SWEN PRIXOS DESIGN DOCUMENT

Authors: Isabella Sturm, Ivan Kovacević, Srđan Lazarević

First Release Date: 21 April, 2017
Final Update Date: 12 May, 2017

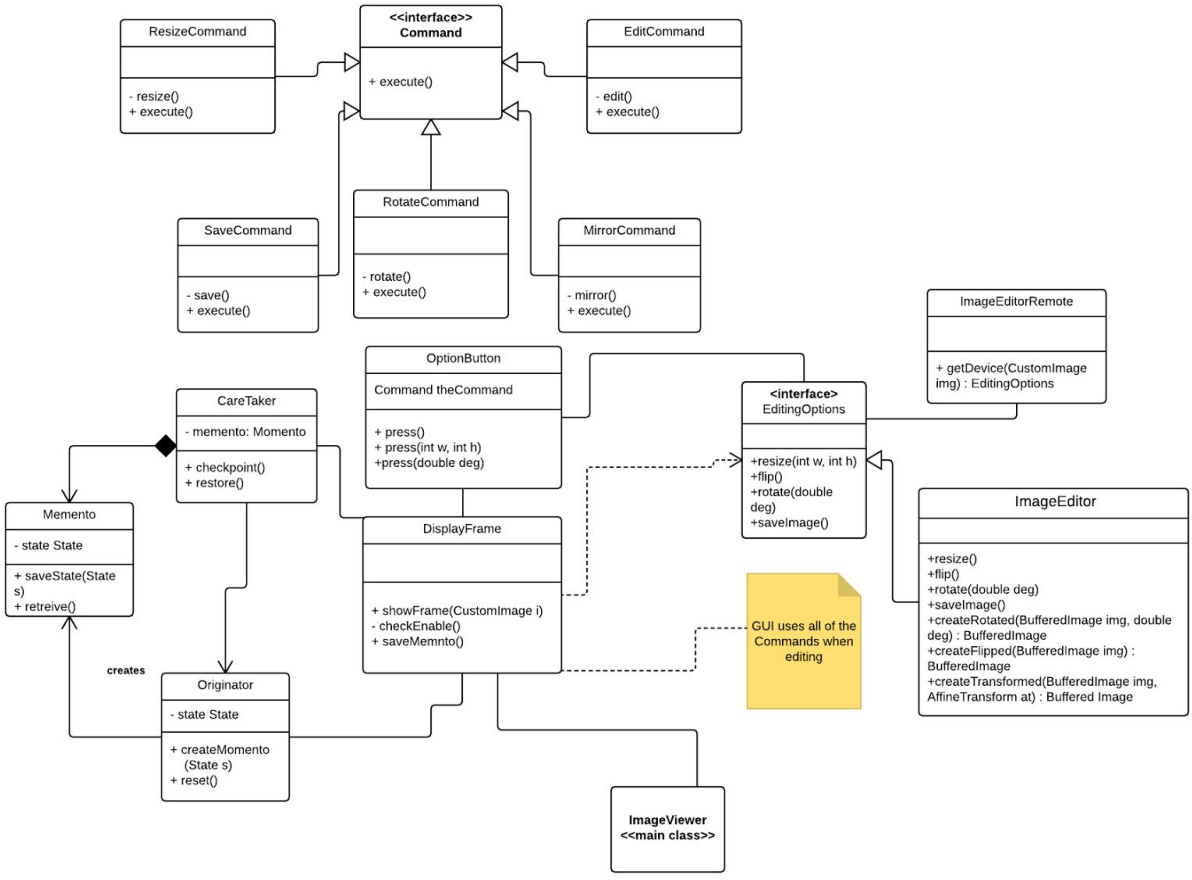
Class Table

Class Name	Class Description	Uses	Used By	Methods
ImageViewer	Main class; GUI. This class provides the user interface of the program.	ResizeCommand, CropCommand, RotateCommand, MirrorCommand, CareTaker, Originator		main() ImageViewer()
Command	Interface for commands that will control the functionalities allowed by the user		ResizeCommand, CropCommand, FlipCommand, RotateCommand, EditCommand	execute() execute(int w, int h) execute(double deg)
ResizeCommand	Implements Command interface; when called by user, resize command prompts user to change size of image	Command interface, EditingOptions	DisplayFrame	resize() execute() execute(int w, int h) execute(double deg)
CropCommand	Implements Command interface; when called, user can crop image	Command interface, EditingOptions	DisplayFrame	crop() execute() execute(int w, int h) execute(double deg)
RotateCommand	Implements Command interface, when called upon, user can chose to rotate image by custom number of degrees or by the	Command interface, EditingOptions	DisplayFrame	rotate() execute() execute(int w, int h) execute(double deg)

	standard 90 or -90 degrees			
FlipCommand	Implements Command interface, when called, user can flip original image horizontally	Command interface, EditingOptions	DisplayFrame	flip() execute() execute(int w, int h) execute(double deg)
CareTaker	Stores mementos and can restore (redo/undo) to another state of the image	Memento CustomImage	DisplayFrame	addMemento(Memento m) getMemento(int i) getSize()
Originator	Creates and saves mementos of the image	Memento CustomImage	DisplayFrame	set(CustomImage newImg) storeInMemento() restore(Memento m)
Memento	Keeps state of mementos of image when image is edited	CustomImage	CareTaker, Originator	getSavedImg()
ImageEditorRemote	Method to return new ImageEditor with the CustomImage to be edited; used as part of the Command pattern	EditingOptions (ImageEditor)	DisplayFrame	getDevice()
EditingOptions	Interface for ImageEditor, used as part of the Command pattern		ImageEditor DisplayFrame	resize(int w, int h) rotate(double deg) saveImage() flip()
ImageEditor	Implements EditingOptions; contains	EditingOptions, CustomImage	DisplayFrame	resize(int w, int h) rotate(double deg) saveImage() flip()

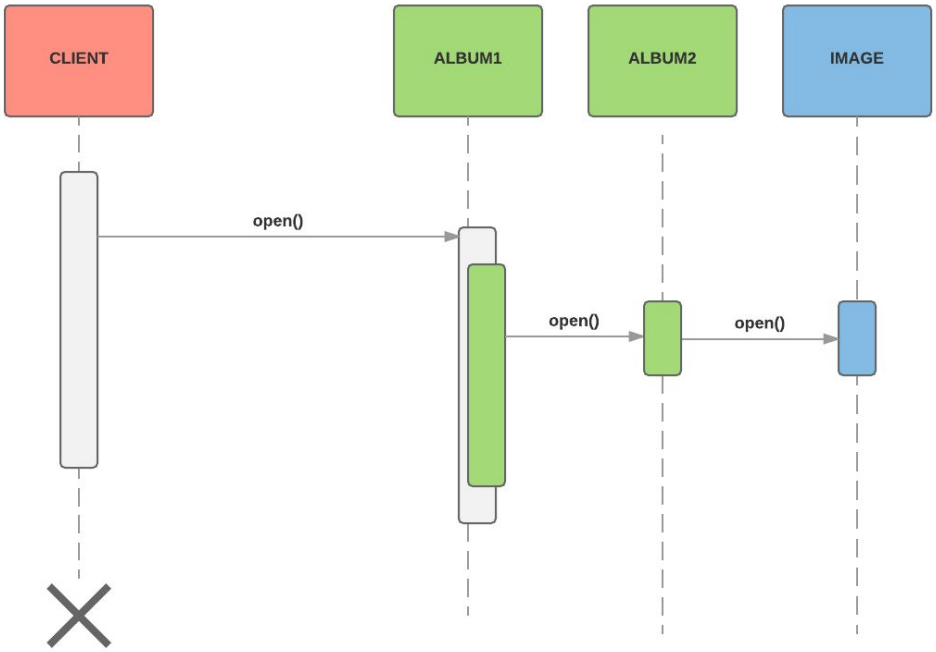
	methods that are involved in editing images			createRotated(BufferedImage image, double deg) createFlipped(BufferedImage image) createTransformed(BufferedImage image, AffineTransform at)
DisplayFrame	Acts as intermediate (facade) between user and application; creates new frame for editing images	CustomImage, EditingOptions, ImageEditor, ImageEditorRemote, OptionButton	ImageViewer	showFrame(CustomImage i)
OptionButton	Part of the Command pattern (invoker); executes command	Command	DisplayFrame	press() press(int w, int h) press(double deg)
CustomImage	Creates ImageIcon and from BufferedImage from image taken from file; multiple images places in a ArrayList to be iterated through and displayed to user		DisplayFrame ImageEditor EditingOptions Memento Originator CareTaker	rightSize() saveImageNew() getBuffered() getWidthImage() getHeightImage() getIcon() getHeight() getWidth() setImageIcon(ImageIcon icon) createBufferedViaImageIcon() setBufferedImage(BufferedImage img)

UML Diagram

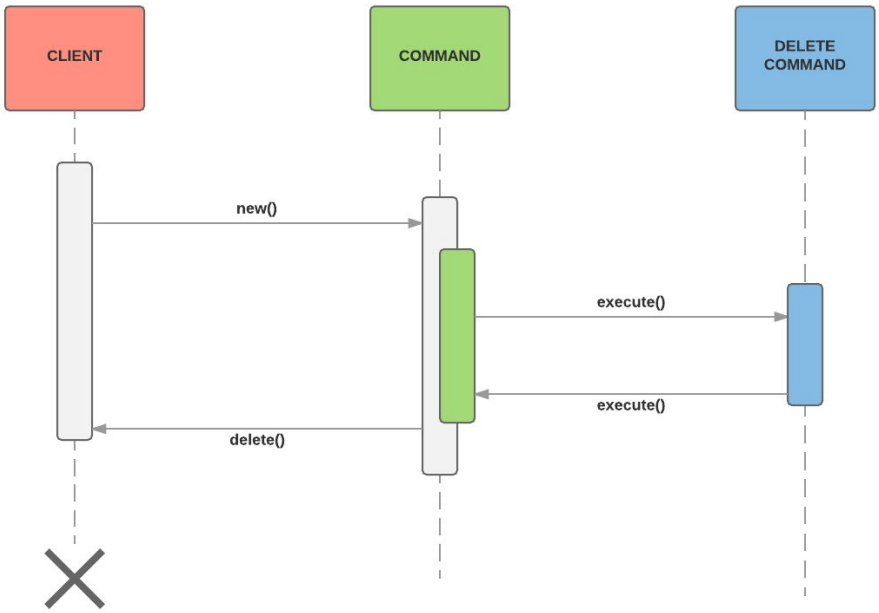

ResizeCommand
- resize()
+ execute()
<<interface>>
Command
+ execute()
EditCommand
- edit()
+ execute()
SaveCommand
- save()
+ execute()
RotateCommand
- rotate()
+ execute()
MirrorCommand
- mirror()
+ execute()
ImageEditorRemote
+ getDevice(CustomImage img) : EditingOptions
OptionButton
Command theCommand
+ press()
+ press(int w, int h)
+press(double deg)
CareTaker
- memento: Momento
+ checkpoint()
+ restore()
<interface>
EditingOptions
+resize(int w, int h)
+flip()
+rotate(double deg)
+saveImage()
Memento
- state State
+ saveState(State s)
+ retreive()
DisplayFrame
+ showFrame(CustomImage i)
- checkEnable()
+ saveMemnto()
ImageEditor
+resize()
+flip()
+rotate(double deg)
+saveImage()
+createRotated(BufferedImage img, double deg) : BufferedImage
+createFlipped(BufferedImage img) : BufferedImage
+createTransformed(BufferedImage img, AffineTransform at) : Buffered Image
GUI uses all of the Commands when editing
creates
Originator
- state State
+ createMomento (State s)
+ reset()
ImageViewer
<<main class>>

Sequence Diagrams

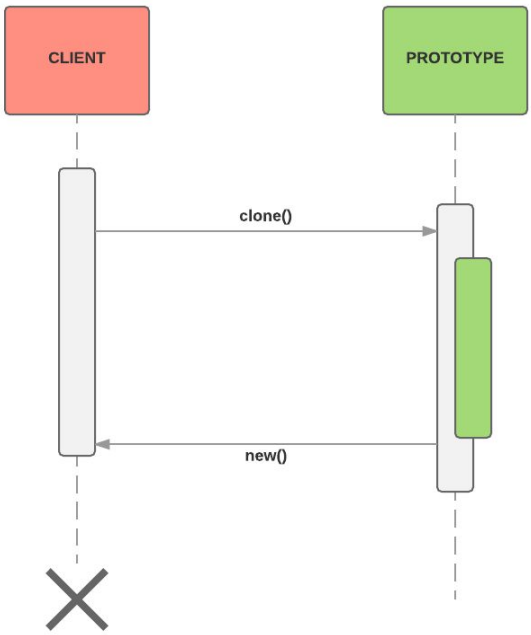
ImageAlbum/ Album/ Image -- Composite Pattern



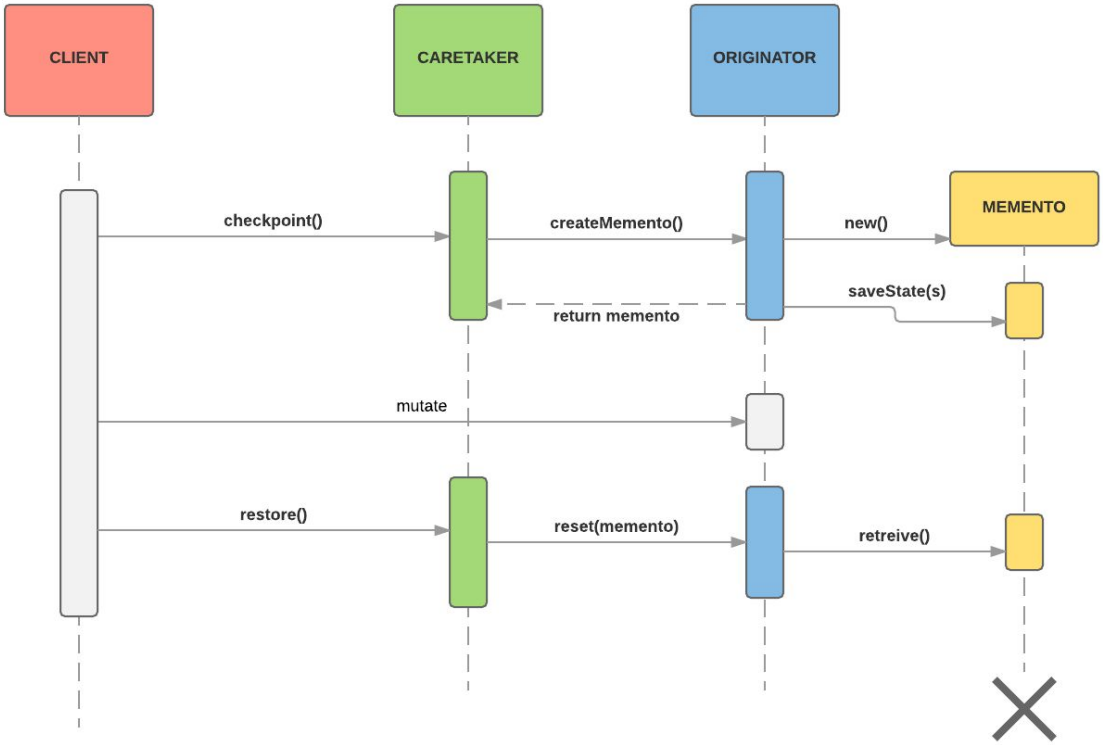
Command (DeleteCommand)



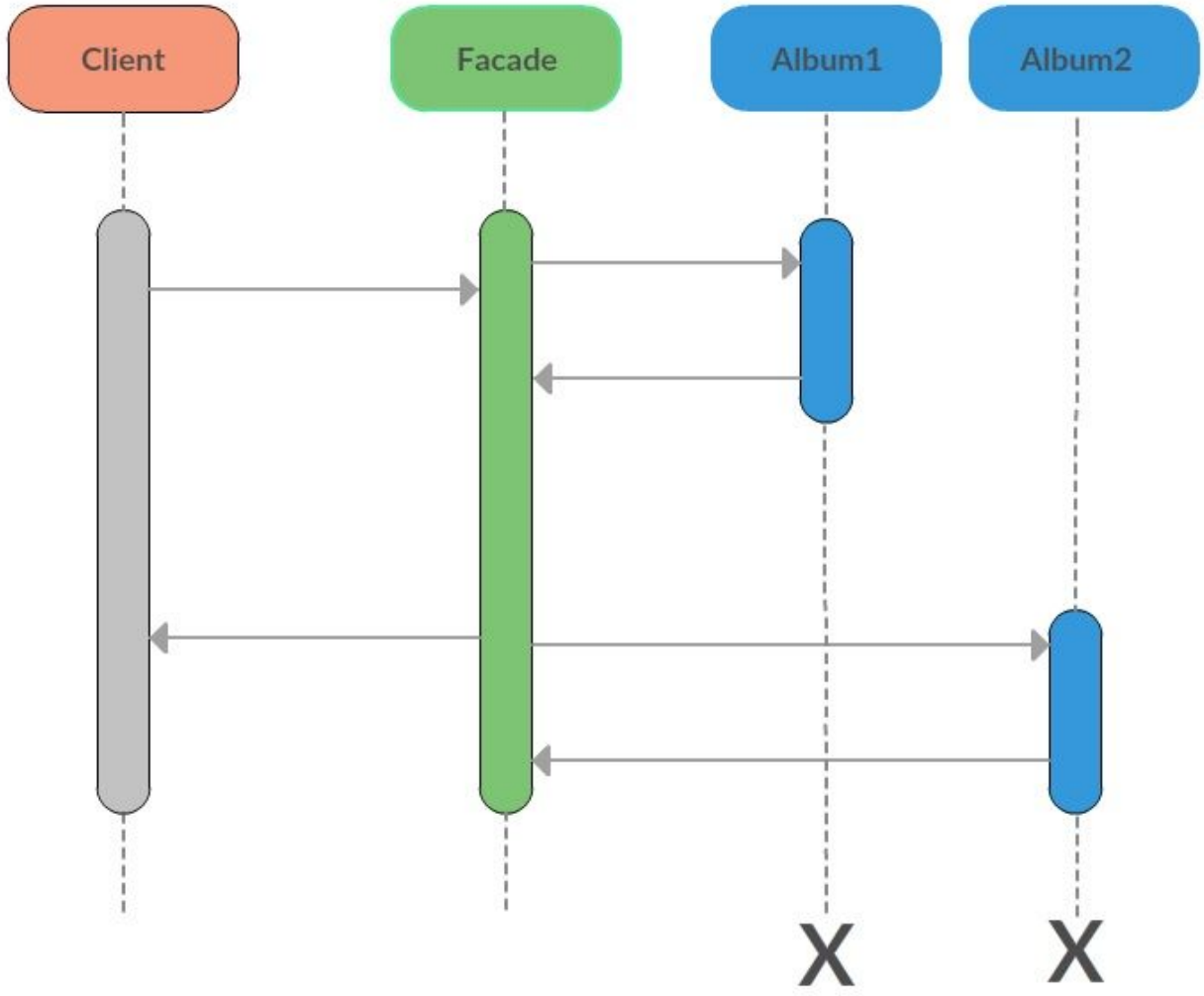
Prototype



Memento



Facade



Patterns

Patterns used in the Project:

Memento - Pattern responsible for all the Undo and Redo functions while editing images.
Command - Responsible for the handling of JButtons in the program's GUI
Facade - Provides a much simpler front end system for the user and hides the complex subsystem of objects.
Observer - Combined with Command pattern; ActionListeners

Dropped Patterns:

Composite - Organizes the images in a hierarchical collection of albums and allows users to go through them.
Prototype (+Factory) - Makes a copy of an image which is then used for editing

Final Remarks and Conclusion

This project proved to be a difficult challenge for our group. For the first few weeks, most of our time working on this project involved us just trying to wrap our heads around what we are suppose to do and how we are going to do it. There are so many ways to go about programming, and there were a lot of elements to this project to take into account. Even as we started coding, the way our group thought about this project changed again and again; but after a certain point, we had to keep moving forward while changing our code around.

We faced many challenges during this project. This project took a lot of time and thinking; sometimes it was hard to figure out where to start. Sometimes, we would get some functionalities to work in a test setting but then have to try to work it into the already existing code. This gave us a lot of trouble. It started to seem like we were going to have to choose between functionality and implementing patterns. However, we did our best to try to do both.

There are several things we would do differently in the future. First, it would be best to - in a more organized thought out way - work on pieces of the project ‘one at a time’ such that it could be pieced together nicely. Some of the problems with the way we went about this project is that there wasn’t a lot of consistency, especially at the beginning, with our objects. For instance, sometimes we were referencing an ImageIcon, sometimes a JLabel, sometimes a CustomImage, and sometimes a BufferedImage.

Overall, this was a very good learning experience. Even though the program did not turn out as we had hoped and had planned, we learned a lot. We learned not only things that were useful for this project itself, but we also learned from our mistakes that can benefit us when working on future projects.