Homework 9 README

PROJECT OVERVIEW: Shape Photo Album Application

This project uses Object-oriented programming to create a Shape Photo Album Application in which a user can perform a multitude of transformations.

MOTIVATION

The motivation for this project is to apply SOLID Principles, Separations of Concerns and to create High Quality Design using the Java programming language.

TECH FRAMEWORK USED

Model Overview

The Photo Album Shape Application Model has a series of concrete classes and methods that allow data parsed in to be manipulated based on the Users commands on how to alter the photos within the album and their screenshots. The primary function of the Model in this Design is to ensure that the Photo Album data is properly managed and all transformations are able to occur successfully based on the User's commands in the terminal.

Changes to Model Structure

The model was slightly adjusted to be able to make the connection between the Controller and the View. Firstly, based on the fact Snapshots descriptions from the user could be optional, the custom description attribute of the Snapshot was adjusted to allow for empty strings in the Snapshot Constructor.

Furthermore, in order for the model successfully to store and preserve all of the photoshapes created at and take a snapshot at specific instance, an Array List called Copy was created to perform a deep copy of all the shapes in each instance of a snapshot before the photoshape list was updated as the file reader continued reading the file and another snapshot was commanded.

In addition to the creation of a new Array List called copy, in order to conform to most of the JFrame and JPanel methods in the Graphic View of the application, the type definitions for both my Color and Dimension variables had to be transformed from doubles to integers.

Controller Overview & Design

The Photo Album Shape Application Controller contains methods that allow it to successfully serve as the intermediary between the Photo Album Model and Views (Web View & the Graphical Swing Window).

The Photo Album Controller Design was created In accordance with SOLID Principles. The PhotoAlbumController concrete class implements the IPhotoAlbumController, which only exposes the Controller class to methods that are necessary to the PhotoAlbumController functionality based on the Interface Segregation principle.

Furthermore to allow for dual functionality for both view types, a String view type was included in the PhotoAlbum constructor to serve as the user's way of letting the controller know which View was being requested. Thus, the Go method served as the primary function to create the graphical view object or the webview object.

The use of this controller design allowed for both views to be serviced without making individual controllers for each of them, thus reducing overall complexity and improving the quality of the program.

Furthermore, the open-closed principle was also employed in the construction of the Photo Album Controller concrete class. This is evident in that it can be extended to make modifications to the controller class without causing any errors to the existing controller concrete classes.

Functions of each Interface/Concrete Class in the Controller Package

- The IPhotoAlbumController interface is used to ensure that the PhotoAlbumController concrete classes (which implement it) are exposed to methods that pertain to the Controller.
- The PhotoAlbum Controller concrete class uses data from the model to delegate what the view should show based on the User input
- In order for the concrete class to perform its duties successfully, Action listeners were also added for controllers to be able to receive all instructions requested by the user when a button in the GUI view is clicked.
- In accordance with the Single responsibility SOLID principle, separate listeners were created for each button to ensure that each button knows its primary functionality.

View Overview
The Photo Album Shape Application View has a series of concrete classes and methods that allow the controller to instruct for a web or graphic view based on the user's preferences. The primary function of the View in this Design is to display the Photo Album Snapshots based on the instructions published in the input file.

View Design Specifications
In accordance with the Interface Segregation principle, the IPhotoAlbumView interface was created to expose view class types to methods that are necessary for their functionality. In this specific case of this PhotoAlbumApplication, the Draw method was the only commonality between both View types.

The View also contains a Color Adapter class to ensure that the color inputs passed in from the user are recognized by java.awt.color class. This ensured a seamless translation between the model color object and the java.awt.color object. The Dependency Inversion principle was also applied in the creation of this class. The use of composition has ensured that the color adapter class is not dependent on any of the IPhotoAlbumView concrete classes and that they are decoupled.

The Web View Concrete class and the GraphicalSwingView concrete class are separated to ensure low coupling between both classes and allow for increased flexibility in code construction.

Functions of each Interface/Concrete Class in the Controller Package
- The IPhotoAlbumView interface is used to ensure that the Web view and the Graphic View concrete classes (which implement it) are exposed to methods that pertain to them.
- The GraphicSwingWindow creates a new instance of the Graphical interface based on the filename specifications and the user's preferences.
- WebView Concrete class uses a string builder to create an html document based on the user's output file name.

Testing
Testing the functionality of the controller and view section of the Application program could be accomplished in a myriad of ways. Either through a controller mock and a mock model or testing different inputs and outputs in the controller and seeing if it's successfully translated in the view. Although constructing a mock model/mock controller is viable, I decided to test the robustness of my program using the various input files

provided on canvas and various output methods. Thus, I was able to confirm maximum flexibility within my program.