TugOfWar Phidgets ReflectionLog

First Rendition of TugOfWar.java:

```
public 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(1);
    redLED.setIsHubPortDevice(true);
    greenButton.setHubPort(5);
    greenButton.setIsHubPortDevice(true);
    greenLED.setHubPort(4);
    greenLED.setIsHubPortDevice(true);

    //Open | Connect your program to your physical devices
    redButton.open(1000);
    redLED.open(1000);
    greenButton.open(1000);
    greenLED.open(1000);

    int redButtonPress = 0;
    int greenButtonPress = 0;
    boolean redState = redButton.getState();
    boolean greenState = greenButton.getState();

    while(redButtonPress < 10 && greenButtonPress < 10){

        if (redState != redButton.getState()) {
            if (redState != false || greenState != false)
                redButtonPress++;
                System.out.println(redButtonPress);

            }
            redState = redButton.getState();

        } else if (greenState != greenButton.getState()) {
            if (greenState != false) {
                greenButtonPress++;
                System.out.println(greenButtonPress);
            }
            greenState = greenButton.getState();
        }

        Thread.sleep(10);
    }

}
```

In this first rendition I recycled the code from the previous UseButtonsAndLEDs project, which contained the basic Phidgets setup, and incrementation structure. In this code I split the if statement, which previously had an or-condition, into two separate if-statements that would increment a count for either button, rather than a combined count. Then instead of having the loop run indefinitely, it would instead run only while the count of either buttons would be less than 10. Additionally I decreased the .*sleep*() time from 150ms, to 10ms, so the program would be able to increment at a faster, unhandicapped, rate.

Final rendition of TugOfWar.java:

```
int redButtonPress = 0;
int greenButtonPress = 0;
boolean redState = redButton.getState();
boolean greenState = greenButton.getState();

while(redButtonPress < 10 && greenButtonPress < 10){

    if (redState != redButton.getState()) {
        if (redState != false) {
            redButtonPress++;
            System.out.println("Red: " +redButtonPress);

        }
        redState = redButton.getState();

    } else if (greenState != greenButton.getState()) {
        if (greenState != false) {
            greenButtonPress++;
            System.out.println("Green: " + greenButtonPress);
        }
        greenState = greenButton.getState();
    }

    Thread.sleep(10);
}

redLED.setState(true);
greenLED.setState(true);
Thread.sleep(1000);
redLED.setState(false);
greenLED.setState(false);

for (int x = 0; x < 5; x++) {
    Thread.sleep(1000);
    if (redButtonPress == 10) {
        redLED.setState(true);
    } else if (greenButtonPress == 10) {
        greenLED.setState(true);
    }
    Thread.sleep(1000);
    redLED.setState(false);
    greenLED.setState(false);
}
```

In this final rendition I added the colour to the print statement, so that it is easier to tell what the count is for each button, and added the win procedure. Now once the while-loop ends both buttons flash for 1 second, and a for-loop begins. This for-loop runs 5 times and contains an if-else-if statement that flashes the winners button for 1 second. The general procedure goes as follows. X wins > X & Y flash > X flashes (x5), and vice versa.