Assignment 4

Task1

a) Reactive programming is a programming paradigm that focuses on managing time-varying values and the propagation of changes. It is like watching a live stream. Instead of waiting for a whole chunk of data to be available, you get updates in real-time and can react to them as they come. It's good for handling constantly changing information.

b) In modern graphical user interfaces (GUIs), elements such as buttons, sliders, or text fields can trigger events when they are interacted with. These events are propagated through the system, potentially causing other components to update in response. For instance, a color picker that changes the background color of a canvas in a drawing application. The color selection flows through the system, impacting the canvas rendering in real-time.

c) An example of a glitch in a reactive program could be in a temperature conversion application. Suppose there is an expression var3 = var1 + var2 where var1 represents the temperature in Celsius and var2 represents a constant conversion factor of 1.8. If the value of var1 changes, the value of var3 should be immediately updated to reflect the new temperature in Fahrenheit.

However, in a naive reactive implementation, if the value of var1 changes before the expression var1 * 1.8 is recomputed, the value of var3 will momentarily be incorrect. This incorrect intermediate state is called a “glitch”.

The consequences of such a glitch could be that the application displays the wrong temperature value, leading to incorrect information being presented to the user. In a safety-critical system, such glitches could have more severe consequences, for example, if the temperature reading is used to control a critical industrial process.

Task 2

a) In JavaFX, setX() and setY() methods are used to set the position of a Node (like a Rectangle) in the X and Y direction respectively. These methods set the layout position of the node, which is the position of the node in its parent's layout.

On the other hand, setTranslateX() and setTranslateY() methods are used to move a Node from its current position in the X and Y direction respectively. These methods do not change the layout position of the node, but rather apply a translation transformation to it.

Event handlers in the createHandles() method are likely used to handle user interactions with the handles, such as dragging them to resize the rectangle.

If we replace setX()/setY() with setTranslateX()/setTranslateY(), the behavior is going to change. If we want to extend the shape of the rectangle or rotate it, the dimensions are going to be disproportionate.

e) Tools:

- **Selection tool (Arrow):** The most basic tool, used to click and drag to select rectangles.
- **Resize handles:** Small squares appear at the corners and sometimes on the sides of a selected rectangle. Clicking and dragging these handles resizes the rectangle.

- **Rotation handle:** A circular handle appears above a selected rectangle. Clicking and dragging this handle rotates the rectangle.

Behavior:

- Clicking a rectangle with the selection tool makes it the active object.
- Hovering over the edges or corners highlights the resize handles.
- Holding Shift while resizing maintains aspect ratio (keeps proportions).
- Holding Ctrl (or Cmd on Mac) allows resizing from the center.

Additional Information:

- **Dimensions:** Some programs display the width and height of the rectangle next to the selection box.
- **Position:** Coordinates (X,Y) might be displayed in the status bar, indicating the rectangle's position on the slide.

Aligning Objects:

- **Guides:** Drag objects to the edges of the slide or other objects to create temporary guides for alignment.
- **Align tools:** Many programs offer dedicated "Align" or "Distribute" menus with options to align objects horizontally, vertically, or center them within a selection.
- **Snap to grid:** Enabling a grid option allows objects to snap to specific points, ensuring precise alignment.