Quarter 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 > Quarter 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
- Whiteboard exercise/Discussion post

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 D2L) 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 two design patterns implemented
- Whiteboard exercise/Discussion post

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
- Whiteboard exercise/Discussion post

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
    - 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 outer-most group is ungrouped
- Undo/Redo Group and Ungroup
- Have at least six design patterns implemented
- Whiteboard exercise/Discussion post **(DUE WEEK 9)**

Non-functional requirements

- Must use at least 6 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).
- The application must be written in Java using the Java SDK 1.14 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 group's 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 shape that has been selected. 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.

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.
5. I highly recommend (but will not take points off) not committing .class and other auto-gen files to source control. You can let git take care of this for you by creating a .gitignore file.

Project Check-ins/Discussion Posts

Check-ins should be posted in the relevant discussion forums by the weeks listed below. There is no penalty for submitting posts early. Due dates are posted on the Submissions page in D2L.

1. Check-in 1 (Due by Week 4)
   Record a quick screenshare and answer the question: How are you drawing a shape? Feel free to use a Sequence Diagram or just walk through the code. Build and run your project in the video and show that you can draw shapes. Post in the Check-in 1 D2L forum.

2. Check-in 2 (Due by Week 6)
   Record a quick screenshare and answer the following questions:
   - Using UML class diagrams, show two of your implemented design patterns.
   - Talk through where you are using the patterns and how it makes your solution more extensible, flexible, and/or maintainable.
   - Demonstrate you understand the composition and usefulness of the pattern in question. Feel free to use analogies to help explain its usefulness.

   Post in the Check-in 2 D2L forum.

3. Check-in 3 (Due by Week 8)
   Record a quick screenshare and answer the following questions:
   - Using UML class diagrams, show two of your implemented design patterns.

  - Talk through where you are using the patterns and how it makes your solution more extensible, flexible, and/or maintainable.
  - Demonstrate you understand the composition and usefulness of the pattern in question. Feel free to use analogies to help explain its usefulness.
  - Show different patterns than from any previous check-in.

   Post in the Check-in 3 D2L forum.

4. Check-in 4 (**Due by Week 9**)
   Record a quick screenshare and answer the following questions:
    - Using UML class diagrams, show two of your implemented design patterns.
    - Talk through where you are using the patterns and how it makes your solution more extensible, flexible, and/or maintainable.
    - Demonstrate you understand the composition and usefulness of the pattern in question. Feel free to use analogies to help explain its usefulness.
    - Show different patterns than from any previous check-in.

   Post in the Check-in 4 D2L forum.

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. The diagrams should conform to the UML notational conventions presented in class.

Alternatively, you may use a whiteboard application to draw out the diagrams as you talk through them in the screenshare.

README File

All text must be typed.

This written document is essentially a guide for grading. It 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.** Bullet lists are preferred.

2. **Link to GitHub repo**. This must be in a text format (as opposed to a screenshot).
3. **List of Design Patterns**. Indicate six design patterns used in your project. I will give credit for up to six unique patterns. For each pattern, list the classes/interfaces involved. I should be able to easily find and grade your pattern. 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.

Deliverables

Final Check-in:

- Submit a ZIP (or RAR or 7z) file containing your src folder and README. I only need one submission per group. Both the src folder and the README should be in the root of the zip file. The src folder should not be zipped individually.
- 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%.
- If your report doesn't have a link to your GitHub or I can't access your GitHub repo, you will get a 0%.
- Plagiarism in the code is ONLY checked at this time.

Grading

Grading will follow these guidelines:

- **Design Patterns: 60 points**
- **Satisfies requirements: 35 pts**.
  - Includes quality of code – follow the SOLID principles!
- **README file: 5 points**

Group Expectations

1. Both group members are expected to contribute approximately equally.
2. Group members must make time for this project and collaboration. If you cannot commit to this, drop the class.
3. If a group member is unresponsive, please reach out to me.

Academic integrity

I do encourage collaboration via the D2L discussion board; however, all submitted work must be your group’s own. Put another way, 95+% of all code written in the project – except for the starter code – MUST have been written by your group. 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.