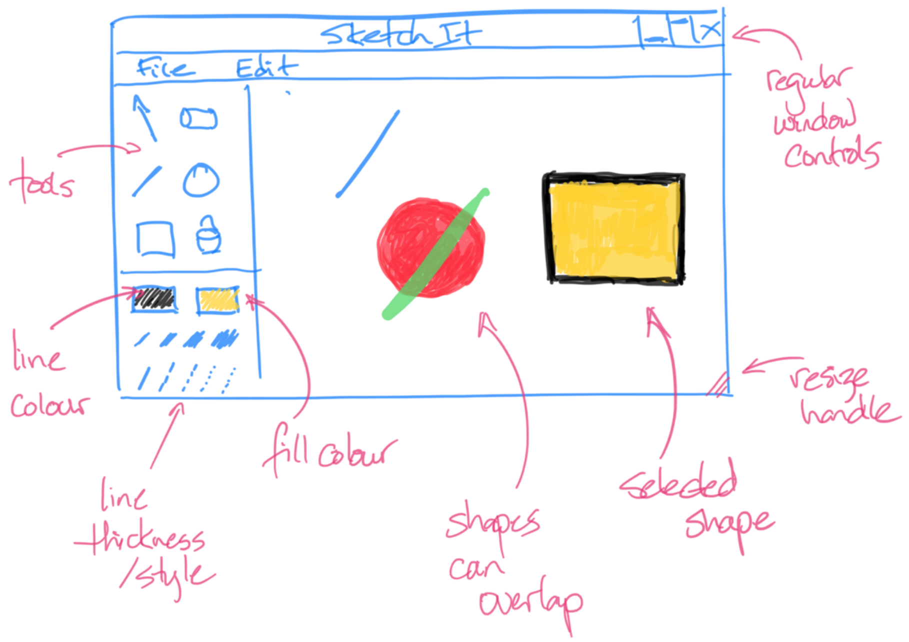
# A3: Widgets & Layout (Java) -- SketchIt

## Synopsis

In this assignment you will implement a vector-drawing program in Java/Java FX. Your program will allow a user to select shapes to draw, set their properties, and draw them on-screen. This type of program would typically be used to draw simple diagrams.

This is a rough mockup that illustrates the features that you will implement, and the basic layout that you must use.



## Objectives

Don't save as image.

- Construct an interface using Java FX components and layouts.
- Support interaction with a drawing canvas, where users can draw shapes using graphics primitives.
- Handle mouse and keyboard interaction, and implement standard interactive features listed.
- Structure your application using the model-view-controller design pattern.

## Features

The main area of your drawing program is a **canvas** that supports drawing shapes. Users can select a shape from the list on the side, specify colour, border thickness and style, then click and drag the mouse on the canvas to draw that shape. A shape "preview" is shown as the mouse is dragged, and the shape is complete when the mouse button is released (i.e. the first click sets the starting position, and the drag operation sets the width of the circle, or creates a line, depending on the shape being drawn, and the shape is completed when the mouse button is released).

Your drawing program will have the following layout and features:

- A **menu bar** containing the following menus and menu items:
  - File:
    - **New**: create a new blank drawing (prompt the user to save if the current drawing is unsaved).
    - **Load**: load a drawing that you previously saved (which is scaled up or down when it loads to fit the current screen resolution).
    - **Save**: save the current drawing using a file format that you determine.
    - **Quit**: exit the application (prompt the user to save before exiting).
- A **tool palette** on the left-hand side, supporting the following tools:

  - **A selection tool**: this allows the user to select a shape that has been drawn. To select a shape, the user should click this tool, then click on an existing shape (resulting in some visual indication to the shape itself to indicate that its been selected). Pressing ESC on the keyboard (or clicking an empty part of the canvas) will clear shape selection. Selecting a shape will cause the colour palette, line thickness and style to update their state to reflect the currently selected shape (e.g. if I select a red circle with the largest line thickness, the colour palette should change to red, and the line thickness should change to the largest line to match the selected shape). Changing colour, line thickness or line style when a shape is selected will update the shape to the new values.
  - **An erase tool**: click on this tool and then click on a shape to remove it. Pressing DEL should also delete the selected shape.
  - **A line drawing tool**: select this to draw a line using the selected properties.
  - **A circle drawing tool**: select this to draw a circle at the point indicated using the selected properties.
  - **A rectangle tool**: select this to draw a rectangle using the selected properties.
  - **A fill tool**: select this tool, and click on a shape to change it's fill colour to the currently selected value.
- A **line colour icon** that graphically displays the currently selected line color. Clicking on this object will bring up a color chooser dialog that lets the user pick a color. This colour will be used as the border colour for any new shapes that are drawn (or the ONLY colour in the case of a line).
- A **fill colour icon** that graphically displays the currently selected fill color. Clicking on this object will bring up a color chooser dialog that lets the user pick a color. This colour will be used as the fill colour for any new shapes that are drawn (it will NOT be used if a line is being drawn).
- A **line thickness palette**: a graphical display of at least three line widths that the user can select. Selecting a line width will set the line thickness or border thickness for any new shapes drawn.
- A **line style palette**: a graphical display of solid or dotted lines with at least least three choices. Selecting a line style will set the line style or border style for any new shapes that are drawn.

The following functionality should be supported:

- The user should be able to select any shape and move it by dragging it with the mouse. Shapes can overlap, with the more recently drawn shape overlapping the older shape (hint: you order them back-to-front as they are created).
- Your interface should clearly indicate which shape, if any, is selected. You may use thicker borders around the tool, or some other visual indicator that you choose. When a shape is selected, the properties should also change to reflect the shape's properties. Changing the properties of a selected shape should update that shape's properties.
- Your interface should enable/disable controls if appropriate (e.g. if an option is invalid at any given time, you should disable the control).
- Your application should support dynamic resizing of the application window, so that the application adapts to different window sizes and resolutions. The tool palettes should adjust based upon available space, and the canvas should resize itself and its contents to fit the new space (hint: this implies that you need to scale and adjust the shapes as well). The exact implementation is a visual design decision left to you, but the application should be equally usable at all resolutions. You may set a "reasonable" minimum size for your application window (e.g. 640x480), but there should be no limit to the maximum size.

## Choose Your Own Feature! (Select one or more features to implement from the list below, totalling 10%).

- **Scale shapes**: the ability to change the scale/size of any shape by selecting it, then grabbing a corner of the shape, and dragging to increase/decrease it's size (i.e. direct manipulation) (5 marks).
- **Rotate shapes**: the ability to grab a corner of a shape and rotate it in real-time (i.e. direct manipulaton). (5 marks)
- **Group shapes**: select multiple shapes, then group them together so that they can be moved as a single entity. If you support resizing or rotation, you must also apply those operations to the group (5 marks).
- **Zoom and pan**: support zooming the canvas in and out, scaling the contents and showing scrollbars as needed. This also requires the ability to pan around the screen if it's zoomed in larger than the viewport (i.e. if you have scrollbars) (10 marks).
- **Customizable toolbar**: support moving and redocking the toolbar to a different side of the screen (5 marks), or drag-to-reorder any of the toolbar elements (5 marks).
- **Multiple document**: add tabs, so that the user can have multiple drawings open simultaneously! You would need to add File-New Tab and File-Close Tab functionality, and handle loading and saving from each tab as well. (10 marks)
- **Cut-copy-paste**: allow users to cut/copy/paste shapes within your application (e.g. to create a copy of a shape). This includes adding standard Edit-Cut, Edit-Copy, Edit-Paste menu items. If a shape is copy-pasted, the new copy should be fully interactive, just as if it had been drawn. You do NOT need to support pasting from an external application (10 marks).
- **Undo-Redo**: allow users to undo the last action (5 marks) or have a full implementation that lets them undo-redo multiple steps. Add functionality under Edit-Undo and Edit-Redo menu items (10 marks).

**25%**
Appropriate use of widgets to meet the requirements above.
**10%**
Additional features from the list above.

## Versions

- 1.0. Feb 5, 2020. Initial draft.
- 1.1. Feb 6, 2020. More additional features. Sketched new mockup.

---

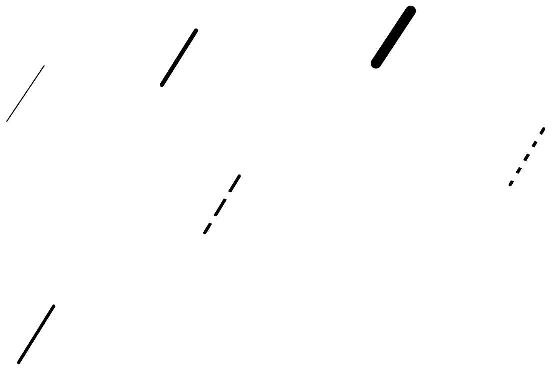
CS 349 User Interfaces (Winter 2020)

Cheriton School of Computer Science

University of Waterloo

$$0.055 \times x + (0.05 \times 0.15 + 0.35 \times 0.12)(1-x) = 0.12 \checkmark$$

$$0.055x + 0.1385 - 0.1385x = 0.12$$



## Technical Requirements

The following is a list of technical constraints and guidelines:

- The assignment must be programmed using only Java 11 and Java FX 11 libraries.
- The starting application window should be no larger than 1600x1200. The window should be resizable, as described above. You should set a reasonable minimum and maximum size for the window.
- The layout should be dynamic and designed appropriately to support scaling the application during resizing.
- Appropriate widgets should be selected and used for the interface. Options should appropriately enabled/disabled in the UI (i.e. if an option is unavailable, it should be disabled).
- You must use the model-view-controller design pattern in this assignment. You are expected to have at least one model and two views, with the appropriate class structure.
- You may use third-party graphics/images, provided that (a) they are licensed to allow you to use them in this way (e.g. Creative Commons), and (b) you comment their origin and license in your README file.
- You may use any sample code provided in-class (including on the class repository). If you do this, you **must** provide a comment in the code indicating the source of the code. If you use and fail to disclose this information, it could be considered an academic violation. You may not use any other third-party code unless you obtain instructor consent ahead of time.

## Submission

Your directory structure for your assignment submission should be in subdirectories under your a3/ directory in your Git repository. It should use a standard file layout for an IntelliJ project. For example, your submission should resemble this (likely with slightly different class, image and sound file names). Note that you should NOT include the contents of the out/ folder (since they will be replaced when we build your project).

```
a3/
├── a3.iml
├── out/
├── readme.md
└── src/
    └── images/
        ├── selection.jpg
        ├── erase.jpg
        └── circle.jpg
    ├── CanvasView.java
    ├── SketchIt.java
    ├── Model.java
    └── ToolbarView.java
```

Your submission needs to include:

- All source code and resources required to build and execute your program.
- An IntelliJ project that compiles everything using the specified JDK and Java FX versions, and which has a Run target that will execute your program.
- A `readme.md' file with any details required to help the TA grade your assignment.

Your `readme.md' file needs to include, at a minimum:

- Your name,
- Your student number,
- The version of Java that you used,
- Your operating system & version,
- The source for your image and sound assets,
- Which additional features (and how to use them!).

## Assessment

Your submission will be assessed roughly as follows:

**5%**
IntelliJ project compiles and runs program.
**60%**
Functional requirements, described above.