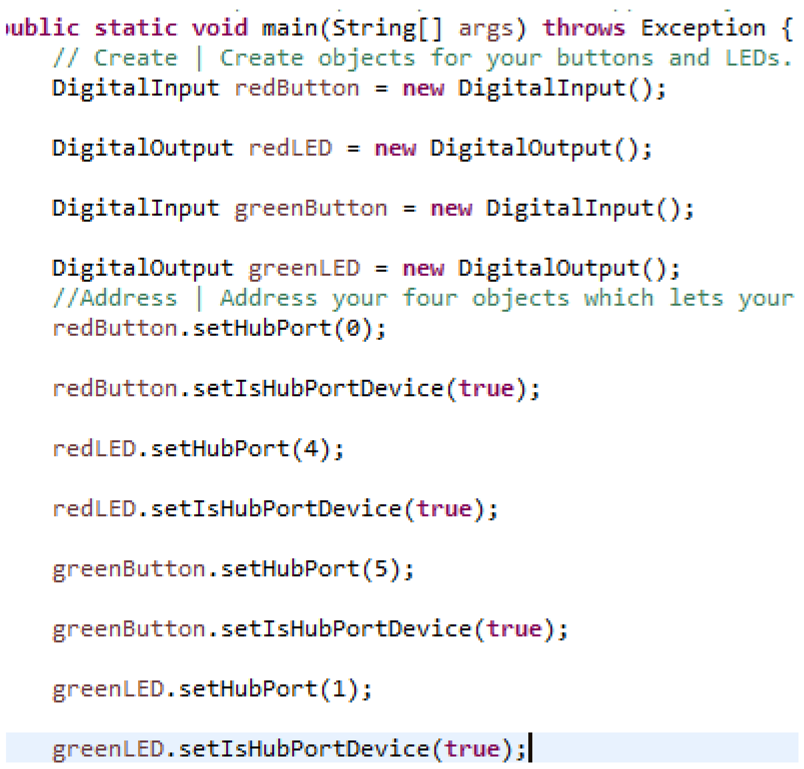
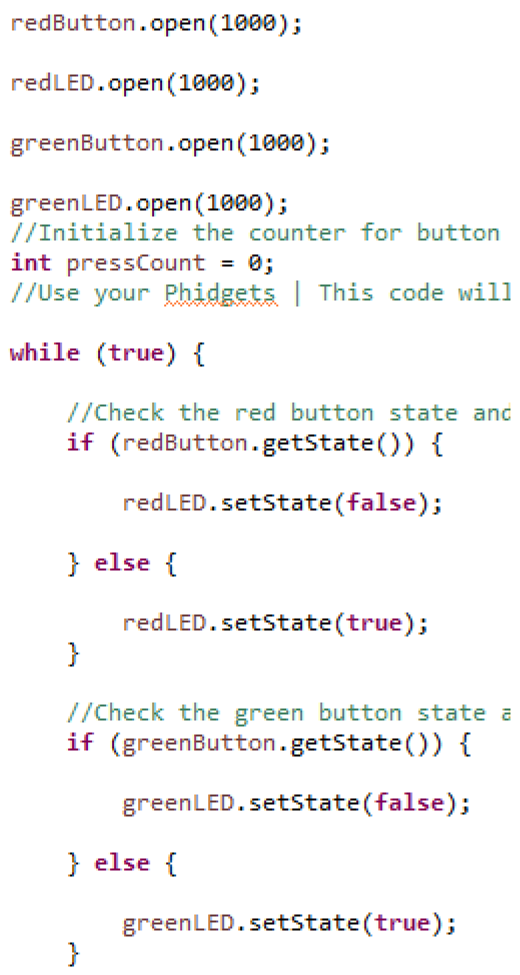
ReflectionLog UseButtonsAndLEDS



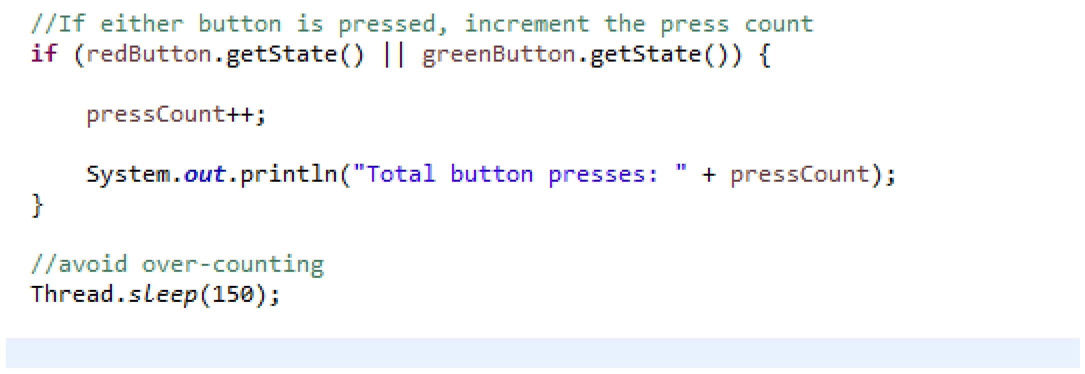
```
ublic static void main(String[] args) throws Exception {
    // Create | Create objects for your buttons and LEDs.
    DigitalInput redButton = new DigitalInput();

    DigitalOutput redLED = new DigitalOutput();

    DigitalInput greenButton = new DigitalInput();

    DigitalOutput greenLED = new DigitalOutput();
    //Address | Address your four objects which lets your
    redButton.setHubPort(0);

    redButton.setIsHubPortDevice(true);

    redLED.setHubPort(4);

    redLED.setIsHubPortDevice(true);

    greenButton.setHubPort(5);

    greenButton.setIsHubPortDevice(true);

    greenLED.setHubPort(1);

    greenLED.setIsHubPortDevice(true);
```

This code creates four objects: two buttons (redButton and greenButton) and two LEDs (redLED and greenLED). It sets the specific hub ports (0, 4, 5, and 1) for each device, telling the program where to find them. The setIsHubPortDevice(true) method indicates that these devices are connected through a hub port. This allows the program to control the buttons and LEDs by addressing the correct ports.



```
redButton.open(1000);

redLED.open(1000);

greenButton.open(1000);

greenLED.open(1000);
//Initialize the counter for button
int pressCount = 0;
//Use your Phidgets | This code will

while (true) {

    //Check the red button state and
    if (redButton.getState()) {

        redLED.setState(false);

    } else {

        redLED.setState(true);
    }

    //Check the green button state a
    if (greenButton.getState()) {

        greenLED.setState(false);

    } else {

        greenLED.setState(true);
    }
```

This code opens connections to four Phidget devices: two buttons (redButton and greenButton) and two LEDs (redLED and greenLED), each with a 1000ms timeout. It initializes a counter (pressCount) to track button presses. The main loop continuously checks the state of each button: if the red or green button is pressed, the corresponding LED is turned off; otherwise, the LED is turned on. This ensures the LEDs are controlled based on the button press states (on when the button is not pressed, off when pressed).



```java
//If either button is pressed, increment the press count
if (redButton.getState() || greenButton.getState()) {

    pressCount++;

    System.out.println("Total button presses: " + pressCount);
}

//avoid over-counting
Thread.sleep(150);
```

This code increments the pressCount variable whenever either the red or green button is pressed. It checks if either button's state is true (pressed), and if so, it increments the pressCount and prints the total number of button presses. To avoid multiple counts for a single press (due to rapid state changes), it pauses the program for 150 milliseconds (Thread.sleep(150)) before checking the button states again. This ensures that the count is only updated once per press event.