/*__Snippet 1__: Error Code

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

When compile program it is compile properly without error but when i run the code it give an error following:

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

public static void main(String[] args)

*/

```
//Correct Code
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!");
}
}*/
```

/* When compile program it is compile properly without error but when i run the code it give an error following:

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
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;
}*/
```

/* When compile program it is compile properly without error but when i run the code it give an error following:

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)

2. void doesn't give any return type

*/

```
//correct Code
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?

- When compile program it is compile properly without error but when i run the code it give an error following:

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

Why is String[] args needed?

- String[] args is an array of String objects that represents command-line arguments passed to the Java program when it is executed.

-The args array allows users to pass input data to the program at runtime.

*/

```
//Correct Code
class Test4 {
```

```
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 have multiple main methods but their parameter types has been differents. and
- program is compile and run and show the output of first print line

*/

// program is compile and run and show the output of first System.out.println("Main method with String[] args")

```
//Correct Program
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");
}
}
```

/***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: cannot find symbol

```
int x = y + 10;
      ^
 symbol:   variable y
 location: class Main

 -Why must variables be declared?
 - it will help compiler for check right type value will be assign  and it will hold the value of variable and without value addition is not process.
*/

// Correct code
class Main {
public static void main(String[] args) {
            int y = 1;
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?
- Java enforces type safety to improve program correctness, reliability, and maintainability
-
*/

/*
error: incompatible types: String cannot be converted to int
int x = "Hello";
*/

//Correct Program
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!"
}
}
```

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

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

- copiler is not calculate the things if not close the paranthesis and didn't get termination of statement.

*/

```
//Correct Program
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: not a statement
int class = 10;
^
error: ';' expected
int class = 10;
  ^
```

```
error: <identifier> expected
int class = 10;
      ^
error: <identifier> expected
System.out.println(class);
             ^
error: illegal start of type
System.out.println(class);

error: <identifier> expected
System.out.println(class);

error: reached end of file while parsing
```

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

- reserved keywords cannot be used as identifiers because they have special meaning in the language and are used to define the structure and behavior of the program

*/

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

/***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: non-static method display() cannot be referenced from a static context
display();
^
error: non-static method display(int) cannot be referenced from a static context
display(5);

- Is method overloading allowed?
- Yes method overloading is allow.
*/

//Correct Code
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?

Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 5 out of bounds for length 3
    at Main.main(Test11.java:14)
        - Error will be occur because assign value to array will be 3 and in program will be written to show output of 5 index from array thats why it show out of array value
*/

```
//Correct Program
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?

- it gives the output of infine loop for infinite time and we used termination condition for avoid infinite loops

*/

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

/***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(Test13.java:13)
```

- because null is a special literal but if we add in inverted coma it became a string
*/

//Correct Program
class Main {
public static void main(String[] args) {
String str = "null";
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?
 error: incompatible types: String cannot be converted to double
double num = "Hello";

- Why does Java enforce data type constraints?
- ----------
*/

//Correct Program
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);
}
}
```
 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

```
int result = num1 + num2;
```
 How should you handle different data types in operations?

- Using type Conversion from just add int in bracket before num2

```
*/

//Correct Program
class Main {
public static void main(String[] args) {
int num1 = 10;
double num2 = 5.5;
int result = num1 + (int)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?

2.0 , No

```
*/

//Correct Program
class Main {
public static void main(String[] args) {
int num = 10;
double result = num / 4;
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: illegal start of expression
int result = a ** b;
        - ** operator is not valid because Java does not support exponentiation
*/

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

- output : 20;

How does operator precedence affect the result?
- in program BODOMAS Rule will be follow
*/

//Correct Program
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(Test19.java:18)

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

//Correct Program
class Main {
public static void main(String[] args) {
int a = 10;
int b = 0;
int result = a / b;
System.out.println(result);
}

```
}
```

/***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: ';' expected
System.out.println("Hello, World")
                        ^
```

 How does the missing semicolon affect compilation?

- it affect on compilation because compiler doesnot not know termination of statement.

*/

```
//Correct Program
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: reached end of file while parsing
}
```

*/

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

}

/***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: illegal start of expression
static void displayMessage() {
^
error: class, interface, or enum expected
```

- No , method can not be declared inside another method

```
}
*/

//Correct Program
class Main {
public static void main(String[] args) {
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");
}
}
}
```
 Error to Investigate: Why does the default case print after "Value is 2"? How can you prevent the program from executing the default case?

```
*/

//Correct Program
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");
}
}
}
```

/***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?

- Because there is not break statement thats why all case statement run and print output. if break statement is there then after run it directly come out from switch statement
*/

```
//Correct Program
class Main {
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");
}
}
}
```

/***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?

- incompatible types: possible lossy conversion from double to int
switch(score) {
- in switch statement long, double,float is not allowed
*/

//Correct Program
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: duplicate case label
case 5:
*/

```
//Correct Program
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 another case 5");
break;
default:
System.out.println("This is the default case");
}
}
}
```