Assignment 2

CS151 Object Oriented Design

Benjamin Garcia

## Part 1: Motivation

The decorator pattern is a pattern in which component objects can be decorated either visually or behaviorally. There is a basic, undecorated object and options to decorate it in a variety of ways. The decorated object can be used in the same way as the undecorated object. This pattern is used so that the basic component class doesn’t have to take on so many responsibilities. It is left to the decorators to add functionality or visual changes. In this project, we are asked to implement the decorator design pattern on orders of prints for a small art photography business. The business sells art prints which can be made in various sizes, on matte or glossy paper, without or with a frame, and without or with one or more mats that come in many colors. There is also the option for prints to be sold digitally through a variety of stock photo agencies.

## Part 2: Solution

The solution I came up with was to have a “Print” interface which defines methods that will be required to be implemented among any instance of the print. These methods were getDescription() (to get the description of the print), printDescription() (to easily print the description to the console for us to verify how the object is decorated), and getCost() (to get a running total cost of the print with all decorations). Then, I made a “BasicPrint” class which implemented the interface and is the most basic, undecorated object of a print containing the attributes “printName” (artistic name), “height”, and “width”. After that, I made 5 different classes that acted as decorators. These decorators also implemented the “Print” interface and took an existing object of type “Print” and essentially wrapped it to add mainly visual changes. These classes were “DigitalPrint”, “FramePrint”, “GlossyPrint”, “MattePrint”, and “MatPrint”. The DigitalPrint decorator attached an agency name to the instance of the Print object passed in as well as an additional cost for selling it digitally. The FramePrint decorator added a frame to the Print object passed in. The GlossyPrint decorator decorated the Print object passed in so that it would be printed on glossy paper. The MattePrint decorator decorated the Print object passed in so that it would be printed on matte paper. The MatPrint decorator added a mat to the Print object which came in a variety of colors defined by an enumeration of “MatColor”. Each decorator modified the description so that it added exactly what was happening to the object when passed through each decorator. Each decorator also added a cost corresponding to each additional decoration. When the printDescription() method was ran on the Print object, it would print out all the information from the BasicPrint as well as any decorations that was added to it along with the total cost of the print with all associated decorations.