Due Nov 26 by 11:59pm **Points** 100 **Available** after Nov 13 at 3pm

Homework 8

Home (https://northeastern.instructure.com/courses/123246/pages/home)

Modules (https://northeastern.instructure.com/courses/123246/modules)

Homework 8: Shapes Photo Album Part 1

Shapes Photo Album Part 1: The Model

The Shapes Photo Album: Part 1

This assignment starts with a context of the application, followed by an outline of what you must achieve. **It is a relatively open-ended assignment** and **does not** spell out explicitly which interfaces, classes, methods and variables you need. Spend time understanding what the application must do, and then design your solution accordingly. This "open ended" approach is intentional, and allowing you freedom in your design decisions is important part of this assignment.

Context

Many of us use our phones (and cameras) to take pictures of friends and family members, and then share those pictures in photo albums - either online or in-person (with actual "photo books").

Pictures can be used to succinctly and effectively illustrate many things. For example, a stationary histogram visually depicts the distribution of some data points. A histogram with moving bar heights visually depicts music (e.g. a music equalizer).

In the next two assignments, we will build an application that helps to create a simple "photo album" from shapes. One of our main goals will be to practice the "separation of concerns" we've covered during lecture. We will support the description of data we want visualized and then have an application that renders the descriptive text properly in a variety of ways.

Concept

Informally using English, we could provide a description for one page of our photo album in a way similar to this:

```
Create red rectangle R with corner at (200,200), width 50 and height
100
Create blue oval C with center at (500,100), radius 60 and 30

R moves to (300,300)
C moves to (500,400)
C changes from blue to green
Take a Snapshot
R changes width from 50 to 25
```

These are descriptive commands that tell us, in sequence what things (shapes) to place on a logical canvas, where to place them, how to color (or resize) them, and when to "take a selfie" (snapshot) of the system.

Our application may then use such a description to:

- "Show" the photo album in various ways. Perhaps we simply want to view the sequence of changes in a spreadsheet, or perhaps we want to see each "snapshot" as a visual rendition of the shapes, on different pages of an electronic album.
- Produce a verbose description for what the photo album will look like (for the visually impaired).
- Other possibilities...

The Shapes Photo Album

We will build this application progressively using the classic Model-View-Controller architecture. **In this assignment (Part 1) we will focus on the model of this application**. There is no starter code for this assignment: you must use the above description and broad ideas to design the interface and implementation of the model. You are deliberately not being told about the specific details of what the controller and views will need or do. You are also not being told about how exactly the application will receive a description from the user (it would not be relevant to the model).

Here are some aspects to think about:

- What does the model represent? Remember: this is a shapes application, so think "application model" here.
- The model should be able to support various kinds of 2D shapes, although currently we have described only rectangles and ovals.
- The model should support adding various kinds of transformations to shapes, such as moving, changing color and scaling.
- As you implement, remember to consider the client perspective (the person or class that is using the model).

How is the photo album seen?

One way the application may show the progression of creating the photo album is to produce a text description of the commands and the model state changes. Here is what a description might look like:

```
Name: R
Type: rectangle
Min corner: (200.0,200.0), Width: 50.0, Height: 100.0, Color:
(1.0,0.0,0.0)

Name: O
Type: oval
Center: (500.0,100.0), X radius: 60.0, Y radius: 30.0, Color:
(0.0,0.0,1.0)
```

(NB: After moving and resizing and changing color of rectangle)

```
Name: R
Type: rectangle
Min corner: (100.0,300.0), Width: 25.0, Height: 100.0, Color:
(1.0,0.0,0.0)

Name: O
Type: oval
Center: (500.0,100.0), X radius: 60.0, Y radius: 30.0, Color:
(0.0,0.0,1.0)
```

(NB: After moving Oval)

```
Name: R
Type: rectangle
Min corner: (100.0,300.0), Width: 25.0, Height: 100.0, Color:
(0.0,1.0,1.0)

Name: O
Type: oval
Center: (500.0,400.0), X radius: 60.0, Y radius: 30.0, Color:
(0.0,0.0,1.0)
```

(NB: After removing the rectangle)

```
Name: O
Type: oval
Center: (500.0,400.0), X radius: 60.0, Y radius: 30.0, Color:
(0.0,0.0,1.0)
```

(NB: Snapshots are similar to "still pictures" or selfies of the shape system)

```
List of snapshots taken before reset: [2022-03-30T11:51:02.174193,
```

```
2022-03-30T11:51:02.246661, 2022-03-30T11:51:02.247210, 2022-03-
30T11:51:02.247794]
```

(NB: Snapshot details)

```
Printing Snapshots
Snapshot ID: 2022-03-30T11:51:02.174193
Timestamp: 30-03-2022 11:51:02
Description: After first selfie
Shape Information:
Name: R
Type: rectangle
Min corner: (200.0,200.0), Width: 50.0, Height: 100.0, Color:
(1.0,0.0,0.0)

Name: O
Type: oval
Center: (500.0,100.0), X radius: 60.0, Y radius: 30.0, Color:
(0.0,0.0,1.0)

Snapshot ID: 2022-03-30T11:51:02.246661
Timestamp: 30-03-2022 11:51:02
Description: 2nd selfie
Shape Information:
Name: R
Type: rectangle
Min corner: (100.0,300.0), Width: 25.0, Height: 100.0, Color:
(1.0,0.0,0.0)

Name: O
Type: oval
Center: (500.0,100.0), X radius: 60.0, Y radius: 30.0, Color:
(0.0,0.0,1.0)

Snapshot ID: 2022-03-30T11:51:02.247210
```

```
Timestamp: 30-03-2022 11:51:02
Description:
Shape Information:
Name: R
Type: rectangle
Min corner: (100.0,300.0), Width: 25.0, Height: 100.0, Color:
(0.0,1.0,1.0)

Name: O
Type: oval
Center: (500.0,400.0), X radius: 60.0, Y radius: 30.0, Color:
(0.0,0.0,1.0)


Snapshot ID: 2022-03-30T11:51:02.247794
Timestamp: 30-03-2022 11:51:02
Description: Selfie after removing the rectangle from the picture
Shape Information:
Name: O
Type: oval
Center: (500.0,400.0), X radius: 60.0, Y radius: 30.0, Color:
(0.0,0.0,1.0)
```

In short, it first describes the shapes that are part of the photo album and their details. Next it describes model state changes as the shapes are transformed/moved/removed. Each snapshot is a "freeze frame" of the model state - think of it as a "system selfie" - the picture captured is that of the shapes in their then-current locations (and state) and is analogous to a "page" in a photo album. You may think of the first part of this output as a “read-back” of the model state changes, perhaps for devices that cannot show the photo album visually, or for users who are visually impaired who have screen readers. The snapshots are historical "save points" that can be retrieved irrespective of the model state.

What to do

1. **Design** a model to represent a shapes photo album. This model may consist of one or more interfaces, abstract classes, concrete classes, enums, etc. Consider carefully what operations it should support. **Create a high-level UML class diagram to represent your design**.

2. **Think** about and include in the model interface operations that you think are relevant and the model should offer.
3. **Document** your model well. Be sure to document clearly what each method does, what purpose it serves and why it belongs in the model.
4. **Implement** your model.
5. **Test** your model.
6. **Implement** the text output rendering of your model according to the format given in the previous section (*How is the photo album seen?*), so we can visualize your data.

What to submit

Submit any files created in this assignment, along with a text README file explaining your design, and your UML class diagram. Your README file and UML should give the graders quick insight into the purposes are for every class, interface, etc. that you include in your model. Verbosity is not a virtue here, so clear, concise documentation is better than reams of verbiage. Think of your graders as your users; help them so that they can quickly get a high-level overview of your solution and code.
NB: *The UML diagram and README do not replace the need for proper Javadoc!*

Grading

Since this is an open-ended assignment with you making design trade-off decisions, there are no server-side tests. For this assignment, you will be graded on

- the design of your model interface(s), in terms of clarity, flexibility, and how plausibly it will support needed functionality;
- the appropriateness of your representation choices for the data (please comment on why you chose the representation you did in the code);
- the forward thinking in your design, in terms of its flexibility, use of abstraction, etc.
- the correctness and style of your implementation, and
- the comprehensiveness and relevance of your test coverage.