**Snippet 1:**

public class Main {

public void main(String[] args) {

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

}

}

**What error do you get when running this code?**

**1.Error**

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

public static void main(String[] args)

**2.explanation**

**-**Main method doesn’t have static keyword

**3.Fix**

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?**

**1.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

**2.explanation**

**-**Main method doesn’t have public keyword

**3.Fix**

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?1.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.explanation**

**-** The main method in uses the void keyword because it does not return any value. However, if the main method is declared with int instead of void, it would suggest that a return value is expected, which is not the standard convention in Java

**3.Fix**

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?**

**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

**Explanation :**

The code compiles but In main method (String args[]) is missing which causes the error.

(String args[])

**Fix:**

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?**

**Explantaion:**

There are two main methods with different parameters like in 1st main method it has(String args[]) and 2nd method has(int args[]).However if there are methods with same identifier ,it will show the overloaded one

**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

**Explanation:**

he primary purpose of declaring variables is to store necessary data in memory, allowing us to use them in our program to perform various operations or tasks.

**Fix:**

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: String cannot be converted to int

int x = "Hello";

**Explantion:**

Java enforces type safety to prevent memory management issues

**Fix :**

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!"

}

}

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

**Error:**

1:-')' expected

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

2:- ';' expected

System.out.println("Hello, World!")

**Explanation:**

The right parenthesis () is essential for closing method calls, method declarations, and control structures, ensuring the program runs successfully. Similarly, a semicolon is crucial for terminating statements

**Fix:**

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:**

Not a statement

int class = 10;

**Explanation:**

Keywords are predefined, reserved identifiers that have special meanings to the compiler.

**Fix:**

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) {

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();

-non-static method display(int) cannot be referenced from a static context

display(5);

**Explanation:**

In Java, overloading a static method refers to defining multiple methods with the same name but different parameter lists within the same class or subclass.

**Fix:**

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:**

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

Explanation:

Because the array size in system.out.println() is greater than the array elements in int[] arr.

**Fix:**

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?**

**Explanation:**

It will go into infinite loop. Using proper loop conditions we can avoid infinite loops.

**Fix:**

public class Main

{

public static void main(String[] args) {

while (true) {

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

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**?

**Exception:**

Exception in thread "main" java.lang.NullPointerException: Cannot invoke "String.length()"

**Explanation:**

Since `str` is null, it doesn't point to any real object in memory. When Java tries to use `length()` on it, there's nothing to refer to, which causes a `NullPointerException`.

**Fix:**

public class Main

{

public static void main(String[] args)

{

String str = "hi";

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";

**Explanation:**

Java enforces type constraints to prevent unbound access to memory.

**Fix:**

public class Main

{

public static void main(String[] args) {

double num = 3.75;

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;

**Explanation:**

By using **type casting** we can handle different data types in operations.

**Fix:**

public class Main

{

public static void main(String[] args)

{

int num1 = 10;

double num2 = 5.5;

double result = (double)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?**

**Output:**

2.4

**Explanation:**

Yes.

**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;

**Explanation:**

n Java, the \*\* operator doesn’t exist.

**Fix:**

public 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.

**Explanation:**

Operators of higher precedence (\*,/) will be evaluated first followed by lower precedence (+, -).

**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

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

**Explanation:**

We cannot divide interger by zero mathematically.but we can handle the exceptions

**Fix:**

public class Main

{

public static void main(String[] args)

{

int a = 10;

int b = 2;

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!")

Explanation :

Semicolon is impotant for terminating 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?**

**Error :** reached end of file while parsing

}

^

**Fix:**

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?**

**1**. illegal start of expression

static void displayMessage() {

**2** .class, interface, or enum expected

}

**Explanation :No.** Method can not be declared under another method.

* **Fix:**

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?**

**Explanation** : 1. Without break keyword the execution will continue

2. By using 'Break' statement .

**Fix** : 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?**

**Explanation :** 1. Because the switch case will continue executing all cases

2. Break statement ensures to exit after a condition is satisfied

.

**Fix :** 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?**

**Explanation :**

1. incompatible types: possible lossy conversion from double to int

switch(score) {

2. The code fails to compile because switch statements in Java do not support double (floating-point) values as the expression type.

**Fix** :

public class Switch {

public static void main(String[] args) {

int score = 85; // Changed from double to int

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:

^

**Explanation :** code contains two case 5: labels, which is not allowed in Java.

**Fix:**

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 another case 5");

break;

default:

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

}

}

}