Good day once again Sir Gayo here is the file for the counter and counterDemo

```java
public class Counter {
  private int count;
// constructor (int count = 1 and count = 2)
  public Counter() {
      count = 0;
  }
//  constructor getter method
  public int getCount() {
      return count;
  }
// constructor setter method
  public void increment() {
      count++;
  }
// constructor decrement method
  public void increment(int times) {
      count += times;
  }
//  constructor getter method
  public void reset() {
      count = 0;
  }
}
```

Introduction

This document provides an overview of the `Counter` and `CounterDemo` classes. The `Counter` class is a simple counter that can be incremented, reset, and queried. The `CounterDemo` class demonstrates, so sir Gayo the usage of the `Counter` class by simulating a sequence of operations and outputting results to the console.

Class Descriptions

Counter.java

- The `Counter` class represents a simple counter with a `count` attribute.
- Methods in this class allow for incrementing the counter, resetting it, and retrieving its current value.

CounterDemo.java

```java
public class CounterDemo {
    public static void main(String[] args) {
        Counter c = new Counter();
        System.out.println("Counter c count = " + c.getCount());
        // System print the counter
        c.increment(times:2);
        System.out.println(x:"Counter c incremented twice");
        System.out.println("Counter c count = " + c.getCount());
        // System print the counter
        int temp = c.getCount();
        System.out.println("Created variable temp from counter c : " + temp);
        // System print the counter
        c.reset();
        System.out.println(x:"Counter c has been reset");
        System.out.println("Counter c count = " + c.getCount());
        // System print the counter
        Counter d = new Counter();
        d.increment(times:5);
        System.out.println("Counter d count = " + d.getCount());
        // System print the counter
        d.increment();
        System.out.println(x:"Counter d incremented");
        System.out.println("Counter d count = " + d.getCount());
```

```java
        // System print the counter
        Counter e = new Counter();
        e.increment(times:6);
        System.out.println("Counter e count = " + e.getCount());

        temp = e.getCount();
        System.out.println("Assigned variable temp from counter e : " + temp);

        e.increment(times:2);
        System.out.println(x:"Counter e incremented");
        System.out.println("Counter e count = " + e.getCount());
    }
}
```

- The `CounterDemo` class demonstrates the functionality of the `Counter` class.
- It creates instances of the `Counter` class, performs operations on them, and outputs the results to the console.

CounterDemo Class Methods

main: This is the main method that demonstrates the use of the `Counter` class by creating instances and calling various methods to perform operations. It also prints the results of each operation to the console.

4. Explanation of the Console Output in CounterDemo

The `CounterDemo` class is structured to perform and display the following sequence of operations:

1. **Initialize Counter `c`**: A new `Counter` object `c` is created and initialized with `count = 0`.
2. **Increment `c` twice**: `c` is incremented by 2, bringing the count to 4.
3. **Assign `temp` variable**: The current count of `c` (4) is stored in `temp`.
4. **Reset `c`**: The counter `c` is reset to 0.
5. **Create and Increment Counter `d`**: A new `Counter` object `d` is created and incremented by 5, resulting in `count = 5`. It is then incremented by 1, bringing it to 6.
6. **Create and Assign Counter `e`**: A new `Counter` object `e` is created, incremented by 6, then assigned to `temp`. `e` is incremented by 2, bringing its count to 8.

The output

```
Counter c count = 0
Counter c incremented twice
Counter c count = 2
Created variable temp from counter c : 2
Counter c has been reset
Counter c count = 0
Counter d count = 5
Counter d incremented
Counter d count = 6
Counter e count = 6
Assigned variable temp from counter e : 6
Counter e incremented
Counter e count = 8
PS C:\Users\CCV YOUTH>
```

1. **Counter c count = 0** – Initializes Counter `c` with a count of 0.
2. **Counter c incremented twice** – Increments `c` twice (count goes up by 2).
3. **Counter c count = 4** – `c`'s count is now 4.
4. **Created variable temp from counter c : 4** – Stores `c`'s count (4) in a temporary variable, `temp`.
5. **Counter c has been reset** – Resets `c`'s count back to 0.
6. **Counter c count = 0** – Shows that `c`'s count is now 0.
7. **Counter d count = 5** – Initializes Counter `d` and sets its count to 5.
8. **Counter d incremented** – Increments `d` by 1.
9. **Counter d count = 6** – `d`'s count is now 6.
10. **Counter e count = 6** – Initializes Counter `e` and sets its count to 6.
11. **Assigned variable temp from counter e : 6** – Assigns `e`'s count (6) to `temp`.
12. **Counter e incremented** – Increments `e` by 2.
13. **Counter e count = 8** – `e`'s count is now 8.

This sequence demonstrates incrementing, resetting, and storing counter values.