**Section 1**: Error-Driven Learning in Java

**Objective:** This assignment focuses on understanding and fixing common errors encountered in

Java programming. By analyzing and correcting the provided code snippets, you will develop a

deeper understanding of Java's syntax, data types, and control structures.

**Instructions:**

1. Identify the Errors: Review each code snippet to identify the errors or issues present.

2. Explain the Error: Write a brief explanation of the error and its cause.

3. Fix the Error: Modify the code to correct the errors. Ensure that the code compiles and runs as

expected.

4. Submit Your Work: Provide the corrected code along with explanations for each snippet.

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

**public static void main(String[] args)**

**Corrected Code: (Solution)**

public class Main {

public ststic 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: 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**

**Corrected Code: (Solution)**

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!");

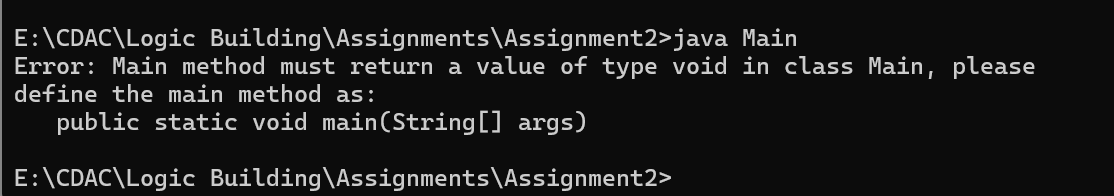
return 0;

}

}

 What error do you encounter? Why is void used in the main method?

**Error:**

****

**Corrected Code: (Solution)**

public class Main {

public static void main(String[] args) {

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

}

}

**Answer:**

Void means main method won’t return any value , other methods in other classes can return values and variables. But main can’t return anything.

**Snippet 4:**

public class Main {

public static void main() {

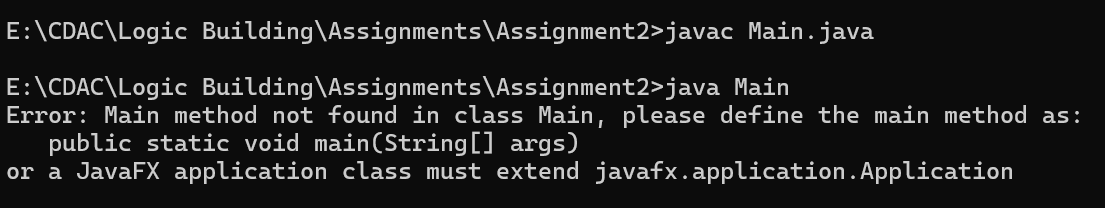
System.out.println("Hello, World!");

}

}

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

**Error:**

****

**Corrected Code: (Solution)**

public class Main {

public static void main(String[] args) {

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

}

**Answer:**

String[] args is needed to pass a Command line argument to a main method.

**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) {

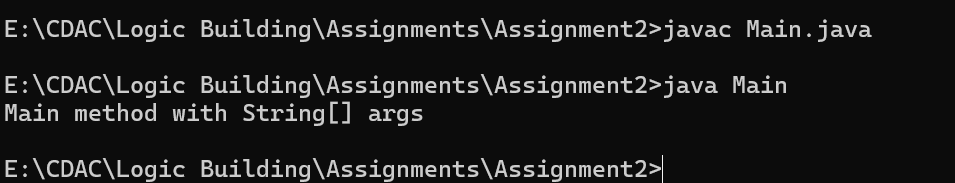
System.out.println("Overloaded main method with int[] args");

}

}

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

**Error:**

****

**Corrected Code: (Solution)**

**Answer:**

Yes, We can Have multiple main methods in our code. There are two main methods in this code this means there is a overloading of a main method. However JVM has recognized only main method which is written as **public static void main(String[] args)**

**Snippet 6:**

public class Main {

public static void main(String[] args) {

int x = y + 10;

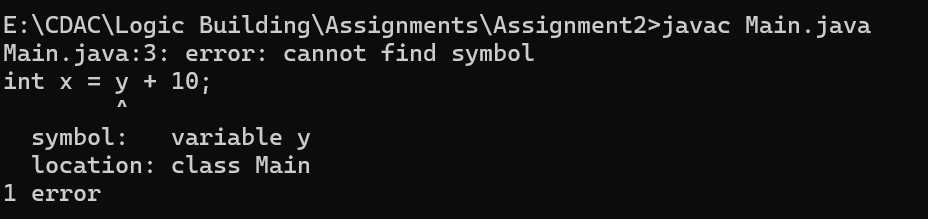
System.out.println(x);

}

}

 What error occurs? Why must variables be declared?

**Error:**

****

**Altered Code: (Solution)**

public class Main {

public static void main(String[] args) {

int y;

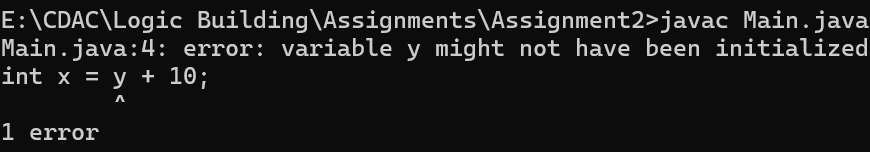
int x = y + 10;

System.out.println(x);

}

}

**Error:**

****

**Corrected Code:**

public class Main {

public static void main(String[] args) {

int y= 0;

int x = y + 10;

System.out.println(x);

}

}

**Answer:**

in this Variable y was not declared, we must declare variable before using it because of declaration reserves a memory to store the value. In this we declared variable and initialized it to resolve issue.

**Snippet 7:**

public class Main {

public static void main(String[] args) {

int x = "Hello";

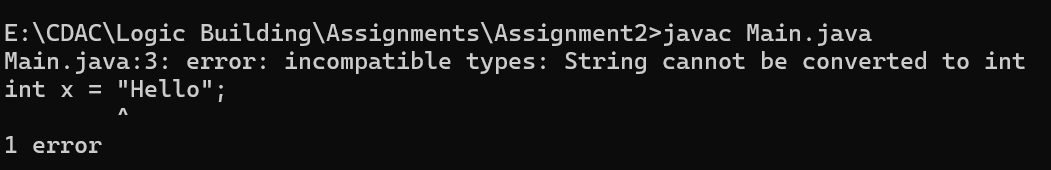
System.out.println(x);

}

}

 What compilation error do you see? Why does Java enforce type safety?

**Error:**

****

**Corrected Code: (Solution)**

public class Main {

public static void main(String[] args) {

String x = "Hello";

System.out.println(x);

}

}

**Answer:**

Java Enforces type safety because of reasons:

* Code Readability and maintainability:

Type safety make it easy for developers to understand and modify the code

* Memory Safety:

Java’s type system helps to prevent illegal memory access, which is a common isuue in c and c++.

* Enables generics for better type safety:

Generics allows java to enforce type constraints at compile time

* Improves performance:

Since type checks are done at compile time java avoids unnecessary type checks at run time, leading to a better performance.

**Snippet 8:**

public class Main {

public static void main(String[] args) {

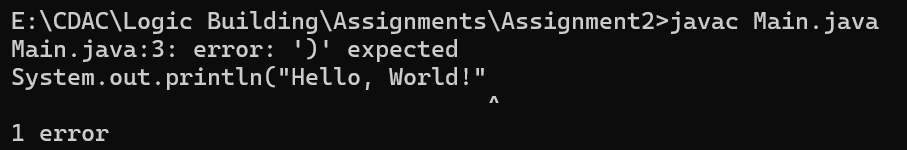
System.out.println("Hello, World!"

}

}

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

**Error:**

****

**Corrected Code: (Solution)**

public class Main {

public static void main(String[] args) {

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

}

}

**Answer:**

A syntax error in programming is a mistake in a structure of a code, like missing parenthesis, incorrect punctuation , or misspelled keywords, which violates rule of programming language and prevents code from compilation. It means the compiler will stop and display an error message indication where the syntax error is located.

**Snippet 9:**

public class Main {

public static void main(String[] args) {

int class = 10;

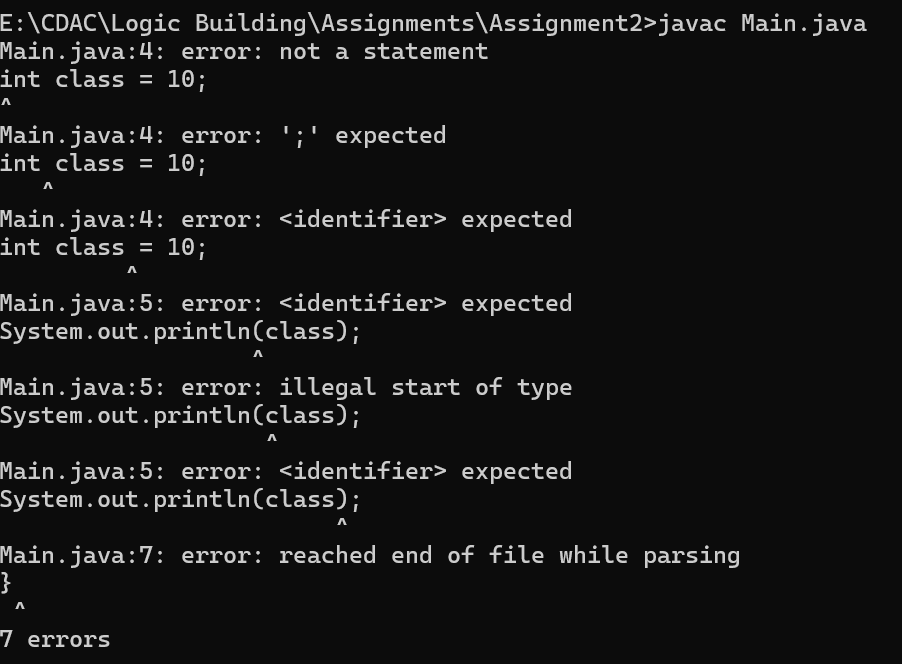
System.out.println(class);

}

}

 What error occurs? Why can't reserved keywords be used as identifiers?

**Error:**

****

**Corrected Code: (Solution)**

public class Main {

public static void main(String[] args) {

int Class = 10;

System.out.println(Class);

}

}

**Answer:**

This Error Occurred because we give a keyword as a variable name(As identifier). We cannot use reserved keywords as identifiers because they are predefined in java language, and assigning them to variables could result in unexpected behaviour.

**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();

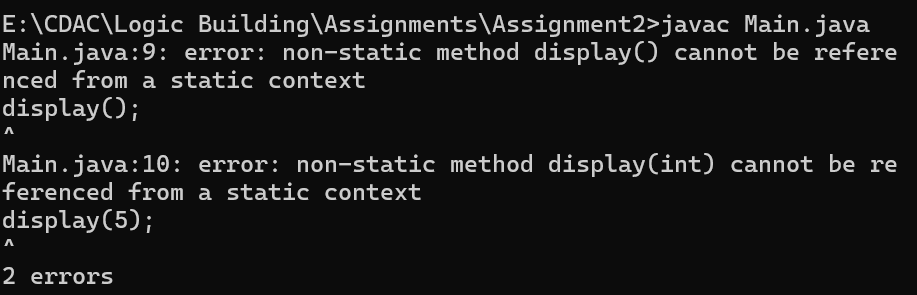
display(5);

}

}

 What happens when you compile and run this code? Is method overloading allowed?

**Error:**

****

**Corrected Code: (Solution)**

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);

}

}

**Answer:**

. The errors we're encountering are caused by trying to call non-static methods from a static context. In Java, static methods (like main) can only directly access other static methods and variables. Non-static methods belong to instances of the class, not the class itself.

In our case, we are trying to call the display() and display(int) methods, but since they are non-static methods, we cannot directly call them from the main method (which is static).

And yes, in this methods can be overloaded in Java.

**Snippet 11:**

public class Main {

public static void main(String[] args) {

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

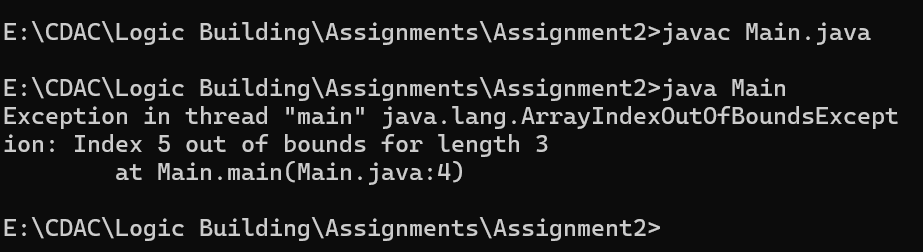
System.out.println(arr[5]);

}

}

 What runtime exception do you encounter? Why does it occur?

**Error:**

****

**Corrected Code: (Solution)**

public class Main {

public static void main(String[] args) {

int[] arr = {1, 2, 3,**4,5,6**};//either add 3 elements in array or change size of array

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

}

}

**Answer:**

This is the Runtime Exception we are getting. **ArrayIndexOutOfBoundsException** this error occurs we try to access an element at an index that does not exist. To resolve that error we added 3 element in array, or else we can change the index to access element.

**Snippet 12:**

public class Main {

public static void main(String[] args) {

while (true) {

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

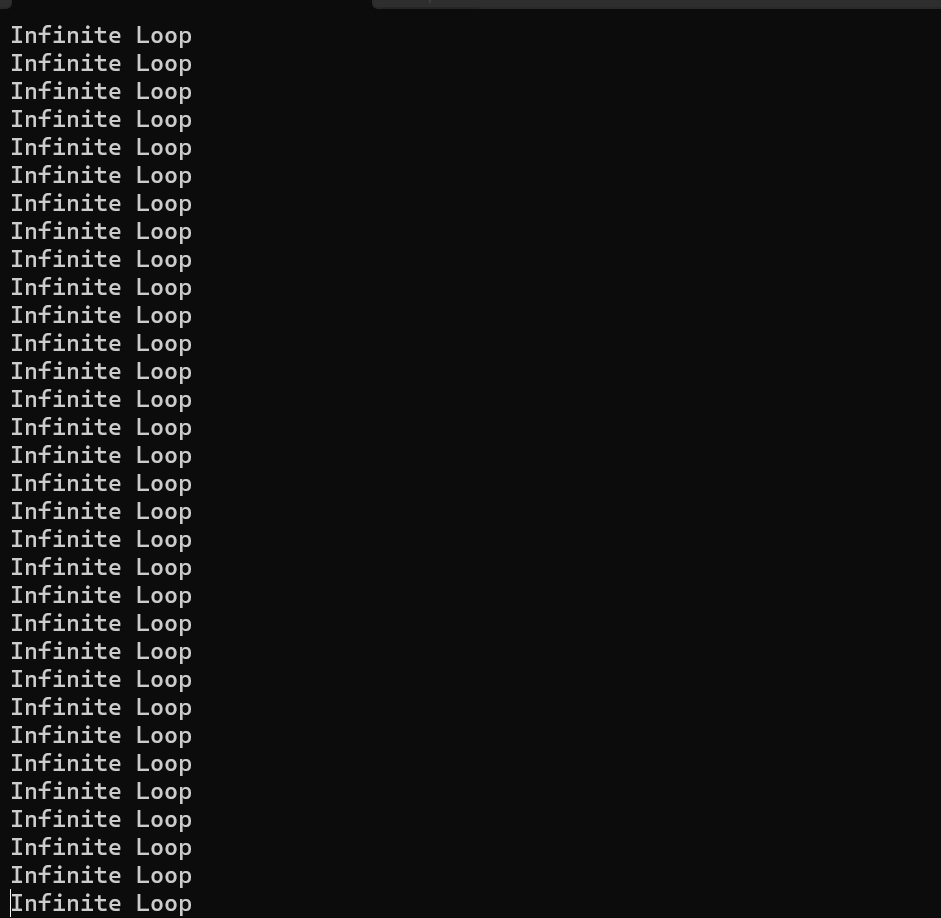
}

}

}

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

**Error:**

****

**Corrected Code: (Solution)**

public class Main {

public static void main(String[] args) {

int a = 5

while (a<=5) //To meet terminating coondition we added condition in while to

//avoid infinite loop.

{

System.out.println("5 times iteration only");

}

}

}

**Answer:**

In this Snippet there was true value was passed as a condition in while loop so the loop was going in infinite execution. So to avoid infinite iteration we added termination condition in while condition.

Condition true mean it will be always enter the loop.

**Snippet 13:**

public class Main {

public static void main(String[] args) {

String str = null;

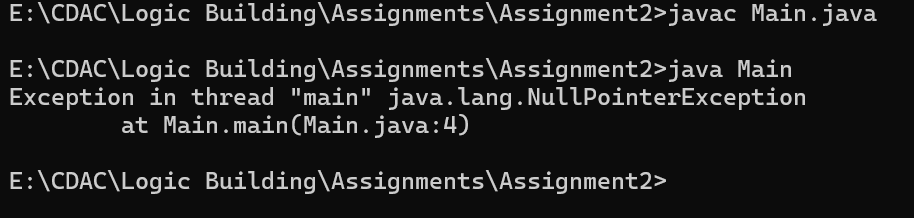
System.out.println(str.length());

}

}

 What exception is thrown? Why does it occur?

**Error:**

****

**Corrected Code: (Solution)**

public class Main {

public static void main(String[] args) {

String str = “String literal” ;

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

}

}

**Answer:**

This type of exception occurs when we try to access or modify object reference which points to null.

To resolve this exception we modified string from null to “String literal”

**Snippet 14:**

public class Main {

public static void main(String[] args) {

double num = "Hello";

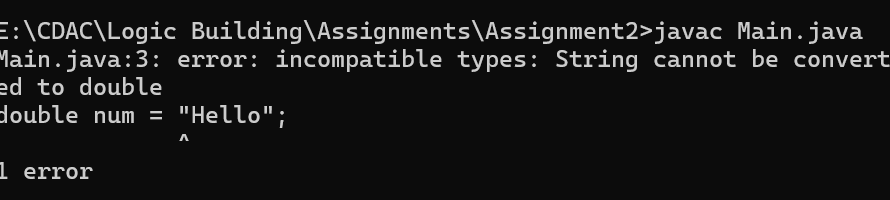
System.out.println(num);

}

}

 What compilation error occurs? Why does Java enforce data type constraints?

**Error:**

****

**Corrected Code: (Solution)**

public class Main {

public static void main(String[] args) {

String num = "Hello";

System.out.println(num);

}

}

**Answer:**

In this error it was saying incompatible type. Means when we declared a variable at that time we were storing string string into double data type variable.

**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);

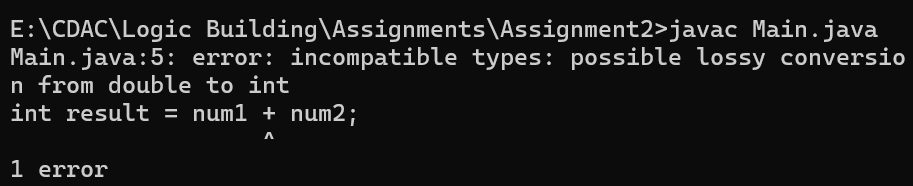
}

}

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

in operations?

**Error:**

****

**Corrected Code: (Solution)**

public class Main {

public static void main(String[] args) {

int num1 = 10;

double num2 = 5.5;

int result = num1 + (int)num2;//explicit narrow typecasting

System.out.println(result);

}

}

**Answer:**

While compiling this code it returned possible lossy conversion from double to int. **as int needs 4 bytes to store the value and double needs 8 bytes** of memory so it’s not possible to store double value into int implecitly, to handle this type of error we need to do **narrow type casting** .

.

**Snippet 16:**

public class Main {

public static void main(String[] args) {

int num = 10;

double result = num / 4;

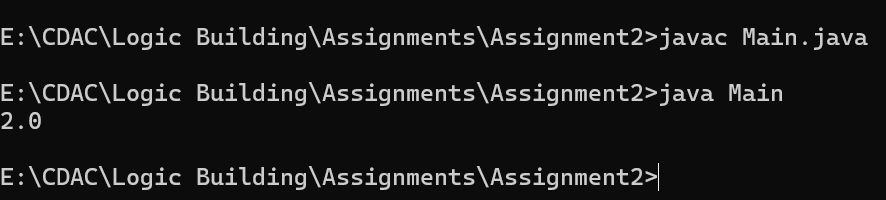
System.out.println(result);

}

}

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

**Output:**

****

**Corrected Code: (Solution)**

public class Main {

public static void main(String[] args) {

int num = 10;

double result = (double) num / 4;

System.out.println(result);

}

}

**Answer:**

The result of this operation was not expected.

But I came to know that the values on which we are performing operation are integer and variable in which we are storing result is double type so it is storing division of integer 10/4 which is 2.0 in double format.

so, to overcome this issue we need to typecast any of the integer into double.

**Snippet 17:**

public class Main {

public static void main(String[] args) {

int a = 10;

int b = 5;

int result = a \*\* b;

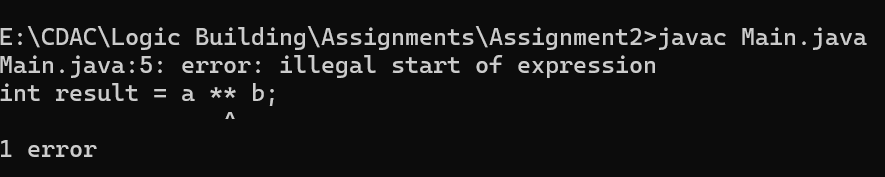
System.out.println(result);

}

}

 What compilation error occurs? Why is the \*\* operator not valid in Java?

**Error:**

****

**Corrected Code: (Solution)**

**Answer:**

Java does not have a built-in operator for exponentiation like \*\* because it's designed with a more explicit approach to mathematical operations. Instead of using a \*\* operator, you can achieve exponentiation using either:

1. **The Math.pow() method** for exponentiation.
2. **Manual loops** or **recursion** (though Math.pow() is generally the preferred approach)..

**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:**

**Corrected Code: (Solution)**

**Answer:**

Yes, It will affect the output.the output of this code will be 20.

**Snippet 19:**

public class Main {

public static void main(String[] args) {

int a = 10;

int b = 0;

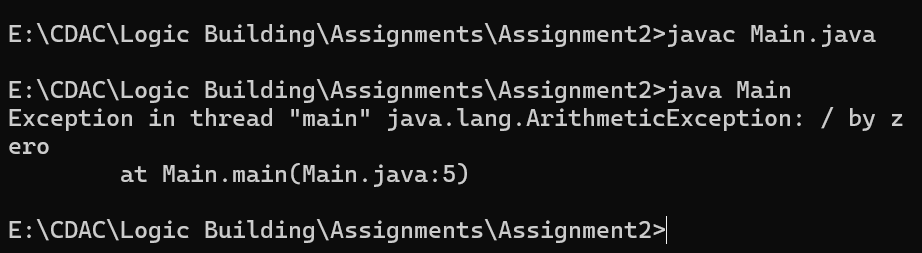
int result = a / b;

System.out.println(result);

}

}

 What runtime exception is thrown? Why does division by zero cause an issue in Java?**Error:**

****

**Corrected Code: (Solution)**

public class Main {

public static void main(String[] args) {

int a = 10;

int b = 1;// we cannot divide by zero to any number

int result = a / b;

System.out.println(result);

}

}

**Answer:**

it’s a mathematical undefined behaviour it can lead our code to deal with undefined or infinite value.

In a integer value it will return error but in floating point values it will return a positive or negative side infinity or **NaN (Not a Number)**

**Snippet 20:**

public class Main {

public static void main(String[] args) {

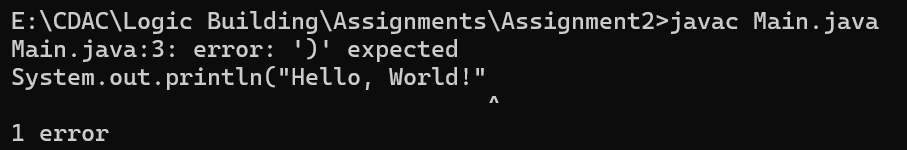
System.out.println("Hello, World")

}

}

 What syntax error occurs? How does the missing semicolon affect compilation?

**Error:**

****

**Corrected Code: (Solution)**

public class Main {

public static void main(String[] args) {

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

}

}

**Answer:**

it will give an syntax error as ‘ ; ’ is expected.

**Snippet 21:**

public class Main {

public static void main(String[] args) {

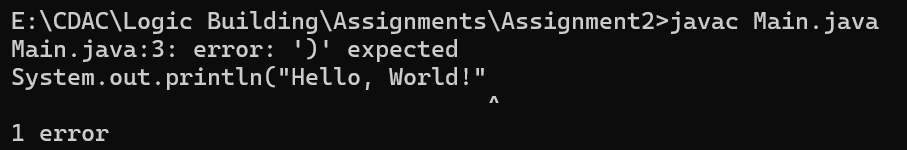
System.out.println("Hello, World!"

}

}

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

**Error:**

****

**Corrected Code: (Solution)**

public class Main {

public static void main(String[] args) {

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

}

}

**Answer:**

In Java, a **semicolon (;)** marks the end of a statement. It's essentially a way to tell the compiler that the current statement is complete, and the next one can begin..

**Snippet 22:**

public class Main {

public static void main(String[] args) {

static void displayMessage() {

System.out.println("Message");

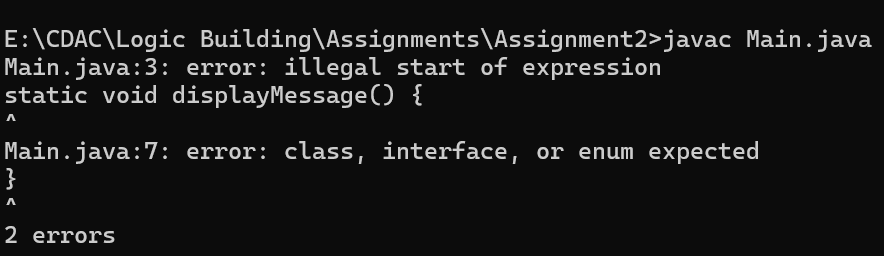
}

}

}

 What syntax error occurs? Can a method be declared inside another method?

**Error:**

****

**Corrected Code: (Solution)**

public class Main {

static void displayMessage() {

System.out.println("Message");

}

public static void main(String[] args)

{ }

}

**Answer:**

These errors are occurring because of a **syntax issue** in your code. Specifically:

1. **illegal start of expression** — This means that something in your code is in an incorrect place, or you're trying to declare a method in a way that's not valid.
2. **class, interface, or enum expected** — This error usually occurs when the code is not properly structured inside a class. In Java, methods need to be inside a class, and if you're trying to define a method outside the class body, it will throw this 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");

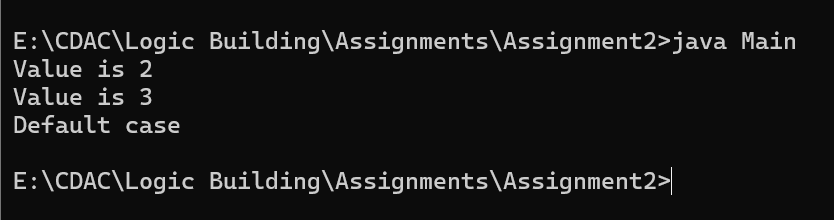
}

}

}

 Error to Investigate: Why does the default case print after "Value is 2"? How can you prevent the program from executing the default case?

**OutPut:**

****

**Corrected Code: (Solution)**

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");

break;

}

}

}**Answer:**

**Yes,**  we can prevent the program to print Default case by brecking every case after writing logic.

**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:**

**Corrected Code: (Solution)**

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");

break;

}

}

}

**Answer:**

the reason behind printing Level1 Level 2 Level 3 is break statement if we use break after completing each case then it will only print case which we selected.

**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!");

}

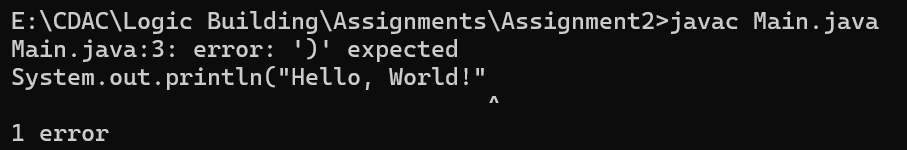
}

}

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:**

****

**Corrected Code: (Solution)**

public class Main {

public static void main(String[] args) {

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

}

}

**Answer:**

A syntax error in programming is a mistake in a structure of a code, like missing parenthesis, incorrect punctuation , or misspelled keywords, which violates rule of programming language and prevents code from compilation. It means the compiler will stop and display an error message indication where the syntax error is located.

**Snippet 26:**

public class Main {

public static void main(String[] args) {

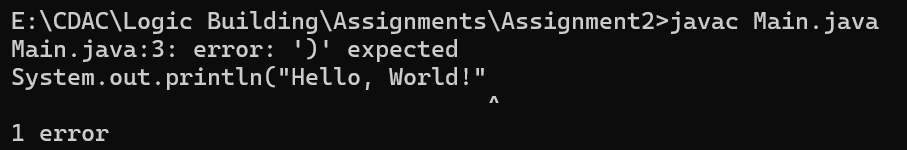
System.out.println("Hello, World!"

}

}

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

**Error:**

****

**Corrected Code: (Solution)**

public class Main {

public static void main(String[] args) {

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

}

}

**Answer:**

A syntax error in programming is a mistake in a structure of a code, like missing parenthesis, incorrect punctuation , or misspelled keywords, which violates rule of programming language and prevents code from compilation. It means the compiler will stop and display an error message indication where the syntax error is located.