

Test 6

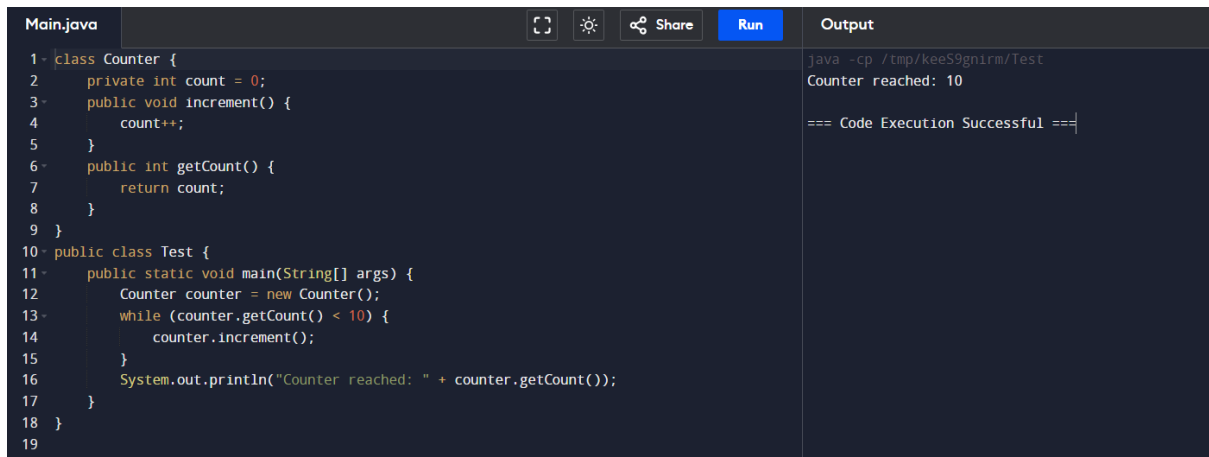
SET B Debugging(26-07-2024)

```
1. public class Counter {  
    private int count = 0;  
    public void increment() {  
        count++;  
    }  
    public int getCount() {  
        return count;  
    }  
}  
  
public class Test {  
    public static void main(String[] args) {  
        Counter counter = new Counter();  
  
        while (counter.getCount() < 10) {  
            counter.increment();  
        }  
  
        System.out.println("Counter reached: " +  
            counter.getCount());  
    }  
}
```

}

❓ Issue: Static field not retaining value across instances.

❓ Solution: Check singleton implementation for proper instance handling.



```
Main.java
1 class Counter {
2     private int count = 0;
3     public void increment() {
4         count++;
5     }
6     public int getCount() {
7         return count;
8     }
9 }
10 public class Test {
11     public static void main(String[] args) {
12         Counter counter = new Counter();
13         while (counter.getCount() < 10) {
14             counter.increment();
15         }
16         System.out.println("Counter reached: " + counter.getCount());
17     }
18 }
19
```

Output

```
java -cp /tmp/keeS9gnirm/Test
Counter reached: 10

=== Code Execution Successful ===
```


```
2. public class Employee {
    private String name;
    public Employee(String name) {
        this.name = name;
    }
    public String getName() {
        return name;
    }
}

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

```
Employee e = new Employee("John");
System.out.println(e.name); // Compilation error
}
}
```

❓ Issue: Direct access to private field name.

❓ Solution: Use getter method getName() to access private fields.

A screenshot of a Java IDE interface. The left pane shows the source code for 'Main.java'. The code defines an 'Employee' class with a private 'name' field, a constructor, and a 'getName()' method. It also defines a 'Test' class with a 'main' method that creates an 'Employee' object named 'John' and prints its name using the 'getName()' method. The right pane shows the 'Output' window with the command 'java -cp /tmp/j27h9ZNtxE/Test' and the output 'John', followed by a success message '=== Code Execution Successful ==='.

```
Main.java
1 class Employee {
2     private String name;
3     public Employee(String name) {
4         this.name = name;
5     }
6     public String getName() {
7         return name;
8     }
9 }
10 public class Test {
11     public static void main(String[] args) {
12         Employee e = new Employee("John");
13         System.out.println(e.getName());
14     }
15 }
16
```

```
Output
java -cp /tmp/j27h9ZNtxE/Test
John

=== Code Execution Successful ===
```

3. Question: Why is the FileNotFoundException not being caught when trying to open a file?

❓ Potential Issue: Make sure the FileInputStream or FileReader is enclosed in a try-catch block.

```
public class FileOpener {
    public void openFile(String filePath) {
        try {
```

```
FileReader fileReader = new FileReader(filePath);
BufferedReader br = new BufferedReader(fileReader);
String line;
while ((line = br.readLine()) != null) {
    System.out.println(line);
}
br.close();
} catch (FileNotFoundException e) {
    System.out.println("&quot;File not found: &quot; + filePath);
} catch (IOException e) {
    e.printStackTrace();
}
}
}
```

```
public class TestFileOpener {
    public static void main(String[] args) {
        FileOpener opener = new FileOpener();
        opener.openFile("&quot;missingfile.txt&quot;);
    }
}
```

```
Main.java
1- import java.io.*;
2- class FileOpener {
3-     public void openFile(String filePath) {
4-         try {
5-             FileReader fileReader = new FileReader(filePath);
6-             BufferedReader br = new BufferedReader(fileReader);
7-             String line;
8-             while ((line = br.readLine()) != null) {
9-                 System.out.println(line);
10-            }
11-            br.close();
12-        } catch (FileNotFoundException e) {
13-            System.out.println("File not found: " + filePath);
14-        } catch (IOException e) {
15-            e.printStackTrace();
16-        }
17-    }
18- }
19- public class TestFileOpener {
20-     public static void main(String[] args) {
21-         FileOpener opener = new FileOpener();
22-         opener.openFile("missingfile.txt");
23-     }
24- }
25-

Output
java -cp /tmp/xgowTzG9Cc/TestFileOpener
File not found: missingfile.txt

=== Code Execution Successful ===
```

4. Question: Why is my array not printing the correct values?

❓ Potential Issue: Ensure the array values are set correctly before printing.

```
public class PrintArray {
```

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

```
        int[] numbers = new int[3];
```

```
        numbers[0] = 10;
```

```
        numbers[1] = 20;
```

```
        numbers[2] = 30;
```




```
        for (int num : numbers) {
```

```
            System.out.println(num);
```

```
        }
```

```
    }
```

}

Main.java	   Share	Run	Output
<pre>1 public class PrintArray { 2 public static void main(String[] args) { 3 int[] numbers = new int[3]; 4 numbers[0] = 10; 5 numbers[1] = 20; 6 numbers[2] = 30; 7 for (int num : numbers) { 8 System.out.println(num); 9 } 10 } 11 } 12</pre>			<pre>java -cp /tmp/eZ3t80zCr5/PrintArray 10 20 30 === Code Execution Successful ===</pre>