Final Project

Introduction

The Final Project gives you the opportunity to apply the object-oriented software development skills you have acquired during the course to a **significant** but **moderately-sized** application. The following sections describe the general behavior of the application and the deliverable requirements for the project.

Starter code can be found on D2L > Content > Final Project.

Overview

We will be building an “MS Paint”-like application in Java called JPaint. It will have the following features:

- Pick a shape
  - Ellipse
  - Triangle
  - Rectangle
- Pick a primary color
- Pick a secondary color
- Select shading type (outline only, filled-in, outline and filled-in)
- Click and drag to draw a shape
- Click and drag to select shapes
- Click and drag to move selected shapes
- Copy selected shapes
- Paste copied shapes
- Delete shape
- Undo last action
- Redo last action
- Group selected shapes
- Ungroup selected shapes
- Selected shapes have dashed outline
- Shapes display on click + drag

I am providing the GUI. You should need to write minimal GUI code. You will need to call into the Java API for drawing shapes. You will also need to write some handlers for click and drag events.

Features by Sprint

Sprint 1 (Ends on Week 4 at 5:30 pm)

- Draw a filled-in Rectangle
    - When mouse is click-and-dragged while in Draw mode, a Rectangle will display on the screen. The Rectangle will match the direction and size of the mouse movement.

Sprint 2 (Ends on Week 6 at 5:30 pm)

- Draw Rectangles, Ellipses, and Triangles
- Draw shapes with various colors
- Draw shapes with various shading types
    - Outline Only – Only shape outline will be drawn. Use Primary Color to draw this.
    - Filled-In – Only the inside of the shape will be drawn – there will be no visible outline. Use Primary Color to draw this.
    - Outline and Filled-In – Both the inside and the outline will be drawn. Use Primary Color for the inside and Secondary Color for the outline.
- Select a shape.
    - In Select mouse mode, select any shapes that are touched by the invisible bounding box created by clicking and dragging to select. You can use (and share on D2L) a Collision detection algorithm that you find. The selection can be imprecise; when selecting, assume any shape (e.g. ellipse or triangle) have an invisible bounding box that surrounds the shape. You can use that bounding box for your collision detection calculation (this is much easier for you!).
    - If you click a single point on the canvas or on a shape while in Select mode, that shape should be selected. This is the default behavior for collision detection – this is easier for you!
    - At this point, nothing visible has to happen.
- Move a shape
    - In Move Mouse Mode, clicking and dragging will offset any Selected shapes by the amount your mouse moves.

Sprint 3 (Ends on Week 8 at 5:30 pm)

- Copy

    - Adds selected shapes to the “clipboard” (just jam them in a list somewhere).
- Paste
    - If there is anything copied, paste it on the screen somewhere. You can paste it at origin (0, 0), the middle of the canvas, or somewhere else that makes sense. Do not paste to the same location; this will not be visible and will get marked as “not working”.
- Delete
    - Deletes the selected shape(s)
- Outline Selected Shapes
    - Shapes that are selected will display with a dashed outline around them. These will need to be offset slightly so they don’t overlap the shape. Move the start point up and to the left 5px and add 10px to the height and width. You can adjust these values depending on personal preference.
    - The outline must be the same shape as the shape that’s selected

Sprint 4 (Ends on Week 10 at 5:30 pm)

- Undo
    - Undo the last operation. The following need to be Undoable:
        - Draw
        - Move
        - Paste
        - Delete
        - Group
        - Ungroup
- Redo
    - Redo the last operation. See undo.
- Group
    - Clicking this button will cause all Selected shapes to operate as a group.
    - Shapes grouped together should be operated on as if they were one shape.
    - To select a grouped shape, any part of the invisible bounding box around the shapes in the group can be selected.
- Ungroup
    - Any selected shapes that are grouped shapes will no longer be grouped.

Non-functional requirements

- Must use at least 5 design patterns. Two may be the same kind of pattern.
- Report (see below for details).
- The application must be written in Java using the Java2 SDK 1.8 or higher.
- Only features and capabilities that are part of the Java2 SDK may be used in the application. *No third-party software* such as BlueJay or JBuilder class libraries or COM/CORBA components.

Suggestions and Hints

Use D2L to talk to your fellow students (and me), bounce ideas off each other, etc. Collaboration is encouraged! **However, the code you turn in must be your own.**

Use Dependency Injection and follow SOLID practices! This will help you keep your code manageable, make it easier to unit test, and allow you to refactor and replace modules more easily.

Use Source Control and an IDE! Source Control will help you go back in time when you realized you really screwed up and need to undo. Check in frequently. An IDE is an incredibly useful tool for refactoring. You can extract code into a method or rename a file and all references very easily. It will also help you organize your classes and interfaces.

A couple notes about the starter code. Main provides you with an idea of how to draw a shape outline, a shape that's filled-in, and a shape that has an outline and is also filled-in. It also shows how to display a selected shape. Commented out is code that is needed to wipe the canvas clean so you can repaint everything – don't get fooled by the name of the method. When working with PaintCanvas/Graphics2D, don't pass the Graphics2D into constructors. Pass the PaintCanvasBase in instead and only get the Graphics2D when you are about to draw. The reason for this is the Graphics2D object can get destroyed by external code, causing things to not show up on the canvas.

Don't procrastinate. ☺

Report

Twenty points of this project grade is based is based on a written report. Although no specific style guidelines are being enforced, the report must be presented in a neat, legible, and consistent format. It is not meant to be long. Most reports are approximately 3 pages (including diagrams).

All text must be typed. Diagrams must be drawn on the computer using a program like http://draw.io or Visio. **Do not use diagrams generated from your IDE.** They never come out right and if I can't read the diagrams, I will take off points. The diagrams should conform to the UML notational conventions presented in class. **Do not hand-draw the diagrams.**

The written report should be structured as follows:

1. **List of missing features, bugs, extra credit, and miscellaneous notes.** List features that **don't** work and make me aware of any bugs in the code. List any extra credit or any other miscellaneous notes as well. Essentially, **list anything I should know when grading.**
2. **Notes on design**. Indicate five design patterns used in your project. There should be some variety in patterns you discuss – you can't discuss the same pattern more than twice.

   For each pattern, in a **few sentences**, describe the classes involved, why you chose to implement that pattern, and what problem it solved. Include a design class diagram as well. Be sure to include all significant class relationships: realization, specialization, and association. Show associations as dependencies, aggregations or compositions when appropriate. Show attributes and methods *only* if they are crucial to understanding the class relations (e.g. for Dependency relationships). *You do not need to show all fields and methods!*

3. **Successes and Failures.** Discuss what went right with your project? What went wrong? Note design issues that arose during development, such as specific decisions, use of design patterns, failures, successes, etc. This should take 1 page or less.

Deliverables

Each sprint end, you will upload your code to D2L with the following requirements. The Report only gets submitted at the end of the fourth sprint.

- Submit a `jar/zip/rar/7z` file containing your src folder and report.
- Test your `jar/zip/rar/7z` file by downloading it from D2L and unzipping it into a fresh directory. Make sure everything works. **IF WHAT YOU TURN IN DOESN'T COMPILE, YOU WILL GET A 0%!**
- Your report must be a *single* file in .PDF or .DOCX format. This requires that you somehow get diagrams into the report. Microsoft word and OpenOffice are very good at this. Please use one of them. **IF I CAN'T OPEN YOUR REPORT, YOU WILL GET A 0%.**

Grading

Grading will follow these guidelines:

- **Application: 80 points**
  - Satisfies requirements.
  - Quality of code – follow the SOLID principles!
- **Report: 20 points**

Academic integrity

I do encourage collaboration; however, all submitted work must be your own. If the work was duplicated, it will be reported to the university as an Academic Integrity violation. I don't mind copying and pasting of utility functions.

As general rules of thumb, I would say the following guidelines dictate whether sharing is allowed or not:

- It is functionality you would reasonably expect to find on Stack Overflow (like how to draw a shape)
- It is some sort of mathematical algorithm (like determining the math for drawing a triangle).
- It is an abstract description of how your code is working (like "I used the factory pattern to create the individual shapes in x class").

I want to avoid being heavy-handed about this. I would prefer you collaborate than not due to being afraid to violate these rules. When in doubt about whether you can share something or not, send me an email and ask. In 99% of cases where you aren't sure, I will likely say it's acceptable.