**Snippet 1:**

public class Main {

public void main(String[] args) {

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

}

}

Runtime Error:

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

public static void main(String[] args)

Solution:

static keyword is missing from the name of the Main method. It should public static void main (String[] args)

Modified 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!");

}

}

Runtime 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

Solution:

public access specifier is missing from the name of the Main method. It should public static void main (String[] args)

Modified 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;

}

}

Runtime 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)

Solution:

Return type of the Main method must be void instead of int as it is a rule/syntax of Java

and it does not return any value. Also, need to remove return 0 as it can’t be used with void.

Modified 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!");

}

}

Runtime 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

Solution:

String args[] (array of strings) is missing from the argument of the Main method. When we execute a code, the command line arguments get recorded here. It is a rule/syntax of java to compulsorily include String <name of array>[] in the Main method.

Modified 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");

}

}

Output:

Main method with String[] args

Remark:

Main method can be overloaded as it is done above (the signature/argument is different) but the execution will only start in the Main method -> public static void main(String[] args). This Main method should be present compulsorily in the code.

**Snippet 6:**

public class Main {

public static void main(String[] args) {

int x = y + 10;

System.out.println(x);

}

}

Compile Time Error:

Main.java:3: error: cannot find symbol

int x = y + 10;

^

symbol: variable y

location: class Main

1 error

Solution:

Variable y is not declared and initialized. The compiler does not know what y is unless it is declared first and y cannot be used in calculations unless it is initialized with some value first for e.g. 0.

Modified code:

public class Main {

public static void main(String[] args) {

int y = 0;

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);

}

}

Compile Time Error:

Main.java:3: error: incompatible types: String cannot be converted to int

int x = "Hello";

^

1 error

Solution:

String cannot be stored inside an int variable. We need to use a String data type variable. Java enforces type safety as different types of data require different amount of memory storage.

Modified Code:

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!"

}

}

Compile Time Error:

Main.java:3: error: ')' expected

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

^

1 error

Solution:

Closing parenthesis ) and ; is missing in System.out.println("Hello, World!" due to which we got compilation error.

Modified 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);

}

}

Compile Time Error:

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: illegal start of expression

System.out.println(class);

^

Main.java:4: error: <identifier> expected

System.out.println(class);

^

5 errors

Solution:

We cannot use a class as identifier here because class is a predefined keyword which already has some meaning in Java and it is also against the rules/syntax.

Modified Code:

public class Main {

public static void main(String[] args) {

int c = 10;

System.out.println(c);

}

}

**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);

}

}

Compile Time 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);

^

2 errors

Solution:

Methods can be overloaded using different signatures/arguments but **we cannot call a non-static method from a static method**. Therefore, need to make the methods static first.

Modified 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);

}

}

Output after modifying:

No parameters

With parameter: 5

**Snippet 11:**

public class Main {

public static void main(String[] args) {

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

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

}

}

Runtime Error:

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

at Main.main(Main.java:4)

Solution: arr is an array of 3 integers (index 0 to 2) but we are trying to print the 5th index value which is not present. Therefore, we are getting the **runtime error ArrayIndexOutOfBoundsException.**

**Snippet 12:**

public class Main {

public static void main(String[] args) {

while (true) {

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

}

}

}

Error:

Infinite Loop will be printed infinitely.

Solution:

Since the condition in the while loop is set to true, it will keep on executing forever. We need to give some appropriate condition which will eventually turn to false to exit out of the loop.

**Snippet 13:**

public class Main {

public static void main(String[] args) {

String str = null;

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

}

}

Runtime Error:

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

at Main.main(Main.java:4)

Solution:

We cannot use str.length() on the String str as it contains null value. We need to store a string inside of it first.

**Snippet 14:**

public class Main {

public static void main(String[] args) {

double num = "Hello";

System.out.println(num);

}

}

Compile Time Error:

Main.java:3: error: incompatible types: String cannot be converted to double

double num = "Hello";

^

1 error

Solution: We cannot store a string inside of a double variable. It is against the syntax of java. Therefore, compile time error occurs.

**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);

}

}

Compile Time Error:

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

int result = num1 + num2;

^

1 error

Solution:

num1 is an integer and num2 is double. Therefore, their addition will be in the form of a double value. We are trying to store a double value (bigger value) into an integer result (smaller value).

We can avoid this by making **result as double** or **type casting num2 as int** -> (int)num2 or **type casting entire addition as int** -> int result = (int)(num1 + num2)

Modified 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);

}

}

Result:

2.0

Remarks:

2.0 is expected as output as 10/4 will give 2 (int/int = int) which will be then converted to double (2.0) when stored inside a double variable.

**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);

}

}

Compile Time Error:

Main.java:5: error: illegal start of expression

int result = a \*\* b;

^

1 error

Solution:

\*\* is an invalid operator. Single \* should be used for multiplication.

Modified Code:

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);

}

}

Output:

20

Remarks:

b has two operands on its sides. + on the left and \* on the right. But \* has higher priority compared to +. Thus, b \* 2 will execute first resulting in 10. Then a + 10 will give 20.

**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);

}

}

Runtime Error:

C:\Users\Admin\Desktop\CDAC\Assignments\Assignment 2>javac Main.java

C:\Users\Admin\Desktop\CDAC\Assignments\Assignment 2>java Main

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

at Main.main(Main.java:5)

Solution:

Division by 0 is not possible in Java (even in C, C++) as it throws a runtime exception.

**Snippet 20:**

public class Main {

public static void main(String[] args) {

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

}

}

Compile Time Error:

Main.java:3: error: ';' expected

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

^

1 error

Solution:

; is missing at the end of line System.out.println("Hello, World"). It will throw a compile time error as ending the statements with ; is part of the syntax of java.

Modified 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

}

Compile Time error:

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

}

^

1 error

Solution:

Closing curly brace is missing for the class block. Therefore, the compiler throws an error stating **reached end of file while parsing**. We need to add it.

Modified Code:

public 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");

}

}

}

Compile Time Error:

Main.java:3: error: illegal start of expression

static void displayMessage() {

^

Main.java:7: error: class, interface, enum, or record expected

}

^

2 errors

Remarks:

We cannot define a method inside another method directly in Java. Therefore we get above compile time error.

**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");

}

}

}

Output:

Value is 2

Value is 3

Default case

Solution:

break statement is missing after case 2. Therefore, all the cases below it are also executing (including default case). To avoid it, we need to add break; after case 2.

Modified Code:

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");

**break;**

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");

}

}

}

Output:

Level 1

Level 2

Level 3

Unknown level

Solution:

break statement is missing after case 1. Also, there are no break statements for any of the cases. Therefore, after case 1 is executed, rest of the cases will also execute (including default case). To only execute case 1, we need to add break statement after it.

Modified 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");

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!");

}

}

}

Compile Time Error:

Switch.java:4: error: patterns in switch statements are a preview feature and are disabled by default.

switch(score) {

^

(use --enable-preview to enable patterns in switch statements)

Switch.java:5: error: constant label of type int is not compatible with switch selector type double

case 100:

^

Switch.java:8: error: constant label of type int is not compatible with switch selector type double

case 85:

^

3 errors

Solution:

Only the variables which have integer values (including characters as they are stored as integers internally) are allowed to be used as a switch selector variable. Double, float is not allowed. Therefore, we can change the data type of score variable to int instead.

Modified 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");

}

}

}

Compile Time Error:

Switch.java:10: error: duplicate case label

case 5:

^

1 error

Remarks:

We cannot use the same integer value for multiple case labels inside the same switch block. This will lead to compile time error: duplicate case label.