Tutorial 7: Observer Design Pattern Report

UML Class Diagram: Task 1

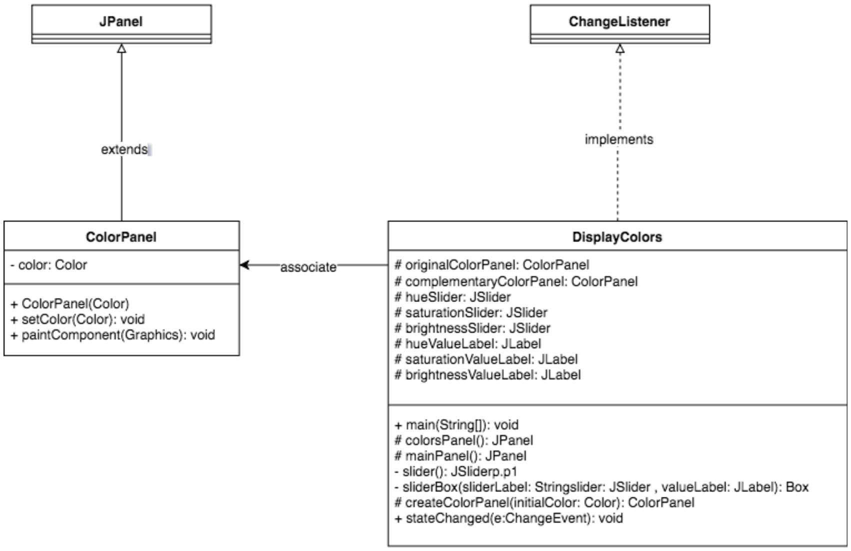

JPanel
ChangeListener
extends
implements
ColorPanel
- color: Color
+ ColorPanel(Color)
+ setColor(Color): void
+ paintComponent(Graphics): void
associate
DisplayColors
# originalColorPanel: ColorPanel
# complementaryColorPanel: ColorPanel
# hueSlider: JSlider
# saturationSlider: JSlider
# brightnessSlider: JSlider
# hueValueLabel: JLabel
# saturationValueLabel: JLabel
# brightnessValueLabel: JLabel
+ main(String[]): void
# colorsPanel(): JPanel
# mainPanel(): JPanel
- slider(): JSliderp.p1
- sliderBox(sliderLabel: Stringslider: JSlider , valueLabel: JLabel): Box
# createColorPanel(initialColor: Color): ColorPanel
+ stateChanged(e:ChangeEvent): void

Figure 1: DisplayColors has both subject and observer in the same class

UML Class Diagram: Task 2

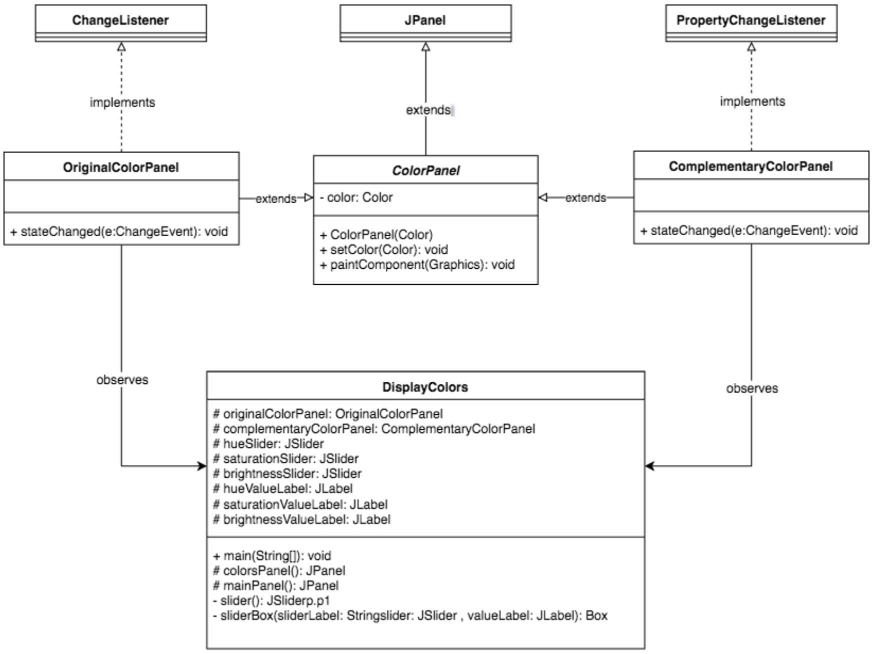

ChangeListener
JPanel
PropertyChangeListener
implements
extends
implements
OriginalColorPanel
+ stateChanged(e:ChangeEvent): void
extends
ColorPanel
- color: Color
+ ColorPanel(Color)
+ setColor(Color): void
+ paintComponent(Graphics): void
extends
ComplementaryColorPanel
+ stateChanged(e:ChangeEvent): void
observes
observes
DisplayColors
# originalColorPanel: OriginalColorPanel
# complementaryColorPanel: ComplementaryColorPanel
# hueSlider: JSlider
# saturationSlider: JSlider
# brightnessSlider: JSlider
# hueValueLabel: JLabel
# saturationValueLabel: JLabel
# brightnessValueLabel: JLabel
+ main(String[]): void
# colorsPanel(): JPanel
# mainPanel(): JPanel
- slider(): JSliderp.p1
- sliderBox(sliderLabel: Stringslider: JSlider , valueLabel: JLabel): Box

Figure 2: Subjectd in DisplayColors is observed by OriginalColorPanel and CompColorPanel

UML Class Diagram: Task 3

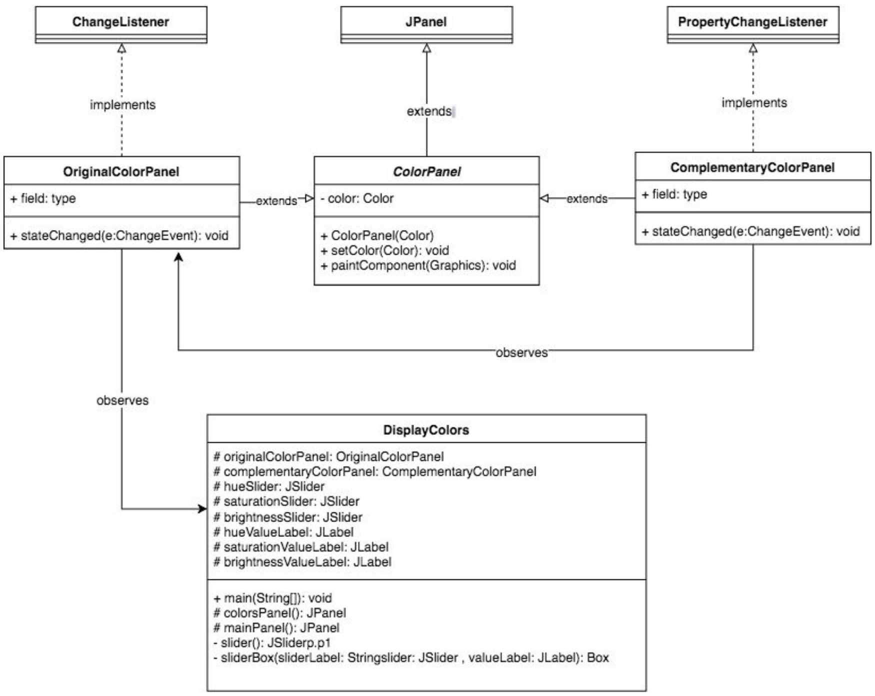

ChangeListener
JPanel
PropertyChangeListener
implements
extends
implements
OriginalColorPanel
+ field: type
+ stateChanged(e:ChangeEvent): void
extends
ColorPanel
- color: Color
+ ColorPanel(Color)
+ setColor(Color): void
+ paintComponent(Graphics): void
extends
ComplementaryColorPanel
+ field: type
+ stateChanged(e:ChangeEvent): void
observes
observes
DisplayColors
# originalColorPanel: OriginalColorPanel
# complementaryColorPanel: ComplementaryColorPanel
# hueSlider: JSlider
# saturationSlider: JSlider
# brightnessSlider: JSlider
# hueValueLabel: JLabel
# saturationValueLabel: JLabel
# brightnessValueLabel: JLabel
+ main(String[]): void
# colorsPanel(): JPanel
# mainPanel(): JPanel
- slider(): JSliderp.p1
- sliderBox(sliderLabel: Stringslider: JSlider , valueLabel: JLabel): Box

Figure 3: Subjectd in DisplayColors is observed by OriginalColorPanel, while CompColorPanel observes OriginalColorPanel to make corresponding changes

Task 4: Model View Controller Design Pattern

In order to refactor the current design in order to akin to the MVC Design Pattern, according to wikipedia (https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller) we will need three major components:

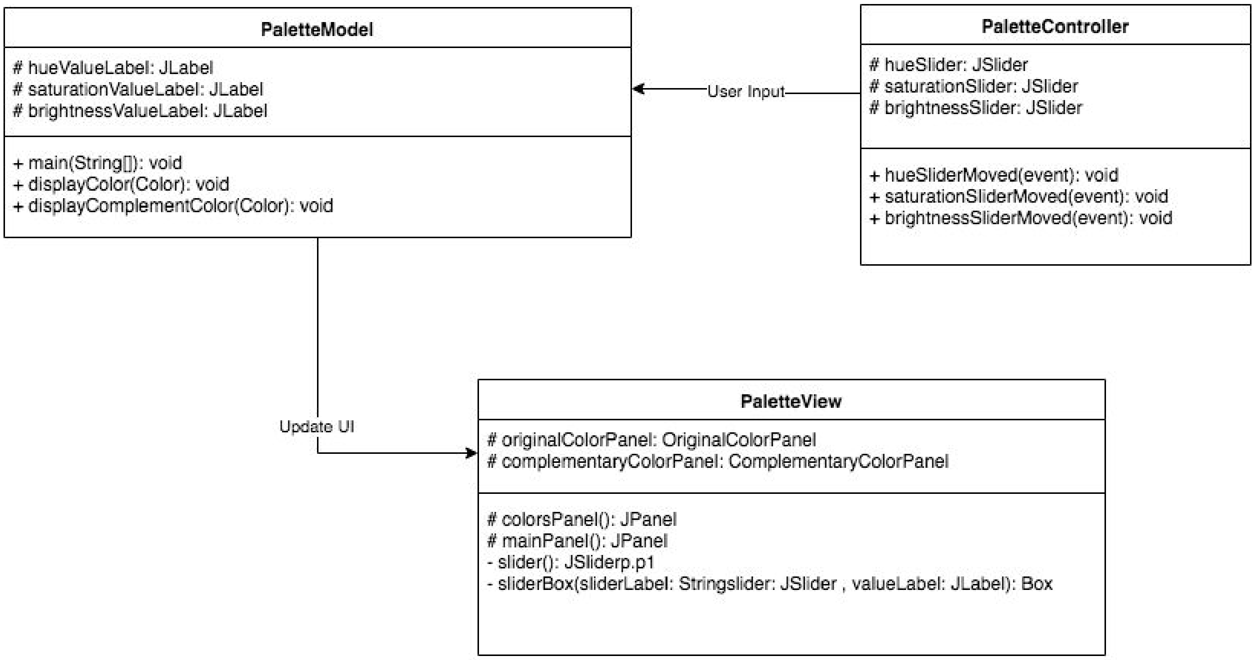
Model: a central component of the pattern. It is the application's dynamic data structure, independent of the user interface. It directly manages the data, logic and rules of the application. In our case, it will manage all the slider position data, hue data, saturation data, brightness data, color conversion rules, and other major logic components of the design. Whenever our model receives user input from the controller, it will convert the slider position data into hue data, saturation data and brightness data. After the conversion to color and complement color is completed, model will update the view to show color changes on the color panel.

View: output representation of information. In our case, it will be the sliders, color panels, and all the Swing GUI components.

Controller: accepts input and converts it to commands for the model or view. In our case, whenever the user changes the position of the sliders, controller will send the updated slider position to the model for it to convert the slider position to hue, saturation, and brightness.

A UML diagram is shown on the next page to provide a general view of the problem designed using MVC:

UML Class Diagram: Task 4



PaletteModel

hueValueLabel: JLabel
saturationValueLabel: JLabel
brightnessValueLabel: JLabel

+ main(String[]): void
+ displayColor(Color): void
+ displayComplementColor(Color): void

PaletteController

hueSlider: JSlider
saturationSlider: JSlider
brightnessSlider: JSlider

+ hueSliderMoved(event): void
+ saturationSliderMoved(event): void
+ brightnessSliderMoved(event): void

PaletteView

originalColorPanel: OriginalColorPanel
complementaryColorPanel: ComplementaryColorPanel

colorsPanel(): JPanel
mainPanel(): JPanel
- slider(): JSliderp.p1
- sliderBox(sliderLabel: Stringslider: JSlider , valueLabel: JLabel): Box