Snippet 1:

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

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

}

}

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

public static void main(String[] args)

snippet 2:

public class Main {

static void main(String[] args) {

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

} }

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

public static void main(String[] args)

snippet 3:

public class Main

{

public static int main(String[] args) {

System.out.println("Hello, World!"); return 0; } }

Main method must return a value of type void in class Main, please

define the main method as:

public static void main(String[] args)

Snippet 4:

public class Main

{

public static void main() {

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

 The main() method also accepts some data from the user. It accepts a group of strings, which is called a string array. It is used to hold the command line arguments in the form of string values.

**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");

} }

o/p

Main method with String[] args

Snippet 6:

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

int x = y + 10;

System.out.println(x);

} }

the data type of a variable must be declared before it is used.

int x = y + 10;

^

symbol: variable y

location: class Main

Snippet 7:

public class Main {

public static void main(String[] args) {

int x = "Hello";

System.out.println(x);

}

}

error: incompatible types: String cannot be converted to int

int x = "Hello";

^

Sippet 8:

public class Main {

public static void main(String[] args) {

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

}

}

error: ')' expected

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

Snippet 9:

public class Main {

public static void main(String[] args) {

int class = 10;

System.out.println(class);

}

}

Class is a keyword in java it can not be declare as identifier.

error: <identifier> expected

int class = 10;

^

error: <identifier> expected

System.out.println(class);

^

ERROR!

error: illegal start of type

System.out.println(class);

^

ERROR!

error: <identifier> expected

System.out.println(class);

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);

}

}

Op:

you cannot call display() directly without creating an instance of the Main class.

Yes, method overloading is allowed in Java. Method overloading occurs when multiple methods in the same class have the same name but different parameter lists.

Snippet:11

public class Main {

public static void main(String[] args) {

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

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

}

}

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

The length of the array is 3 and index is 5 therefore it is array iut of bond exception.

Snippet:12

public class Main {

public static void main(String[] args) {

while (true) {

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

}

}

}

Op:

Infinite loop

Use Break Statements to avoid infinite loops.

Snippet:13

public class Main {

public static void main(String[] args) {

String str = null;

System.out.println(str.length());

}

}

Op:

Exception in thread "main" java.lang.NullPointerException: Cannot invoke "String.length()" because "<local1>" is null

if a null local variable is used, then JVM prints the name of variable as <local> as shown

Snippet:14

public class Main {

public static void main(String[] args) {

double num = "Hello";

System.out.println(num);

}

}

Op:

error: incompatible types: String cannot be converted to double

double num = "Hello";

^

Inappropriate use if datatype

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);

}

}

Op

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

int result = num1 + num2;

^

Before performing operations, verify the types of the operands to ensure they're compatible.

Snippet:16

public class Main {

public static void main(String[] args) {

int num = 10;

double result = num / 4;

System.out.println(result);

}

}

Op

2.0

As expected the output is in double form.

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);

}

}

Op:

error: illegal start of expression

int result = a \*\* b;

^

The \*\* operator is not valid in Java for exponentiation because Java does not have a built-in exponentiation operator.

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);

}

}

OP

20

Expressions with higher-precedence operators are evaluated first.

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);

}

}

operation that has no meaning in ordinary arithmetic and is, therefore, undefined.

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

at Main.main

Snippet 20:

public class Main {

public static void main(String[] args) {

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

}

}

error: ';' expected

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

 it occurs syntax of program is not syntactically correct.

Snippet 21:

public class Main {

public static void main(String[] args) {

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

// Missing closing brace here

}

error: reached end of file while parsing

}

Snippet 22:

public class Main {

public static void main(String[] args) {

static void displayMessage() {

System.out.println("Message");

}

}

}

error: illegal start of expression

static void displayMessage() {

^

Snippet 23:

public class main {

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");

}

}

}

Op

Value is 2

Value is 3

Default case

Break statement is missing at the end of the case therefore all the cases from case 2 will be executed.

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");

}

}

}

Op

Level 1

Level 2

Level 3

Unknown level

Break statement is missing at the end of the case therefore all the cases from case 1 will be executed.

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!");

}

}

}

Op

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

switch(score) {

^

The double values can be too large or too small for an int and decimal values will get lost in the conversion. Hence, it is a potential lossy conversion.

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: duplicate case label

case 5:

1 error

Reason:

no two constant-expression values within the same switch statement can have the same value. Execution of the switch statement body begins at the first statement in or after the matching labeled-statement