|  |
| --- |
|  |
| Software Design Principles and Patterns(SWEN 383) |
| Image Album Project - Part 1 |
| Team 3 - Dominik Cvjetković, Mateo Mustapić, Antonio Šutalo |
|  |
|  |

|  |
| --- |
|  |

Contents

[Classes 3](#_Toc446275058)

[Libraries 3](#_Toc446275059)

[Responsibilities and Collaborators 4](#_Toc446275060)

[Diagrams 5](#_Toc446275061)

[Overview Diagram 5](#_Toc446275062)

[Subsystem 5](#_Toc446275063)

[Object Diagram 6](#_Toc446275064)

[Rationale 6](#_Toc446275065)

# Classes

In order to implement this project we have decided on having four separate classes where each class handles different responsibilities in order to create a high cohesion environment and promote the SoC design principle.

The classes we aim to use are:

* IMG Album Class
  + This is a controller class used to run the application. It creates and stores all objects created by using the other classes as well as displays the images for the user to interact with. Essentially this class behaves like a facade for the entire application.
* 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 file name, album or image tags. Class also creates image objects and loads appropriate data into them and returns these image objects to the controller class (IMG Album) for storage.
* 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

|  |  |
| --- | --- |
| 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 |

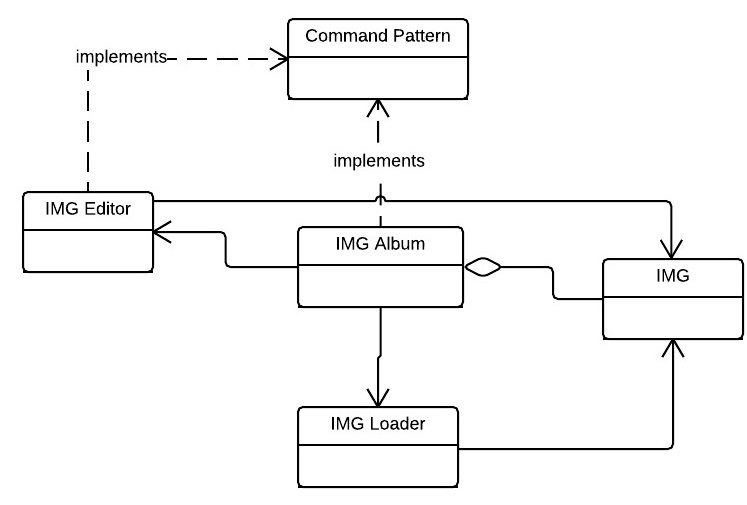
|  |  |
| --- | --- |
| IMG Loader | |
| Knows a possible file name/tag/album name  Discovers images based on possible file name/tag/album name  Finds out file name/tag/album name of a discovered image  Creates a new image object  Returns the new image object to the IMG Album | IMG  IMG Album |

|  |  |
| --- | --- |
| 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 Loader  IMG Editor |

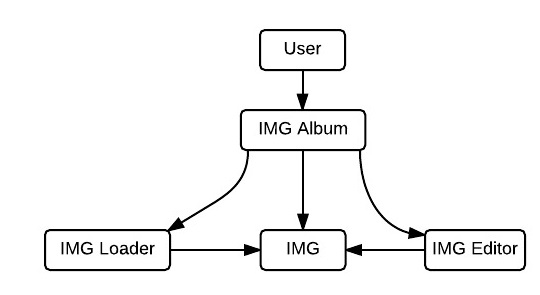
|  |  |
| --- | --- |
| IMG Editor | |
| Knows the image that is being edited  Knows the command being used for editing  Edits the image | IMG Album  Command Interface |

# Diagrams

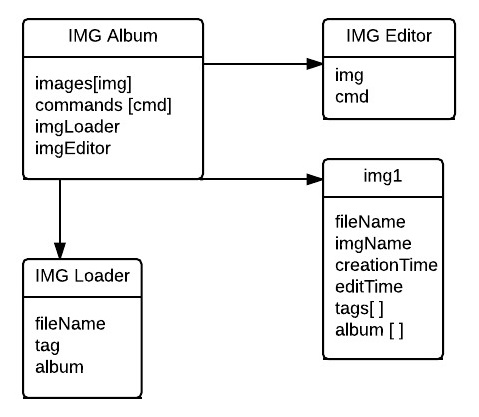
## Overview Diagram



## Subsystem



## Object Diagram



## Rationale

We believe that an approach similar to the facade pattern would be the best choice for this application for the sake of the SoC principle. Even though the design in its current form is not a completely accurate depiction of the Facade pattern it still retains some of its strong suits which allows us to maintain a system in which most of the problems would be easy to track down.

Furthermore we believe that the best way to keep track of all of the changes done to an image would be by storing all of the changes that have been done on an image in form of a command object. This way we can maintain a reference for all of the changes and the exact manner in which they have been done. With all of this in mind we agreed that this usage of the command pattern coupled with an imitation of the facade pattern would be the best course of action for this application.