|  |
| --- |
|  |
| Software Design Principles and Patterns(SWEN 383) |
| Image Album Project - Part 2 |
| Team 3 - Dominik Cvjetković, Mateo Mustapić, Antonio Šutalo |
|  |
|  |

|  |
| --- |
|  |

Contents

[Classes 3](#_Toc449733716)

[Libraries 3](#_Toc449733717)

[Responsibilities and Collaborators 4](#_Toc449733718)

[Diagrams 6](#_Toc449733719)

[Overview Diagram 6](#_Toc449733720)

[Subsystem 6](#_Toc449733721)

[Object Diagram 7](#_Toc449733722)

[Sequence Diagram 8](#_Toc449733723)

[Rationale 9](#_Toc449733724)

# Classes

To enhance the previously created project we will expand the functionality of the existing application by adding a class which is a combination of adapter and builder patterns as well as another class to serve as a factory class for creating images of the required format. The rest of the classes will remain relatively unaltered with a major change being in the IMG Album Class which will gain a second constructor option.

* Album Adapter/Builder
  + Serves to control the album or albums being displayed. At first it checks the users requirements in order to determine if an album viewer for a directory is needed or the user wants to work with the new version which displays multiple albums at the same time. Since this class is the one that the user sees primarily this also doubles as the main facade for the application.
* IMG Album Class
  + Based on the users choice this class can either load images from a directory or from a JSON file that stores details about every image.
  + The class creates and stores all objects created by using the other classes as well as displays the images for the user to interact with.
* IMG Class
  + Class used to create image objects which stores all relevant data about each picture. It will be comprised of the standard get and set methods.
* IMG Loader Class
  + The loader class plays the role of finding images based on either a directory or a JSON file. Class creates a factory object using information on what type of image needs to be loaded and it informs it of details about the image itself. The images created by the factory are returned to the IMG Album for storage
* IMG Factory
  + Factory starts the process of creating the image type that was requested by the loader class. Furthermore it forwards image information to the image object itself and returns it to the loader.
* IMG Editor Class
  + Editor class handles any image editing that is required by the user. In order to do this it takes an image object that needs to be edited from the controller class as well as the information related to the type of editing and afterwards returns the altered image object to the controller.

## Libraries

In order to achieve the full functionality that we have intended for our classes we will need to implement the Command Interface as well as use utilities such as Java 2D. At this point in time before we start exploring the full functionality of our code we cannot predict how many other non common libraries will be included.

## Responsibilities and Collaborators

|  |  |
| --- | --- |
| Album Adapter/Builder | |
| Creates IMG Albums  Knows the IMG Albums  Displays IMG Albums | IMG Album |

|  |  |
| --- | --- |
| IMG Album | |
| Knows the IMG Loader  Knows all IMG objects  Knows the IMG Editor  Knows all commands used on a current image  Demands loading and creation of images  Demands changing of images  Displays images | IMG Loader  IMG  IMG Editor  Command Interface |

|  |  |
| --- | --- |
| IMG Loader | |
| Knows a possible file name/tag/album name  Discovers images based on possible file name/tag/album name  Finds out file name/tag/album/type of a discovered image  Creates a new factory object to create images  Returns the new image object to the IMG Album | IMG  IMG Album |

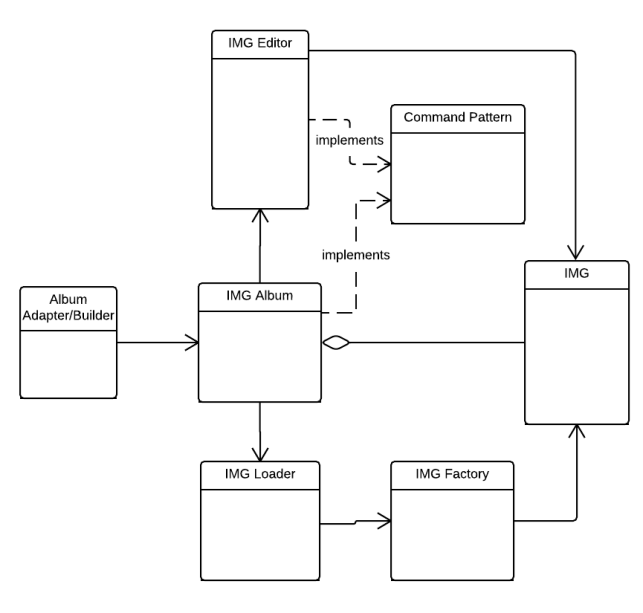
|  |  |
| --- | --- |
| IMG Factory | |
| Knows a possible file name/tag/album name  Knows image type  Creates images based on their type  Returns images | IMG  IMG Loader |

|  |  |
| --- | --- |
| IMG | |
| Knows image file name  Knows image name  Knows date/time created/edited  Knows all image tags  Knows all image albums  Can change all information if needed  Can return all information if needed | IMG Album  IMG Factory  IMG Editor |

|  |  |
| --- | --- |
| IMG Editor | |
| Knows the image that is being edited  Knows the command being used for editing  Edits the image | IMG Album  Command Interface |

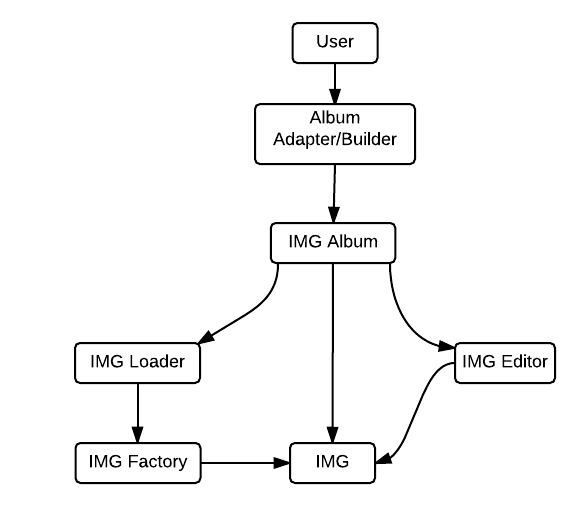
# Diagrams

## Overview Diagram



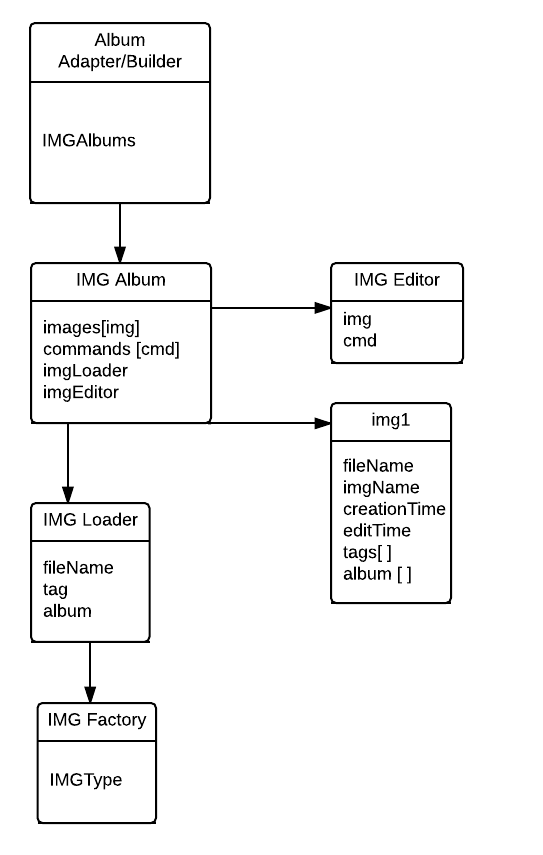
## Subsystem

The user can directly access the top of the facade, the Album Adapter/Builder class which controls creation of IMG Albums. IMG album can interact with the next level of the subsystem which is an adapted depiction of the facade pattern. Normally subsystem classes would be unaware of existence of other parts of the subsystem however in our case the elements are aware of some but not all of the subsystem components.IMG Album knows about the IMG Loader, IMG and IMG Editor. IMG Editor is aware of the IMG Album and the IMG classes while IMG Loader is aware of the IMG Factory, IMG and IMG Album classes.



## Object Diagram

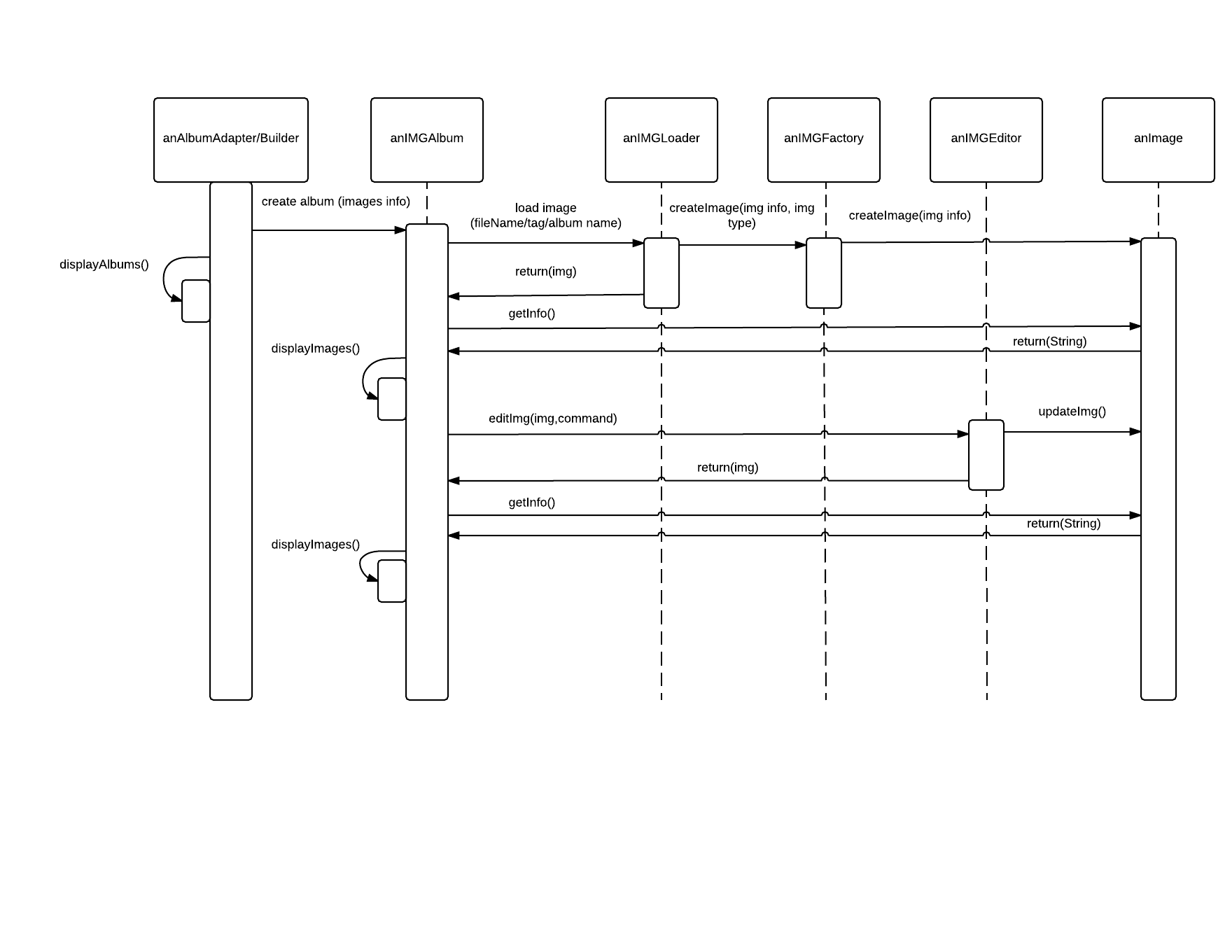
During run-time the application will consist primarily of the Album Adapter/Builder object which starts with the application itself. This class object can instantiate either one or multiple instances of IMG Album objects. When needed the IMG Loader object will be created and it will be discarded after it has served its purpose. The IMG Loader creates an IMG Factory object which is used to create IMG objects of various types. This object is discarded along with the IMG Loader after loading is complete. The IMG Editor gets created and destroyed in the same manner as the IMG Loader. This approach was chosen to simplify the procedure of storing information about image editing in order to avoid conflicts in regards to how each image was edited. Furthermore during the run-time IMG objects will be created for every image that is used in an album. These image objects will persist as long as the album persists.



## Sequence Diagram

The sequence diagram is showing the handling of one of the most common procedures of the application - loading an album and its files. This process would occur when the user starts the application and creates an album.

The Album Adapter/Builder starts and IMG Album class which starts the IMG Loader class and uses one of its methods to load the image that this album needs to have. These images can be provided with either a JSON file name or a directory. The IMG Loader class after finding the image(s) in question creates an instance of the IMG Factory to create image objects by with the information it has discovered about them and returns all of the images to the IMG Album. The IMG Album class gets all relevant information for the image and then uses that information (title, caption, tags) to update the GUI and show the images. Once the user wants to edit an image the IMG Album class starts the IMG Editor and informs it about the image object that needs to be edited as well as the command which needs to be executed. The IMG Editor updates the image in question after the modifications are done and returns the new image back to the IMG Album which proceeds to check for new information about the image and updates the GUI and show the edited image.



## Rationale

We believe that the presented details show an easy way to enhance the already existing application. With this approach the facade pattern is reinforced with an additional layer on top of the existing structure. This new layer acts as a mix of the adapter and builder patterns as it allows for creation of two types of albums based on users choice. One option is to create the album that was seen in previous iteration of the application while the other is to create multiple albums by using a JSON file with information on what each album consists of. Furthermore the factory provides easy creation of more than one type of image (jpeg, gif, png) while maintaining the existing structure.