code will compile but only the method with String[] args will be recognized as the program entry point

Explain: Java allows method overloading but the JVM specifically looks for main(String[] args) as the entry point making the int[] version effectively unreachable as a main method

Fixed Code: Code will work

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

Error: Variable 'y' is not declared before being used in the calculation

Explain: Java requires all variables to be explicitly declared and initialized before use as it is a statically-typed language

Fixed Code:

```
public class Main {
public static void main(String[] args) {
int y = 5;
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?

Error: Incompatible types - cannot convert String to int

Explain: Java enforces strict type checking and prevents assigning a String literal to an integer variable

Fixed Code:

```
public class Main {
public static void main(String[] args) {
String x = "Hello";
int x = 42;
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?

Error: Missing closing parenthesis and missing semicolon in println statement

Explain: Java requires proper syntax with balanced parentheses and semicolons at the end of statements

Fixed 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 reserved keywords be used as identifiers?

Error : class is a reserved keyword and cannot be used as a variable name.

Explain: Java reserves certain words (like class, public, static, etc.) for specific functionalities, so they cannot be used as variable names to avoid conflicts with the language syntax.

Fixed Code: public class Main {

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

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?

Error : Instance methods cannot be called directly inside main(), which is static.

Explain: Marking the methods as static allows them to be accessed without an instance of the class.

Fixed Code:

```
public class Main {
public static void display() {
System.out.println("No parameters");
}
public static void display(int num) {
System.out.println("With parameter: " + num);
}
public static void main(String[] args) {
display();
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 : ArrayIndexOutOfBoundsException at runtime.

Explain: The array arr has only 3 elements (0, 1, 2) but the code tries to access index 5, which is out of bounds causing an exception.

Fixed Code:

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

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

Error : No compilation error, but the program runs infinite loop

Explain: The condition while(true) is always true, so the loop never terminates continuously prints.

Fixed Code:

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

```
System.out.println("Loop running...");
break;
}
}
}
```

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?

Error : NullPointerException at runtime.

Explain: The variable str is assigned null, and calling str.length() on null causes the exception since null does not reference any object.

Fixed Code: Null Check

```
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, cannot get 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?
Error : Incompatible types: String cannot be converted to double

Explain: Java is strongly typed, meaning a String ("Hello") cannot be assigned to a double because they are different data types. Java enforces type safety to prevent unexpected behavior and logical errors

Fixed Code:

```
public class Main {
public static void main(String[] args) {
double num = 10.5;
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);

}

}
```

⏱ What error occurs when compiling this code? How should you handle different data types

in operations?

Error : Incompatible types: possible lossy conversion from double to int

Explain: The expression num1 + num2 results in a double value but it's being assigned to an int variable (result). Java does not allow implicit conversion from double to int because it may lead to data loss (fractional part is lost)

Fixed Code:

```
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);
}
}
```

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

Error : No compilation error, but the output might be unexpected.

Explain: num / 4 performs integer division because num is an int and 4 is an int. This results in 2 (instead of 2.5), which is then stored as 2 in result.

Fixed 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?

Error : error: bad operand types for binary operator '**'

Explain: Java does not support the ** operator for exponentiation

Fixed Code:

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

Error : No compilation error. The code runs correctly.

Explain: Operator precedence follows BODMAS rules

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?

Error : ArithmeticException: / by zero at runtime.

Explain: In Java, dividing an integer by zero (a / b when b = 0) is not allowed because it leads to an undefined mathematical operation, causing an ArithmeticException.

Fixed 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("Error: Division by zero is not allowed.");
}
}
}
```

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?

Error : error: ';' expected

Explain: In Java, every statement must end with a semicolon (;).

Fixed 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!");

// Missing closing brace here
```

}

🕐 What does the compiler say about mismatched braces?

Error : Compiler Error

Explain: The compiler expects a closing brace (}) for the Main class, but it is missing. Java requires properly matched opening { and closing } braces to define blocks of code

Fixed 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?

Error : error: illegal start of expression

Explain: Methods cannot be declared inside other methods.

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

Error : No compilation error

Explain: Case statements do not automatically stop execution unless a break; statement is used.

Fixed 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 : No compilation error

Explain: switch cases do not stop automatically unless a break; statement is used. Without break;, execution continues to the next cases even if they don’t match.

Fixed 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?

Error : error: incompatible types: possible lossy conversion from double to int

Explain: The switch statement only supports certain data types (byte, short, char, int, String, and enum). Floating-point types (float, double) are not allowed because they can have decimal values, making exact case matching unreliable.

Fixed Code:

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

🕐 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 : error: duplicate case label

Explain: Each case label inside a switch statement must be unique

Fixed Code:

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