SE 450
Summer 2021
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 > 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 selected shapes
- Undo last action
- Redo last action
- Group selected shapes
- Ungroup selected shapes
- Selected shapes have dashed outline

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)

- Draw a filled-in Rectangle
    - Click and drag while in Draw mode – a Rectangle will display on the screen on mouse release. The Rectangle will match the direction and size of the mouse movement. Rectangle does not need to display while clicking and dragging – it will suddenly appear on the screen only once the mouse is released.
- Undo/Redo Draw
- Have at least one design pattern implemented

Grading Notes:

For grading purposes, the order in which shapes appear on canvas don't matter. If you draw one shape on top of another, undo, then redo, and the order changes, that's okay!

Sprint 2 (Ends on Week 6)

- 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 Discord) a Collision detection algorithm that you find. The selection can be imprecise; when selecting, assume any shape (e.g. ellipse or triangle) has 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 a shape while in Select mode, that shape should be selected. If you click a single point on the canvas or select an empty area, the selected shapes should be deselected. This is the default behavior for collision detection and shouldn't require any modification – this is easier for you!
    - You should be able to click and drag into any part of a shape to select it – it does not need to be completely surrounded

    - 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.
    - Moving should not deselect any shapes
- Undo/Redo Move
- Have at least three design patterns implemented

Grading Notes:

- The ability to move a shape is dependent on the ability to select a shape.
- Shape selection *must* include the ability to click and drag to select multiple shapes at once. You should not be able to click on shapes one at a time to select
- You can move by clicking and dragging anywhere on the screen, you don't need to click and drag on the highlighted shape(s).

Sprint 3 (Ends on Week 8)

- Copy
    - Adds selected shapes to the "clipboard". Nothing visible occurs on the screen.
    - Copying should not deselect shapes
- Paste
    - If there is anything on the clipboard, paste it on the screen somewhere. You can paste it at origin (0, 0), some offset of the original shapes, or somewhere else that makes sense. Do not paste to the same location as the original shapes; you will not be able to see the pasted shapes and it 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
- Undo/Redo Paste and Delete
- Have at least four design patterns implemented

Sprint 4 (Ends on Week 10)

- 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.
  - The selection box should display along the boundaries of the group of shapes, NOT the individual shapes o Groups can be part of other groups
- Ungroup
  - Any selected shapes that are grouped shapes will no longer be grouped.
  - If a selected group is comprised of one or more groups, only the outermost group is ungrouped
- Undo/Redo Group and Ungroup
- Have at least five design patterns implemented

Non-functional requirements

- Must use at least 5 unique design patterns. Please note you won't get credit for using MVC as a design pattern. Further, you won't get credit for using the Java SDK implementations of patterns (e.g. Observer/Iterator).
- Must use Source Control (see details below).
- Report (see below for details).
- The application must be written in Java using the Java SDK 11 or higher.
- Only features and capabilities that are part of the Java2 SDK may be used in the application. *No third-party libraries may be added to the project.*

Suggestions and Hints

Use Discord 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. See "Graphics tips.pdf" for more information on how the Java graphics system works

The "Factory pattern" is not a specific pattern and it will not be given credit. Specific patterns such as Abstract Factory, Factory Method, and Static Factory are acceptable to use.

Don't procrastinate. ☺

Source Control Guidelines

Please note, failure to adhere to these guidelines may result in a 0 on the project!

1. Your repo must remain PRIVATE throughout the quarter
2. I will provide instructions for giving me access to view your repo. You MUST give me access by the last check-in or you will get a 0 on the project.
3. Your repo MUST contain source code. In other words, it must not consist solely of .class files or .zip/.rar/etc. files and must contain the complete set of .java files used in your project.
4. You must have at least 4 **substantive** commits per sprint. I will not, for example, count commits that update the readme file.

Report (Due Week 10)

Twenty points of this project grade are 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. Link to GitHub repo. This must be in a text format (as opposed to a screenshot).
3. **Notes on design**. Indicate five design patterns used in your project. I will give credit for up to five unique patterns.

   For each pattern, in a **few sentences**, describe the classes involved, why you chose to implement that pattern, and what problem it solved. For each pattern, include a UML class diagram that represents how the pattern is used. 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!***

   If I cannot find the pattern in your code or the pattern in your code isn't actually used anywhere in your application, you will not get credit.

4. **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

At each sprint end, you will upload your code to D2L with the following requirements. The Report gets submitted separately in the final week.

Project Check-ins:

- Submit a ZIP produced by gradle. DO NOT build a zip by hand!

- Test your file by downloading it from D2L and unzipping it into a fresh directory. Make sure everything works. **IF YOUR TURNED-IN CODE DOESN'T COMPILE, YOU WILL GET A 0%!**

- In D2L, in the comments section when you submit, you MUST indicate which design pattern(s) you have added and which classes are involved. Further, you should ONLY describe design patterns added during this sprint. In other words, I would expect one design pattern for check-in one, two for check-in two, one for check-in three, and one for check-in four (so a total of five over the quarter). **Failure to do so will result in a 0 for the check-in.**

Final Check-in:

- **Your report must be a *single* file in .PDF 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%.**

- **If your report doesn't have a link to your GitHub or I can't access your GitHub repo, you will get a 0%.**

- You are not submitting code for this check-in, only the report. The code from check-in 4 is used for grading.

- Plagiarism in the code is ONLY checked at this time.

Grading

Grading will follow these guidelines:

- **Application: 80 points**
  - Satisfies requirements (30 pts).
  - Quality of code/design patterns – follow the SOLID principles! **(50 points)**

- **Report: 20 points**

Academic integrity

I do encourage collaboration; however, all submitted work must be your own. Put another way, 95+% of all code written in the project – except for the starter code – MUST have been written by you. If the work was duplicated, it will be reported to the university as an Academic Integrity violation.

As general rules of thumb, I would say the following guidelines dictate what you can use from external sources:

- 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 static factory pattern to create the individual shapes in x class”).

When in doubt about whether you can share something or use something from an external source, **send me an email and ask**. You must get my permission to use additional sources.