CS 4773
Homework Assignment 3: A drawing program without drawing
Due: In class, Tuesday, October 25

**You may work with one partner on this assignment**

You may submit as many times as you wish up to the deadline. Only the last submission by the deadline will be graded. However, all Blackboard submissions for your team must be from the same team member.

**DESCRIPTION**
In this assignment, you will design and build a program that stores an arbitrary sequence of commands that manipulate and use a drawing list. A drawing list is used to produce a graphical scene, e.g., [create a red rectangle, move it to coordinates (5,5), create a blue circle, move it to coordinates (10,2), draw the red rectangle, draw the blue circle]. While it would be fun to actually draw the shapes that we will store and manipulate in our list, we shall instead focus only on the creation and maintenance of the drawing list.

The commands that your program will be able to process are:

CREATE RECTANGLE [w] [h]
- [w] is the required width of the rectangle
- [h] is the required height of the rectangle
- The origin of a created rectangle is the upper left-hand corner at (0,0)
- The color of a created rectangle is Red

CREATE CIRCLE [r]
- [r] is the required radius of the circle
- The origin of a created circle is the center at (0,0)
- The color of a created circle is Blue

SELECT [any shape number from 1..n]
- Selects a shape to use with commands MOVE, DRAW, COLOR, DELETE
- n is the total number of shapes available
- At most one shape can be selected at a time
- Calling SELECT on a shape that does not exist in the list will produce the following error message "ERROR: invalid shape for SELECT"

MOVE [x] [y]
- Moves the currently selected shape's origin to the specified (x,y)
- If no shape is selected then output "no shape selected" to the console

DRAW
- Outputs the currently selected shape information to the console, one shape per line. E.g.,
  ```
  Rectangle, Color: Red, Origin: (5,5), Width: 10, Height: 7
  ```
  Or:
  ```
  Circle, Color: Blue, Origin: (10,9), Radius: 12
  ```
- If no shape is selected then output "no shape selected" to the console

COLOR [c]
- Sets the color of the currently selected shape.
- [c] is one of the following valid colors:

    - `Red, Blue, Yellow, Orange, Green`
- If no shape is selected then output "no shape selected" to the console

DELETE
- Removes the selected shape from the scene
- If no shape is selected then output "no shape selected" to the console
- The deleted shape is no longer selected

DRAWSCENE
- Draws all shapes in the scene using their current origins and colors
- Output for each shape is the same as using the DRAW command with one shape per line

UNDO
- "Undoes" the most recent command in the drawing list
- UNDO can be called after UNDO, which undoes the next most recent command in the drawing list
- Undoing a SELECT command sets the currently selected shape to the previously selected shape (which could be "no shape")
- Undoing a MOVE command restores the selected shape to its previous position
- Undoing a COLOR command restores the previous color
- Undoing a CREATE command deletes the shape
- Undoing a DELETE restores the previously deleted shape to the scene. It should also restore the currently selected shape to the restored shape
- Undoing DRAW or DRAWSCENE has no effect
- The UNDO command is not stored on the drawing list (i.e., you cannot UNDO an UNDO)

You will start with an empty drawing list. Each command is processed as it is received, which means the shapes in the scene will be modified as certain commands are processed. Note that UNDO can undo the previously processed command, so you will need to come up with a design to accommodate that functionality and you need to use the Memento pattern to provide the necessary support for undoing the current configuration of shapes.

<u>Running your program</u>
Your program should take the name of a text file of commands as its only command line argument:

```
java -cp <path-to-jar-file> <your-class> <file-name>
```

Included with this assignment description is a sample input file and a sample of the corresponding output (which should be written to the console).

Your object-oriented design must include the Strategy, Command, and Memento patterns. You also need to create and include a UML class diagram that includes all of the classes in your project and indicates where the 3 patterns are implemented.

Use our clean-coding standards. Keep your class sizes <= 200 total lines of code.

**DELIVERABLES**
1. Make this assignment its own Maven project called cs4773-hw3
2. Put your class diagram named assignment3UML in your project. Clearly identify your Strategy, Command and Memento patterns

3. Zip up the following in a file called hw3.zip
    a. Your src directory
    b. Your class diagram
    c. Your pom file
4. Print and hand-in your class diagram and source code created by the DirToPDF tool. Make sure that your name and abc123 ID are on your class diagram

**RUBRIC**

-10 points: Not handing in a listing created by the DirToPDF tool
30 points: All functionality implemented correctly
30 points: All patterns implemented correctly
20 points: Domain objects and responsibilities are well-designed
10 points: Code is readable. No Javadoc is required.
10 points: UML class diagram