Credit Name: CSE 2920 - CSE Project C

Assignment: ButtonAndLEDEvents

How has your program changed from planning to coding to now? Please Explain

```
public class ButtonAndLEDEvents {
        //Turn on/off LEDs with Global Variables
        static boolean turnRedLEDOn = false;
        static boolean turnGreenLEDOn = false;
        static int player1 = 0;
        static int player2 = 0;
```

I set up global variables which control the state of the red and green leds as I did in the ButtonEvents program. Likewise i did the same with the player scores as we are making another TugOfWar.

```
//Create
DigitalInput redButton = new DigitalInput();
DigitalInput greenButton = new DigitalInput();
DigitalOutput redLED = new DigitalOutput();
DigitalOutput greenLED = new DigitalOutput();

//Address
redButton.setHubPort(0);
redButton.setIsHubPortDevice(true);
greenButton.setHubPort(5);
greenButton.setIsHubPortDevice(true);
redLED.setHubPort(1);
redLED.setIsHubPortDevice(true);
greenLED.setHubPort(4);
greenLED.setIsHubPortDevice(true);

//Open
redLED.open(1000);
greenLED.open(1000);
redButton.open(1000);
greenButton.open(1000);
```

Set up all buttons and LEDs

```
//Event
redButton.addStateChangeListener(new DigitalInputStateChangeListener() {
    public void onStateChange(DigitalInputStateChangeEvent e) {
        //Record button state to turn on/off the red LED
        if(e.getState())
        {
            turnRedLEDOn = true;
            player1 += 1;
            System.out.println("Player 1: " + player1);
        }

        else {
            turnRedLEDOn =false;
        }

    }
});
```

I made a StateChangeListener for the redButton. Here, if the state of the redbutton after a change is true then the turnRedLEDOn is true, player1's score increases by 1 and there score is displayed. Otherwise the turnRedLEDOn is false.

```
//Event
greenButton.addStateChangeListener(new DigitalInputStateChangeListener() {
    public void onStateChange(DigitalInputStateChangeEvent e) {
        //Record button state to turn on/off the green LED
        if(e.getState()) {
        turnGreenLEDOn = true;
        player2+=1;
        System.out.println("Player 2: " + player2);

    }

    else {
        turnGreenLEDOn =false;
    }

}});
```

The same thing is done for the green button and thus for player 2.

```
while(player1 < 10 && player2 < 10) {
    //turn red LED on based on red button input
    redLED.setState(turnRedLEDOn);
    //turn green LED on based on green button input
    greenLED.setState(turnGreenLEDOn);
    //sleep for 50 milliseconds
    Thread.sleep(50);

}
```

Then similar to the first TugOfWar while both player's score is below 10 the states of each LED is controlled based on the state of turn(Red/Green)LEDon which depends on changes made to their respective button. The program sleeps for 50ms to control the frequency at which the button records an input.

```
redButton.close();
greenButton.close();

if(player1 == 10) {

     greenLED.close();
     System.out.print("Player 1 Wins!");

     for(int i = 0; i<5; i++)
     {
          redLED.setState(true);
             Thread.sleep(500);
             redLED.setState(false);
             Thread.sleep(500);
     }
 }
else {
    System.out.print("Player 2 Wins!");
    redLED.close();
     for(int i = 0; i<5; i++)
     {
          greenLED.setState(true);
             Thread.sleep(500);
             greenLED.setState(false);
             Thread.sleep(500);

     }
}
```

Buttons aren't used after the previous segment so they are closed which prevents errors as buttons are pressed quickly by both players.

Then if player 1 has a score equal to 10 then they win so there is an appropriate print statement and then a for loop flashes the red LED 5 times. The greenLED is closed because at times it could be left on right after the buttons were pressed. Closing it ensures that its off. If player 1 doesn;t have a score of 10, then player 2 wins and the

same output is given but with the green LED flashing 5 times and the redLED closes for the same reason as aforementioned.