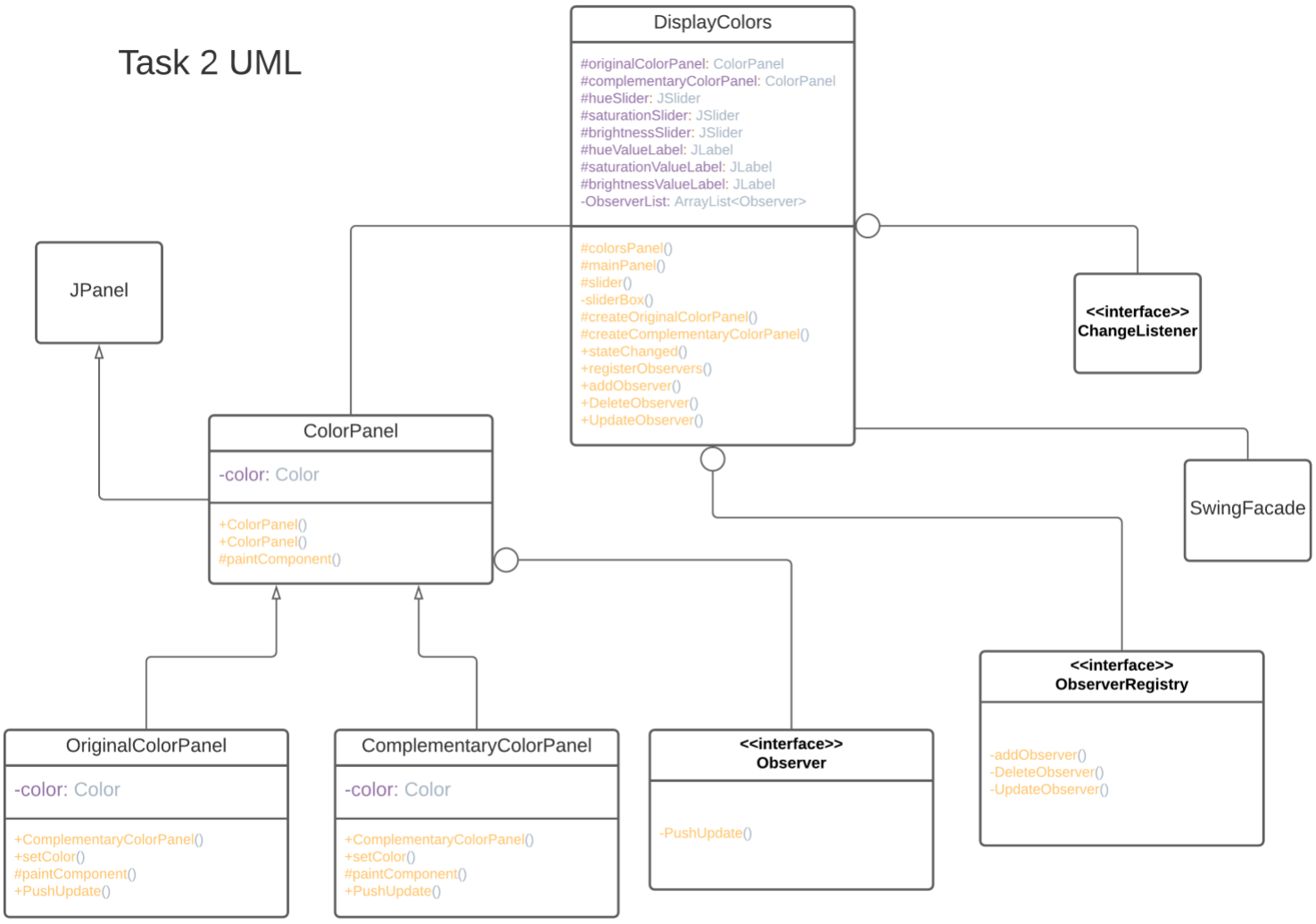
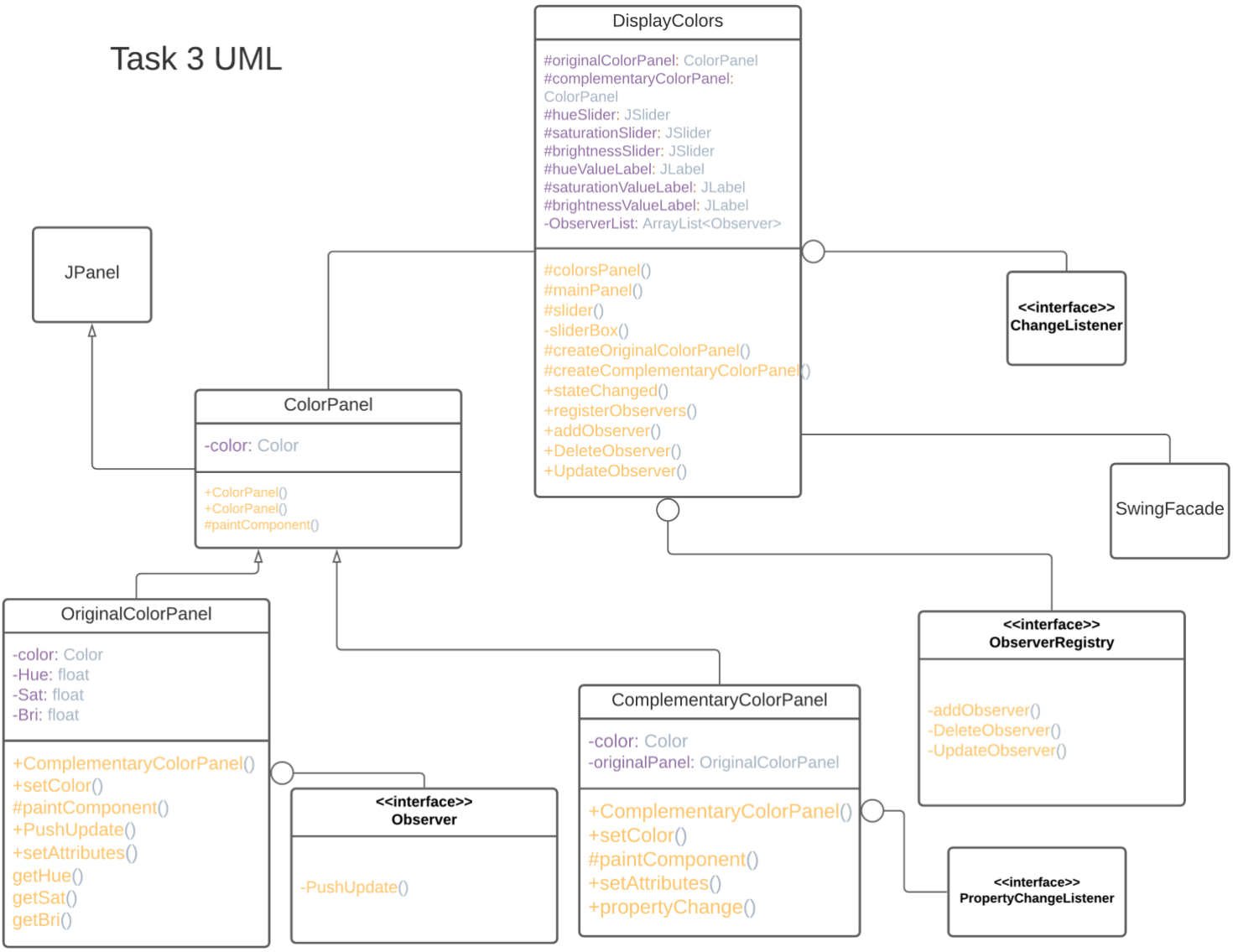
Task 2 UML

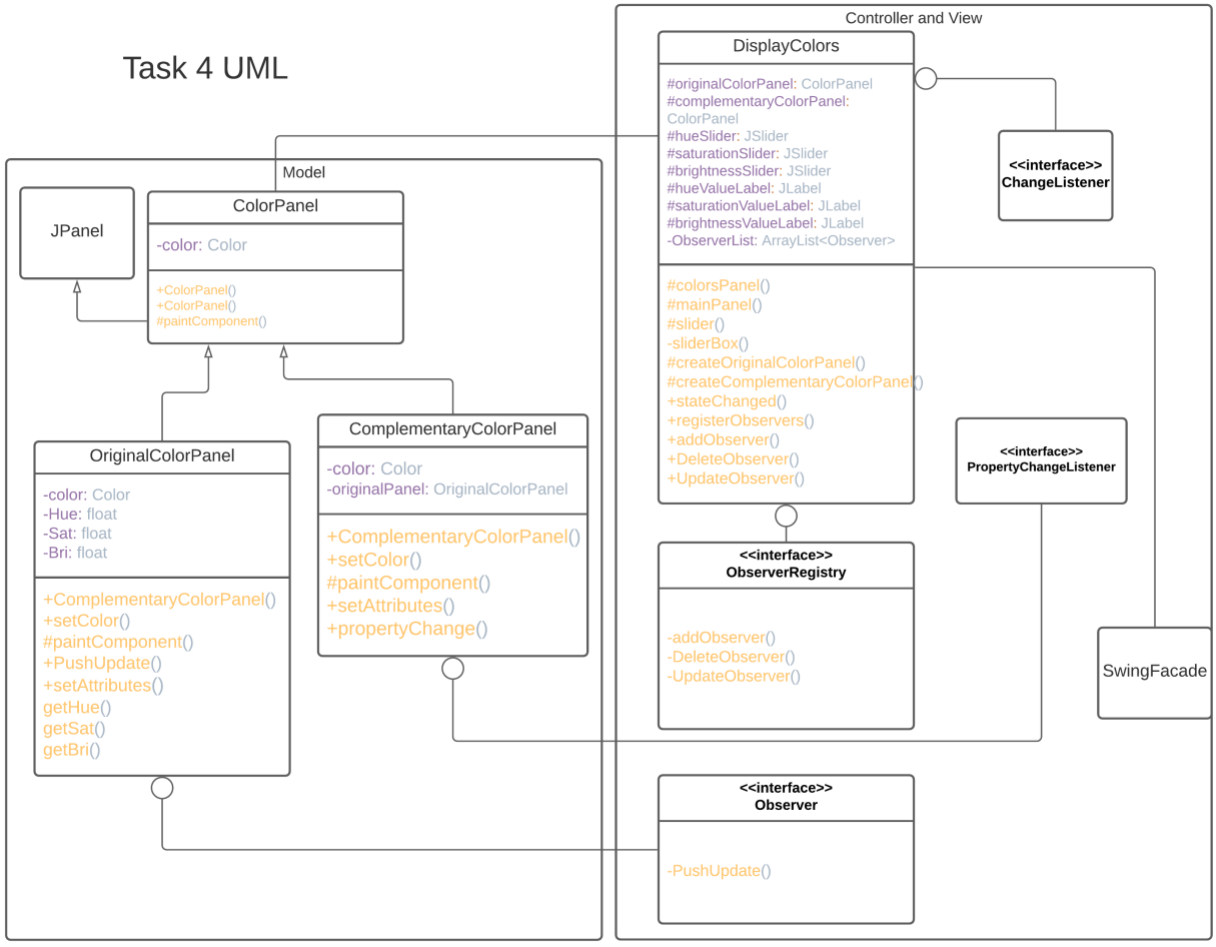

DisplayColors
#originalColorPanel: ColorPanel
#complementaryColorPanel: ColorPanel
#hueSlider: JSlider
#saturationSlider: JSlider
#brightnessSlider: JSlider
#hueValueLabel: JLabel
#saturationValueLabel: JLabel
#brightnessValueLabel: JLabel
-ObserverList: ArrayList<Observer>
#colorsPanel()
#mainPanel()
#slider()
-sliderBox()
#createOriginalColorPanel()
#createComplementaryColorPanel()
+stateChanged()
+registerObservers()
+addObserver()
+DeleteObserver()
+UpdateObserver()
JPanel
<<interface>>
ChangeListener
ColorPanel
-color: Color
+ColorPanel()
+ColorPanel()
#paintComponent()
SwingFacade
<<interface>>
ObserverRegistry
-addObserver()
-DeleteObserver()
-UpdateObserver()
OriginalColorPanel
-color: Color
+ComplementaryColorPanel()
+setColor()
#paintComponent()
+PushUpdate()
ComplementaryColorPanel
-color: Color
+ComplementaryColorPanel()
+setColor()
#paintComponent()
+PushUpdate()
<<interface>>
Observer
-PushUpdate()

Task 3 UML


DisplayColors
#originalColorPanel: ColorPanel
#complementaryColorPanel: ColorPanel
#hueSlider: JSlider
#saturationSlider: JSlider
#brightnessSlider: JSlider
#hueValueLabel: JLabel
#saturationValueLabel: JLabel
#brightnessValueLabel: JLabel
-ObserverList: ArrayList<Observer>
#colorsPanel()
#mainPanel()
#slider()
-sliderBox()
#createOriginalColorPanel()
#createComplementaryColorPanel()
+stateChanged()
+registerObservers()
+addObserver()
+DeleteObserver()
+UpdateObserver()
JPanel
<<interface>>
ChangeListener
ColorPanel
-color: Color
+ColorPanel()
+ColorPanel()
#paintComponent()
SwingFacade
OriginalColorPanel
-color: Color
-Hue: float
-Sat: float
-Bri: float
+ComplementaryColorPanel()
+setColor()
#paintComponent()
+PushUpdate()
+setAttributes()
getHue()
getSat()
getBri()
<<interface>>
Observer
-PushUpdate()
ComplementaryColorPanel
-color: Color
-originalPanel: OriginalColorPanel
+ComplementaryColorPanel()
+setColor()
#paintComponent()
+setAttributes()
+propertyChange()
<<interface>>
ObserverRegistry
-addObserver()
-DeleteObserver()
-UpdateObserver()
<<interface>>
PropertyChangeListener

Task 4 Explanation:

MVC design pattern is the process of distinguishing a project's data model (model), presentation information (view), and control information (control) into three objects. The Model contains only the pure application data, it contains no logic describing how to present the data to a user. The View presents the model's data to the user. The view knows how to access the model's data, but it does not know what this data means or what the user can do to manipulate it. The Controller exists between the view and the model. It listens to events triggered by the view (or another external source) and executes the appropriate reaction to these events. In most cases, the reaction is to call a method on the model. Since the view and the model are connected through a notification mechanism, the result of this action is then automatically reflected in the view. Looking back at task 2, to adhere to MVC design aspects, I would start by making the ColorPanel and subsequent classes the data model of the project since in these classes our data model is emerges. I would also make the DisplayColor and any sliders and listeners into the control and the view of the project. DisplayColor views the model to the user, and it controls future and past changes to our data model by the help of observers of change like the sliders and listeners.



Resources:

- https://www.geeksforgeeks.org/mvc-design-pattern/