Credit Name: CSE 2920 - CSE Project C

Assignment: ButtonEvents

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

```
public class ButtonEvents {

    static boolean turnGreenLEDon = false;
     //Handle Exceptions
    public static void main(String[] args) throws Exception {

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

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

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

I made a global boolean variable which sets the state of the green LED. It is a global variable because it needs to be accessed by multiple parts of the program which run on different threads,

Then I set up the green and red buttons and the green LED and opened them.

```
//Event
redButton.addStateChangeListener(new DigitalInputStateChangeListener() {
    public void onStateChange(DigitalInputStateChangeEvent e) {
        if (e.getState())
        {
            System.out.println("Pressed");
        }

        else
        {
            System.out.println("Not Pressed");
        }
    }
});
```

For the red button to print pressed or not pressed using events, I used a method addStateChangeListener. This detects a change 'e' that occurs to the red button. Only when the button changes state will the program check the state of the button. Therefore if the state of the red button, when a change is made, is true the button is pressed and so the print statement reflects that. Otherwise the system prints that the button is "Not Pressed".

```
greenButton.addStateChangeListener(new DigitalInputStateChangeListener() {
    public void onStateChange(DigitalInputStateChangeEvent e) {
        if (e.getState())
        {
            turnGreenLEDon = true;
        }

        else
        {
            turnGreenLEDon = false;
        }
    }
});
```

Similarly, the prompt asked me to give the green button a state change event so I used the state change events to turn the green LED on and off. Therefore using the addStateChangeListener method when the state of green button changes if the state at that moment is true then the global variable,turnGreenLEDOn, defined at the beginning is set to true otherwise it is set to false.

```
        //Keep program running
        while (true) {
            Thread.sleep(10);
            greenLED.setState(turnGreenLEDon);
        }
    }
}
```

Then in a while true loop, the greenLED's state is set to be equal to turnGreenLEDOn. The argument Thread.sleep(#), in this case 10ms, makes no difference to the program as the the green LED's state only depends on whether or not a change was made to the greenButton.