**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!");

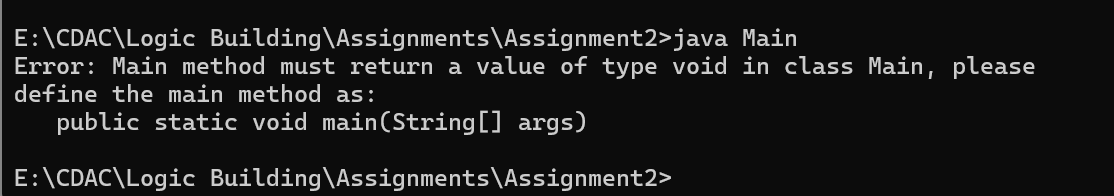
return 0;

}

}

 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() {

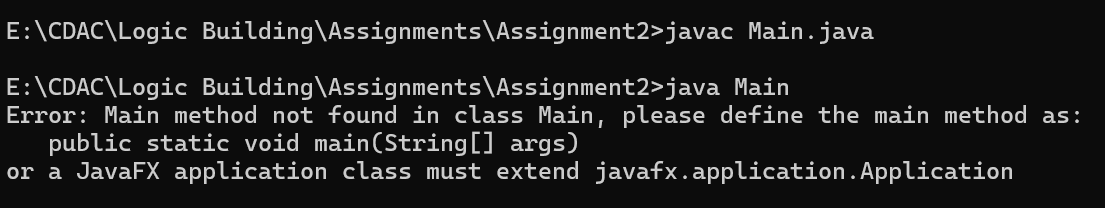
System.out.println("Hello, World!");

}

}

 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) {

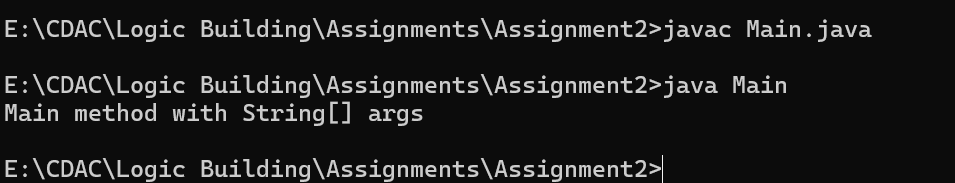
System.out.println("Overloaded main method with int[] args");

}

}

 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;

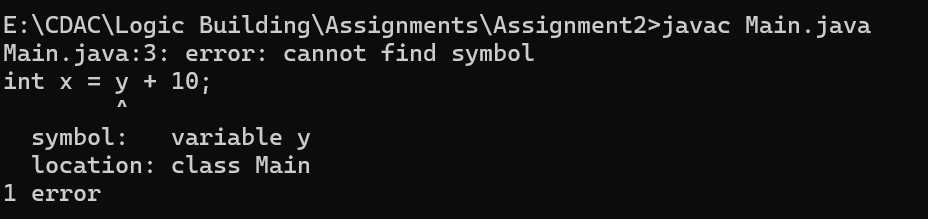
System.out.println(x);

}

}

 What error occurs? Why must variables be declared?

**Error:**

****

**Altered Code: (Solution)**

public class Main {

public static void main(String[] args) {

int y;

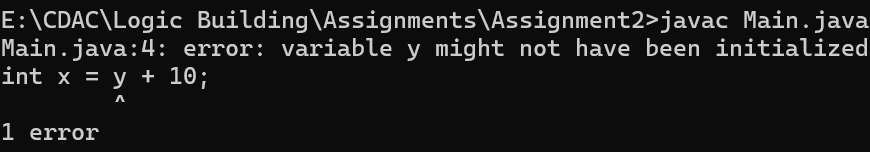
int x = y + 10;

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 4:**

public class Main {

public static void main(String[] args) {

int x = "Hello";

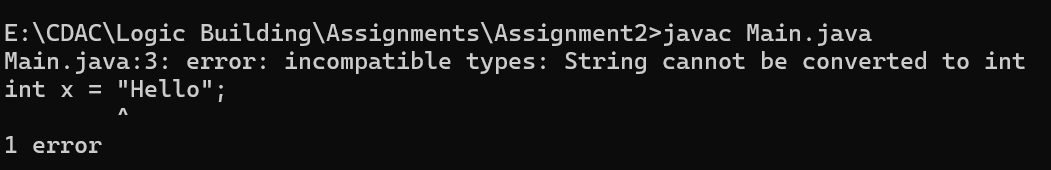
System.out.println(x);

}

}

 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 issue 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.