VaultofCodes

Assignment 1:

Task1:

Review the following codes, find and fix errors also explain the errors

Assignment 1: Java Code Review and Error Correction

Level: Beginner
Objective: Identify, correct, and explain errors in Java code snippets to strengthen understanding of Java syntax and structure.

Instructions:

For each code snippet below:

1. **Identify** the errors in the code.
2. **Rewrite** the corrected code.
3. **Explain** each error and why the correction works.

Code Snippets

---

1. **Code Snippet 1**

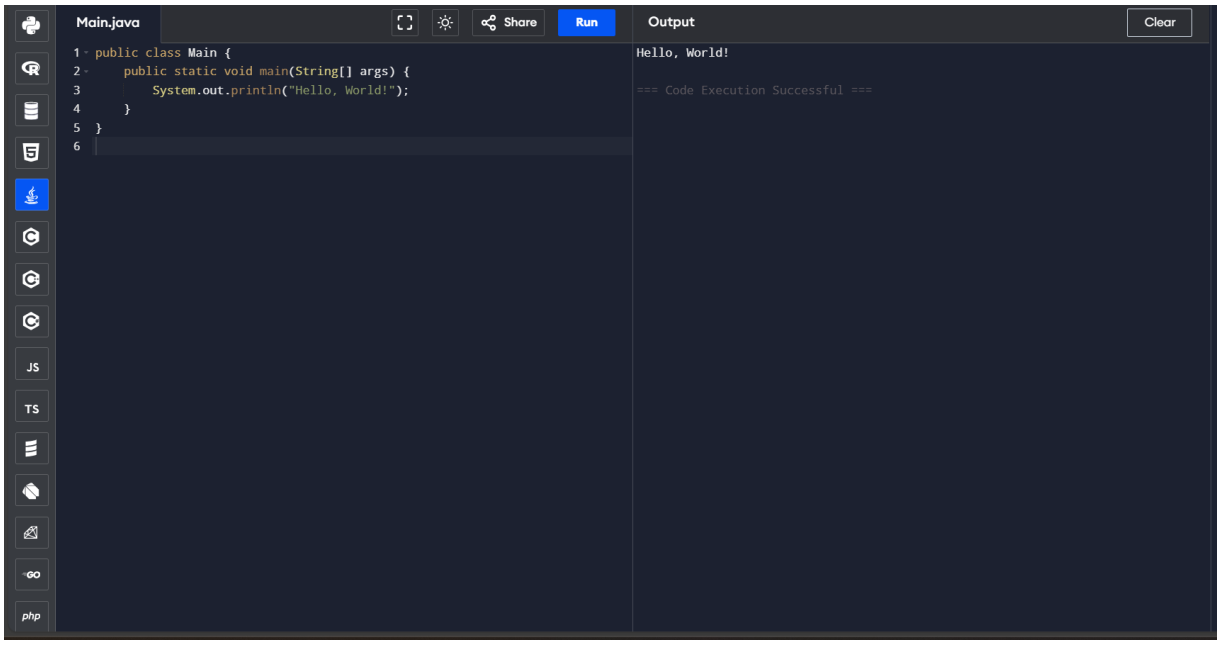
```
java Copy code
public class Main {
    public static void main(String[] args) {
System.out.println("Hello, World!")
    }
}
```

Corrected Code:

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

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

OUTPUT:



Explanation:

The original code has a syntax error because it is missing a semicolon at the end of the System.out.println("Hello, World!") statement, which is required in Java to mark the end of a command. Without the semicolon, the compiler cannot determine where the statement finishes, causing a compilation error. Additionally, the cramped formatting, with the statement placed on the same line as the method declaration, makes the code harder to read and goes against Java's recommended style guidelines. Adding the semicolon and properly formatting the code fixes the error and improves readability.

---

2. Code Snippet 2

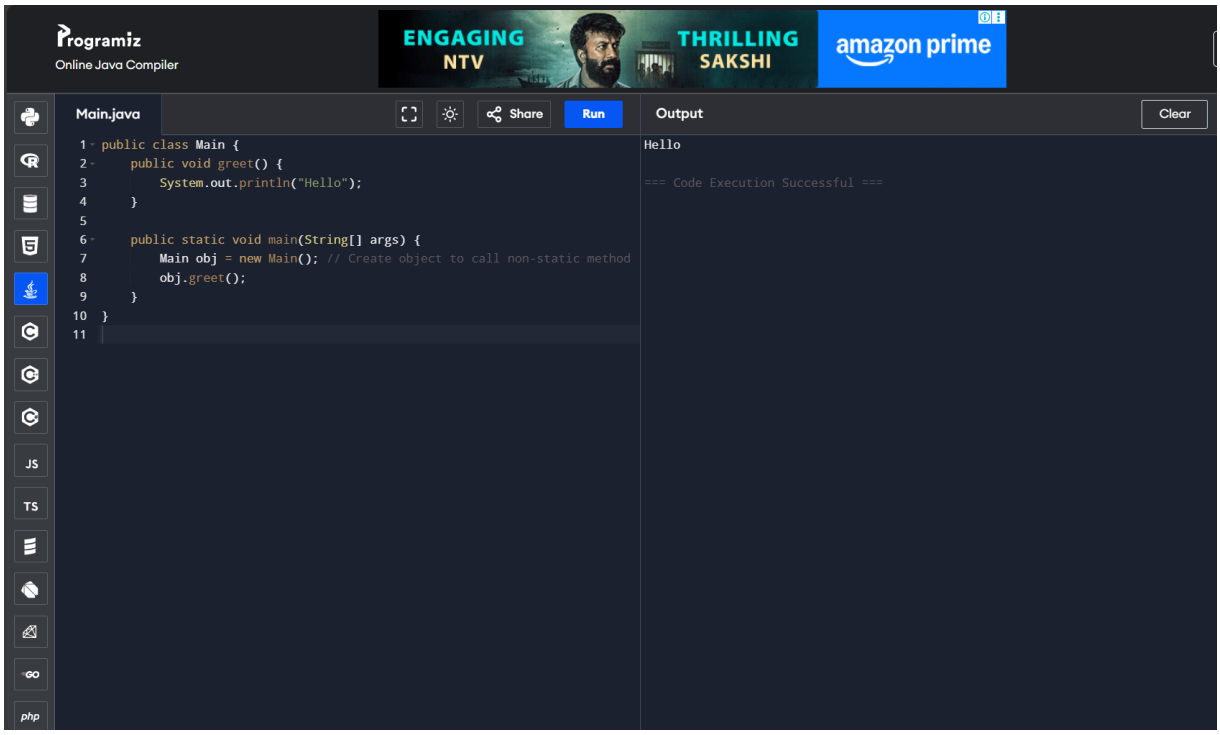
```
java Copy code public
class Main {      public
void greet() {
        System.out.println("Hello");
    }
greet();
}
```

Corrected Code:

```
public class Main {
    public void greet() {
        System.out.println("Hello");
    }

    public static void main(String[] args) {
        Main obj = new Main(); // Create object
to call non-static method
        obj.greet();
    }
}
```

OUTPUT:



Explanation:

The error occurs because greet(); is placed directly inside the class body, which is invalid in Java since statements like method calls can only exist inside methods, constructors, or initializer blocks. Additionally, without a main method, the Java program has no defined entry point to start execution. The fix involves adding a main method where we create an instance of Main and call the greet() method, as greet() is non-static and requires an object to be invoked. This correction both makes the code syntactically valid and runnable.

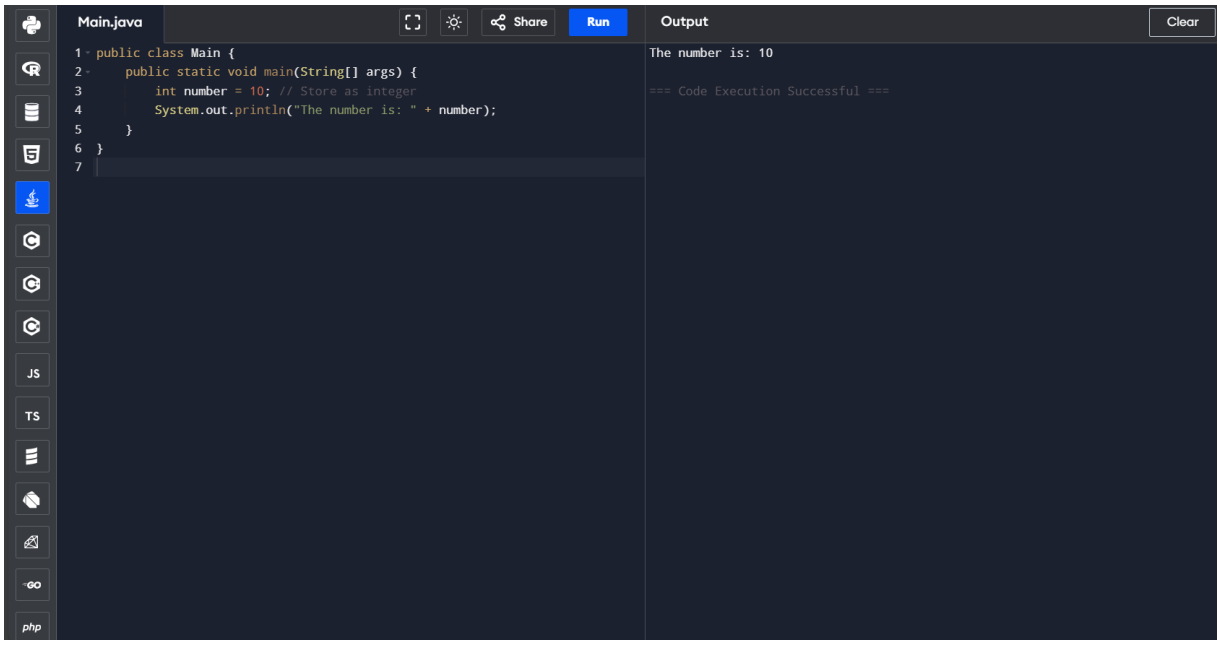
3. Code Snippet 3

```
java Copy code
public class Main {
    public static void main(String[] args) {
        int number = "10";
        System.out.println("The number is: " + number);
    }
}
```

Corrected Code:

```
public class Main {
    public static void main(String[] args) {
        int number = 10; // Store as integer
        System.out.println("The number is: " +
number);
    }
}
```

OUTPUT:



Explanation:

The error happens because Java is strongly typed, meaning a variable declared as int cannot directly store a string value like "10". To fix this, either assign the integer value 10 directly to the int variable, or if starting with a string, use Integer.parseInt("10") to convert the string into an integer. This ensures that the data type of the value matches the declared type of the variable, making the code compile and run correctly.

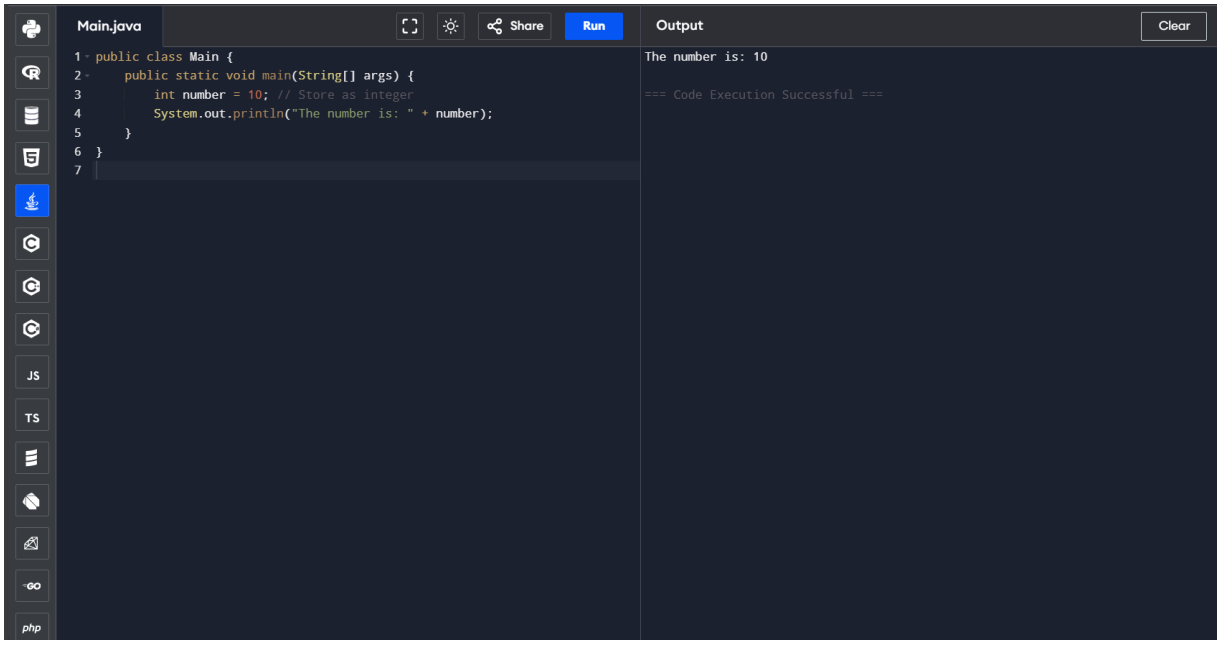
4. **Code Snippet 4**

```
java Copy code
public class Main {
    public static void main(String[] args) {
int[] numbers = {1, 2, 3, 4};
        System.out.println("The fifth element is: " + numbers[4]);
    }
}
```

Corrected Code:

```
public class Main {
    public static void main(String[] args) {
        int[] numbers = {1, 2, 3, 4};
        System.out.println("The fourth element
is: " + numbers[3]);
    }
}
```

Output:



Explanation:

This code throws an ArrayIndexOutOfBoundsException at runtime because Java arrays are zero-indexed, meaning an array with four elements has valid indices 0 to 3. Trying to access numbers[4] goes beyond its bounds. The fix is to either use a valid index, such as numbers[3] for the last element, or add a length check before accessing an index to avoid out-of-bounds errors, ensuring the program runs without exceptions.

5. Code Snippet 5

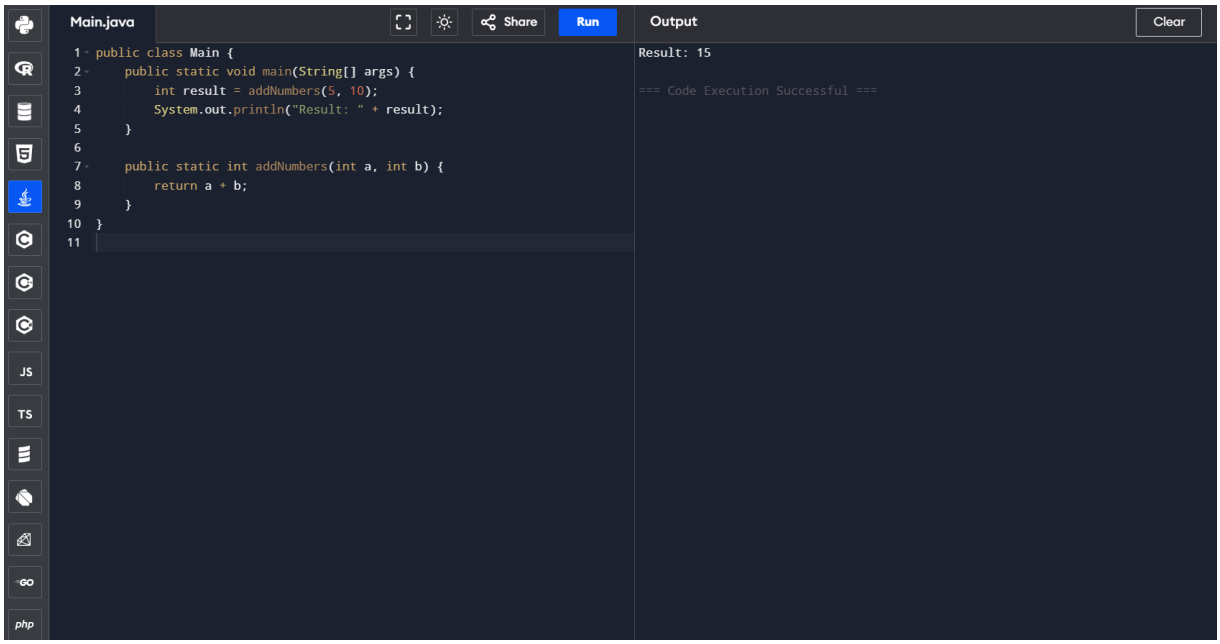
```
java Copy code
public class Main {
    public static void main(String[] args) {
int result = addNumbers(5, 10);
System.out.println("Result: " + result);
    }
    public int addNumbers(int a, int b) {
return a + b;
    }
}
```

Corrected code:

```
public class Main {
    public static void main(String[] args) {
        int result = addNumbers(5, 10);
        System.out.println("Result: " + result);
    }

    public static int addNumbers(int a, int b) {
        return a + b;
    }
}
```

Output:



Explanation:

The compilation error occurs because main is static and tries to call the non-static method addNumbers without creating an object. In Java, static methods belong to the class, while non-static methods belong to instances of the class, meaning you can't call an instance method

directly from a static method. The fix is to either declare addNumbers as static so it can be called directly from main, or create an object of Main and call the method on that object, ensuring the call is valid in Java's static context rules.

---

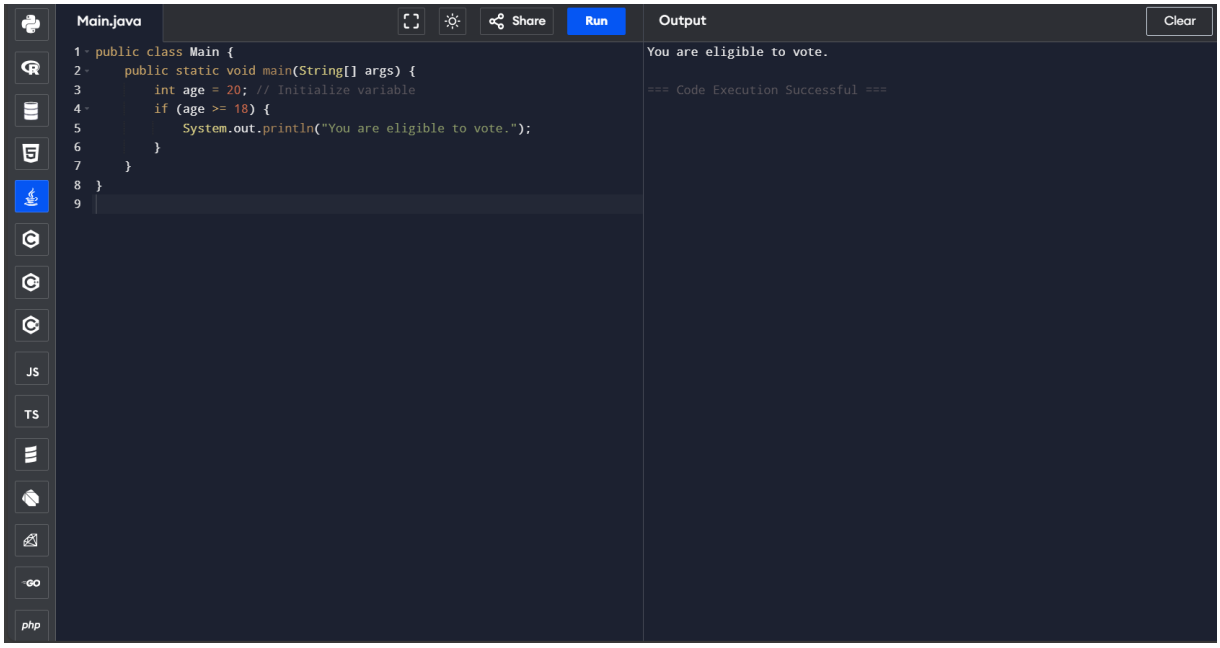
6. Code Snippet 6

```
java Copy code
public class Main {
    public static void main(String[] args) {
int age;          if (age >= 18) {
            System.out.println("You are eligible to vote.");
        }
    }
}
```

Corrected Code:

```
public class Main {
    public static void main(String[] args) {
        int age = 20; // Initialize variable
        if (age >= 18) {
            System.out.println("You are eligible
to vote.");
        }
    }
}
```

Output:



Explanation:

The error occurs because local variables in Java must be initialized before they are used, and here age is declared but never assigned a value before the if condition checks it. This leads to a compile-time error. To fix it, we can initialize age with a value before using it, or read the value from user input, ensuring the variable has a defined state before it is evaluated.

---

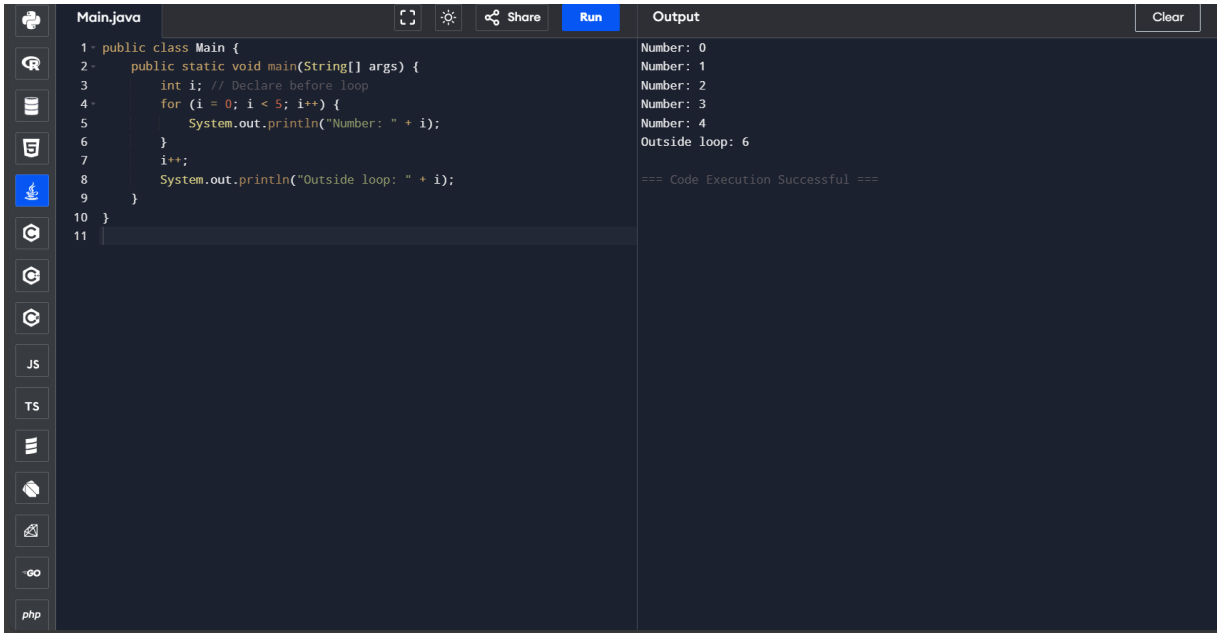
7. **Code Snippet 7**

```
java Copy code
public class Main {
    public static void main(String[] args) {
for (int i = 0; i < 5; i++) {
            System.out.println("Number: " + i);
        }
i++;
        System.out.println("Outside loop: " + i);
    }
}
```

Corrected code:

```
public class Main {
    public static void main(String[] args) {
        int i; // Declare before loop
        for (i = 0; i < 5; i++) {
            System.out.println("Number: " + i);
        }
        i++;
        System.out.println("Outside loop: " +i);
    }
}
```

Output:



Explanation:

The error occurs because the variable i is declared within the for loop header, which limits its scope to the loop block only. Once the loop finishes, i no longer exists, so trying to increment or print it outside the loop causes a compilation error. To fix this, declare i outside the loop if you need to access it afterward, or restructure the code so i is only used inside the loop where it is in scope.

---

8. Code Snippet 8

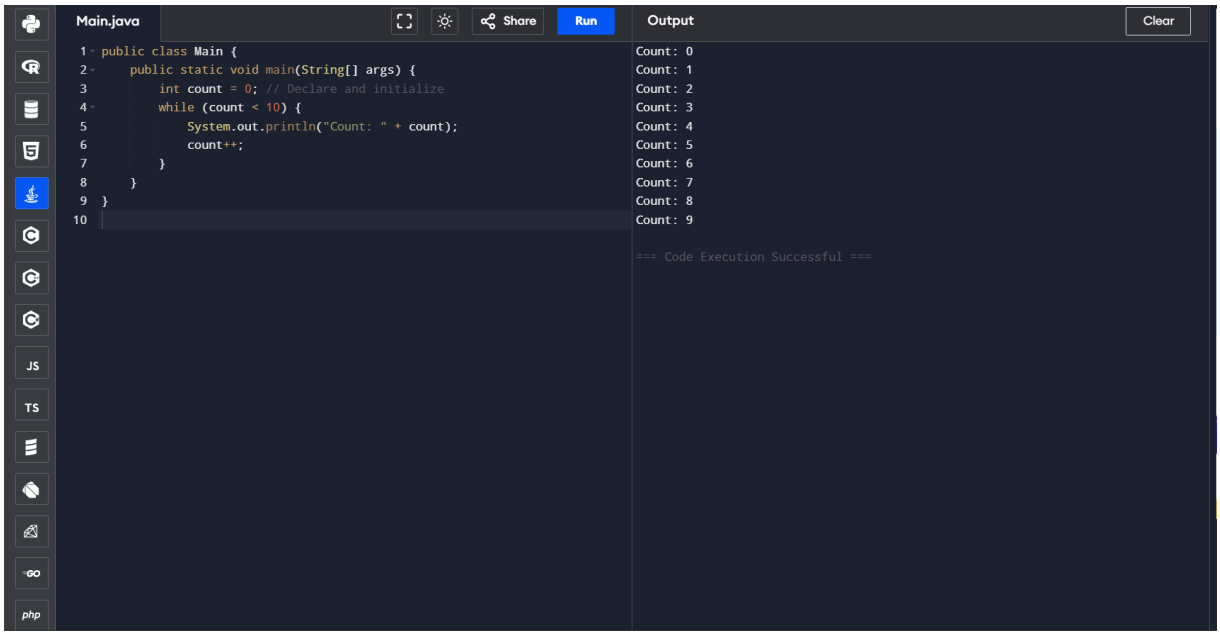
```
java Copy code
public class Main {
    public static void main(String[] args) {
while count < 10 {
            System.out.println("Count: " + count);
count++;
        }
    }
}
```

Corrected code:

```
public class Main {
    public static void main(String[] args) {
        int count = 0; // Declare and initialize
        while (count < 10) {
            System.out.println("Count: " +
count);
```

```
                count++;
            }
        }
}
```

Output:



Explanation:

The compilation error occurs because the while loop condition is missing parentheses, which are required in Java to enclose the loop's Boolean expression. Additionally, the variable count was never declared or initialized, and Java requires local variables to be explicitly assigned a value before use. Declaring count as an integer with an initial value (e.g., int count = 0;) and enclosing the condition in parentheses fixes the issue, allowing the loop to run from count = 0 to count = 9.

Submission Guidelines:

- **Format**: Submit your answers in a PDF document.
- **Explanation**: For each snippet, briefly explain the error(s) and provide the corrected code.

Write the corrected code in a file with explanations of errors Format:

Write the error

Write corrected code

Explain the error

Submit this file in pdf.

Task2:

Create a presentation on the topic

-Introduction and history of java

-OOPs in Java

Submit in pdf format