CISC 3120

C14: JavaFX: Overview and Programming User Interface

Hui Chen

Department of Computer & Information Science

CUNY Brooklyn College

Outline

- Recap and issues
- Architecture and features of JavaFX
- JavaFX build-in UI elements
- Assignments

Recap and Issues

- Projects
  - Project 1 & 2
  - Upcoming project: project 3
    - GUI application
- Midterm
  - Grades to be posted by Thursday, October 19
- Group-discussion on the "HelloWorldFX" application

Lessons from Project 1

- Java naming convention
  - How should you name constants and variables?
- Improving reading & coding difficulty
  - Using literals
    - Named constants are better
  - Divide-and- conquer: writing methods

Naming Constants and Variables

- Which one of the two should you write according to the Java coding convention?

```
final static int GAME_BOARD_WIDTH = 80;
```

```
final static int gameBoardWidth = 80;
```

- Which one of the two should you write?

```
int GAME_BOARD_WIDTH = 80;
```

```
int gameBoardWidth = 80;
```

Naming Constants

- Which one of the two should you write according to the Java coding convention?

```
final static int GAME_BOARD_WIDTH = 80;
```

```
final static int gameBoardWidth = 80;
```

- Which one of the two should you write?

```
int GAME_BOARD_WIDTH = 80;
```

```
int gameBoardWidth = 80;
```

Using Literals

- Which one is easier to understand when you read?

```
if (numGuesses < 10) {
    …
}
```

```
final static int MAX_ALLOWED_GUESSES = 10;

…

if (numGuesses < MAX_ALLOWED_GUESSES) {
…
}
```

Divide-and-Conquer: Writing Methods

- Which one is easier to read and code?

```
public class TargetGameLauncher
{
   public static void main(String[] args)   {
      CommandLineParser parser = new DefaultParser();
      int gameWidth = 80, gameHeight = 25, gameLevel = 0;
      Options options=new Options();
      options.addOption("w","width", true,"width parameter");
      options.addOption("h","height", true,"height parameter");
      options.addOption("l","level", true,"level parameter");

      try {
          CommandLine line = parser.parse(options, args);
          if(!(line.getOptionValue("w")==null))
              w = line.getOptionValue("w");
              gameWidth = Integer.parseInt(w);
              ……
      } catch (ParseException exp) {
      }
     TargetGame game =
        new TargetGame(gameWidth,gameHeight,gameLevel);
      game.play();
   }
}
```

```
public class TargetGameLauncher
{

   public static void main(String[] args)   {
      parseGameOptions(args);
      TargetGame game =
        new TargetGame(gameWidth,gameHeight,gameLevel);
      game.play();
   }


   private static void parseGameOptions(String[] args) { …


   ……


  }

  private static int gameWidth;
  private static int gameHeight;
  private static int gameLevel;
}
```

Questions?

- Lessons from Project 1
  - Java naming convention
  - Using literals
  - Divide-and-conquer: writing methods

"Programs must be written for people to read, and only incidentally for machines to execute."

-- H. Abelson and G. Sussman (in "The Structure and Interpretation of Computer Programs")

Discussed the "HelloWordFX" App in Groups

- What did we learn from it?
- What were your questions?

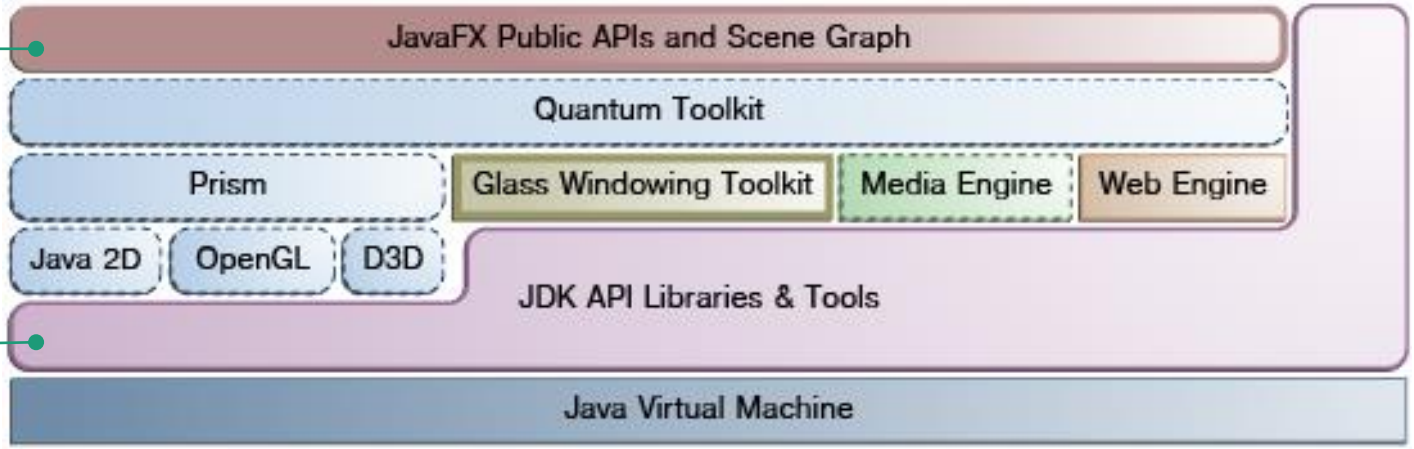
Writing JavaFX GUI Application

- Overview of JavaFX
- JavaFX application life cycle
- JavaFX application structure
- Write JavaFX application from scratch
  - Learn new ones from existing knowledge and skills
  - Learn to use Java API documentation
  - Learn a few concepts in GUI and computer graphics
- Learn to reuse JavaFX build-in UI components

JavaFX Overview

- A Java API
  - Consisting of classes & interfaces in a few Java packages
  - Dealing with graphics and media
  - for creating rich client applications
    - whose look & feel are customizable via Cascading Style Sheets (CSS)
  - cross platforms
    - Desktop, mobile, embedded, and the Web

JavaFX Architecture



- Develop apps with JavaFX public APIs and JDK API libraries and tools
- Powered by JVM, Graphic System, and Windowing toolkit

Features of JavaFX

- Graphics: supports *DirectX* and *OpenGL*, software render fallback (via Prism, OpenGL, Direct3D)
  - 3D graphics: supports light sources, material, camera, 3-D shapes and transformations; Common visual effects
- Interfacing with native operating systems to provide windows management, timers, and event queues (Glass windows toolkit)
- Multimedia: support playbacks of web multimedia content based on the GStreamer multimedia framework (Media engine)
- Web: provides a Web viewer and full browsing functionality based on WebKit (Web engine)
- Multi-threaded: concurrent application, Prism render, and media threads (Quantum toolkit)
- Text: supports bi-directional text and complex text scripts
- I/O devices: supports multi-touch and Hi-DPI
- Build-in UI controls, layouts, themes, and CSS styling
- Swing interoperability
- JavaFX API: application lifecycle; stage; scene; transition & animation; canvas; print; event; css; fxml; collections; utils; Java beans; javascript

JavaFX API

- Full package list at http://docs.oracle.com/javase/8/javafx/api/toc.htm
  - javafx.application: provides the application life-cycle classes.
  - javafx.stage: provides the top-level container classes for JavaFX content.
  - javafx.scene: provides the core set of base classes for the JavaFX Scene Graph API.
  - javafx.scene.control: prebuilt UI control classes
  - javafx.scene.text: provides the set of classes for fonts and renderable text.
  - javafx.scene.layout: prebuilt container classes defining user interface layout.
  - ……

JavaFX Stage and Scene

"All the world's a stage, and all the men and women merely players."

-- As You Like It, Act II, Scene VII, William Shakespeare

JavaFX Stage

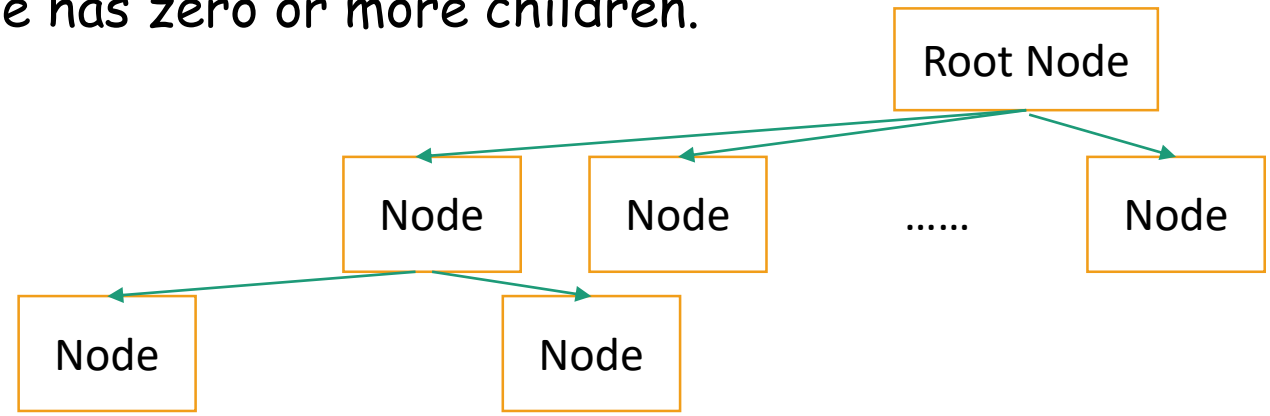
- A JavaFX runtime constructs a primary stage
  - java.stage.Stage: the top level JavaFX container
  - Visually represented by a "window" in windows-based operating systems (such as, Windows, Mac OS X)
  - An applications can construct additional stage
  - The application needs to construct and set scenes for a stage
    - JavaFX scene graph

JavaFX Scene Graph

- Represent visual elements of user interface.
  - Elements can be displayed inside a window
- Handles input.
- Can be rendered.

Scene Graph

- Elements organized as a hierarchical structure, like a tree.
  - An element in a scene graph is called a node.
    - Each non-root node has a single parent.
    - Each node has zero or more children.



Node in Scene Graph Tree

- Example nodes
  - a layout container, a group, a shape, a button …
- Each node has an ID, style class, bounding volume, and other attributes
  - Effects, such as blurs and shadows
  - Opacity
  - Transforms
  - Event handlers (such as mouse, key and input method)
  - An application-specific state
- javafx.scene.Node: abstract class

Working with Scene Graph

- Packaged in javafx.scene
  - Nodes (elements)
    - javafx.scene.Node: abstract class
    - UI controls, text, charts, containers, shapes (2-D and 3-D), images, media, embedded web browser, and groups
  - State
    - Transforms (positioning and orientation of nodes), visual effects, and other visual state of the content
  - Effects
    - Simple objects that change the appearance of scene graph nodes, such as blurs, shadows, and color adjustment

JavaFX Application

- Entry point: the Application class
  - javafx.application.Application
    - abstract void start(Stage primaryStage)

JavaFX Application Life-Cycle

- JavaFX runtime does the following, in order,
  - Constructs an instance of the specified Application class (via the launch(String[] args) method)
  - Calls the *init()* method that can be overridden
  - Calls the *start(javafx.stage.Stage)* method that must be overridden in subclass)
  - Waits for the application to finish, which happens when either of the following occur:
    - the application calls *Platform.exit()*
    - the last window has been closed and the implicitExit attribute on Platform is true
  - Calls the stop() method (can be overridden)

JavaFX Application: Remarks

- The *start(javafx.stage.Stage)* is an abstract method, and must be overridden in the subclass
- The init() and stop() method have concrete implementations, but do nothing, and can be overridden.
- Explicitly terminating JavaFX application
  - calling Platform.exit() is the preferred method
  - Calling System.exit(int) is acceptable, but the *stop()* method will ***not*** run.
- JavaFX should not and cannot be used after System.ext(int) is called or the stop() is returned.

Questions?

- JavaFX architecture and features (from 10,000 feet high)
- JavaFX stage and scene
- JavaFX application cycle

Write the First JavaFX Application from Scratch: Demo

"Everything should be built top-down, except the first time."

-- Alan Perlis

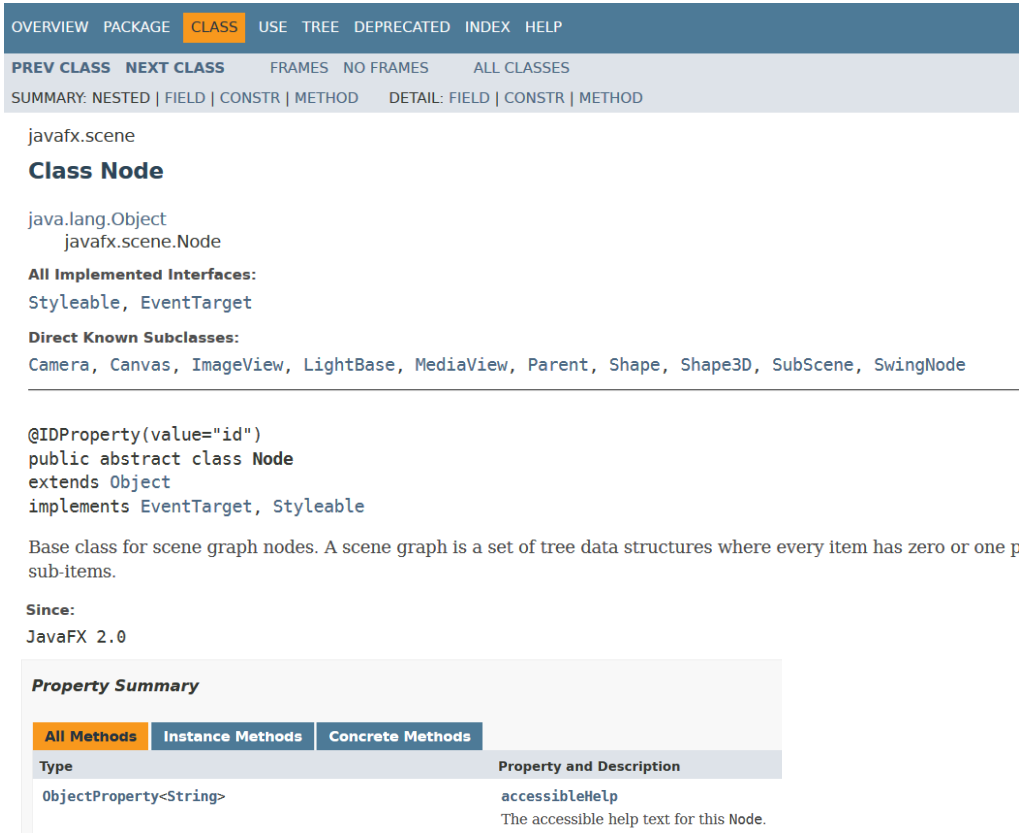
Write the First JavaFX Application from Scratch: Demo

- Create a concrete subclass extending the JavaFX Application class (javafx.application.Application)
- (Curtains down) Construct a scene graph containing a tree of nodes
  - The simplest tree contains a single root node (select a concrete subclass of nodes)
    - http://docs.oracle.com/javase/8/javafx/api/javafx/scene/Node.html
- Set scene for the stage
- (Curtains up) Show the scene

Java API Documentation

- Class documentation
  - Package hierarchy
  - Class name
  - Implemented interfaces
  - Known subclasses
  - Class declaration line
    - Abstract or concrete
    - Super class
  - Description
  - Compatibility
- Properties
  - Public instance variables
- Fields
  - Public class variables and constants
- Constructors
- Methods
  - Method summary
  - Methods inherited
- Property detail

Java API Documentation



OVERVIEW PACKAGE CLASS USE TREE DEPRECATED INDEX HELP

PREV CLASS NEXT CLASS FRAMES NO FRAMES ALL CLASSES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

javafx.scene

Class Node

java.lang.Object
    javafx.scene.Node

All Implemented Interfaces:

Styleable, EventTarget

Direct Known Subclasses:

Camera, Canvas, ImageView, LightBase, MediaView, Parent, Shape, Shape3D, SubScene, SwingNode

---

```
@IDProperty(value="id")
public abstract class Node
extends Object
implements EventTarget, Styleable
```

Base class for scene graph nodes. A scene graph is a set of tree data structures where every item has zero or one p
sub-items.

Since:

JavaFX 2.0

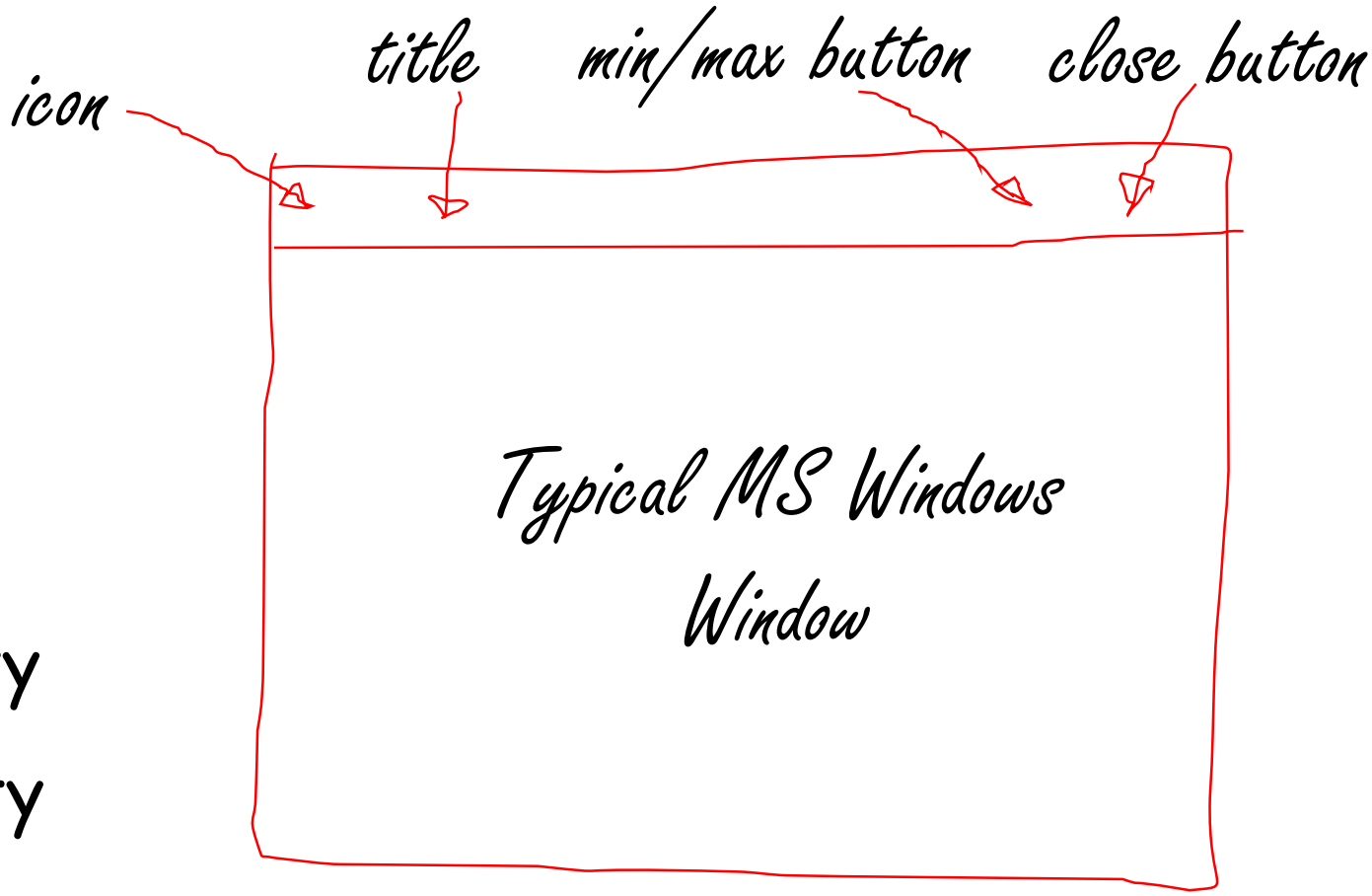
Property Summary

All Methods | Instance Methods | Concrete Methods

Type	Property and Description
ObjectProperty<String>	accessibleHelp The accessible help text for this Node.

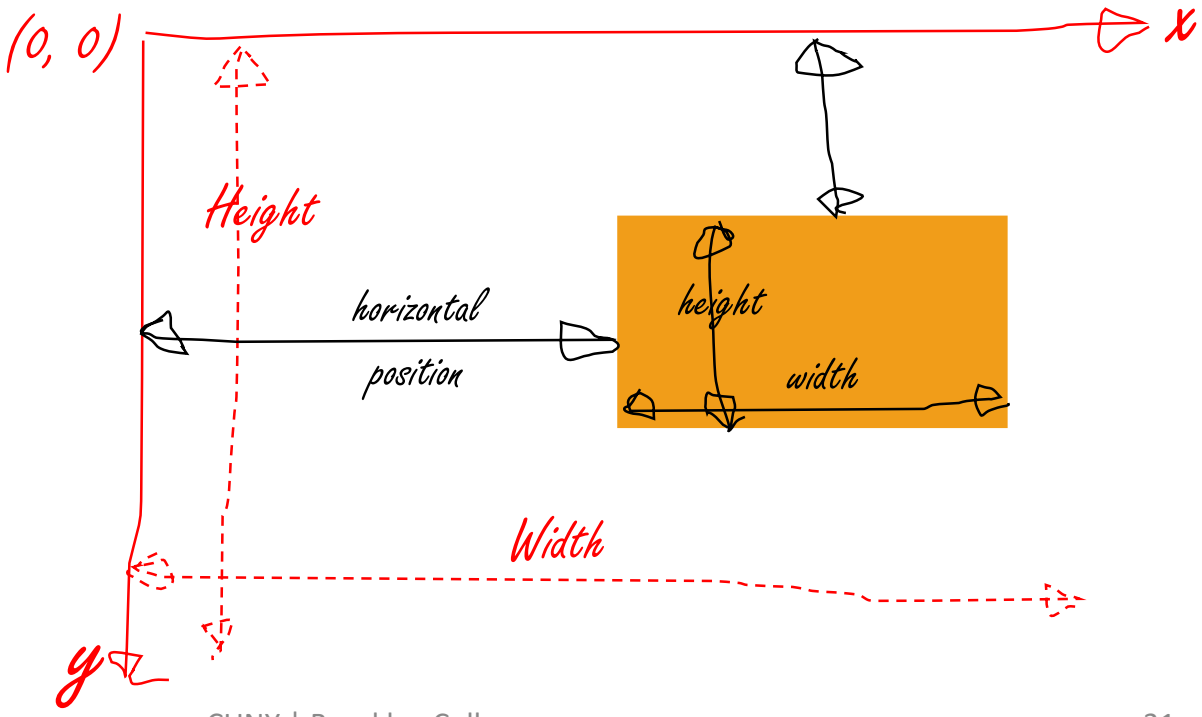
GUI Windows

- Size
- Shape
- Title
- Icon
- Modality
- Visibility



Scene Node Coordinate System

- A traditional computer graphics "local" coordinate system (javafx.scene.node)



Static Factory Method

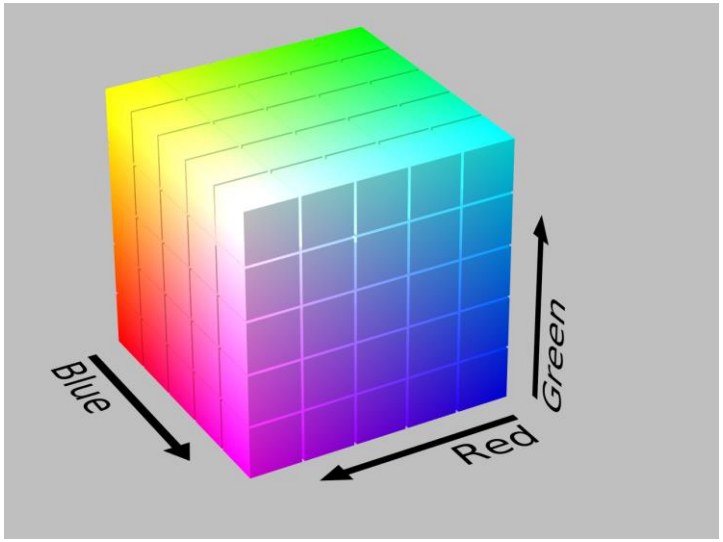
- A static method that returns an instance of the class
  - Examples:
    - static Color hsb(double hue, double saturation, double brightness, double opacity)
    - static Color rgb(int red, int green, int blue, double opacity)
- In your application design: advantage and disadvantage?

Color Space

- Color is a human perception
- (Mathematical) models for color are developed
  - Including a model for human perceptual color space
  - Examples
    - Machine first
      - Additive: Red-Green-Blue (RGB)
      - Subtractive: Cyan-Magenta-Yellow-Black/Key (CMYK)
    - Human first
      - Hue-Saturation-Brightness (HSB)
    - Processing first
      - LAB (Luminance and a & b color channels)

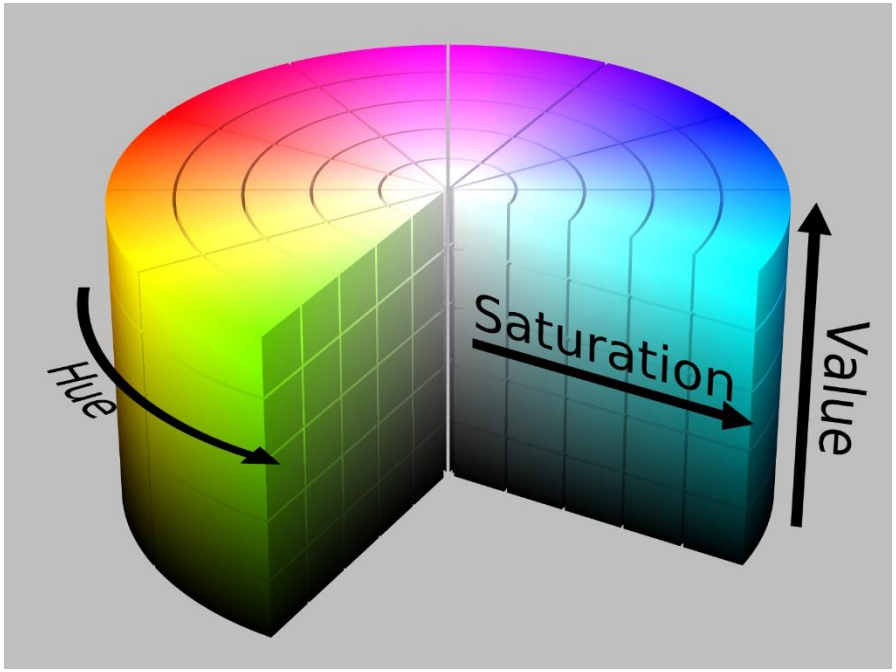
Standard Red-Green-Blue (sRGB)

- Red, Green, Blue
  - 0. – 1.
- Alpha (transparency or opacity)
  - 0.0 – 1.0 or 0 – 255; 1. or 255
  - 0. or 0: completely opaque
  - 1. or 1: completely transparent



Hue-Saturation-Brightness (HSB)

- Hue:
  - 0. – 360.
- Saturation:
  - 0. – 1.
- Brightness (or Value):
  - 0. – 1.
- Alpha (transparency or opacity)
  - 0.0 – 1.0 or 0 – 255; 1. or 255
  - 0. or 0: completely opaque
  - 1. or 1: completely transparent



Blocking and Non-Blocking

- The show() method of a Stage object does not block the caller and returns "immediately".
- The showAndWait() method of a Stage object shows the stage and waits for it to be hidden (closed) before returning to the caller.
  - Cannot be called on the primary stage

Events

- Representing arrivals of inputs
- Mouse events
  - Mouse pressed, mouse released, mouse clicked (pressed & released), mouse moved, mouse entered, mouse exited, mouse dragged
- Keyboard event
  - Key pressed, key released, key typed (pressed & released)
- Gesture event, touch event, …

Events

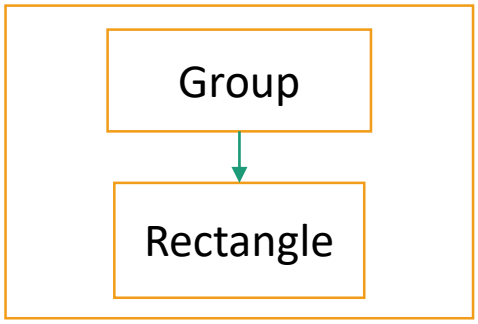
- Event source
  - where (an object) an event is being handled
    - The object needs to have an event handler
- Event target
  - Specifies the path an event travels (from one object to another)
- Event type
  - Additional classification to events of the same Event class.

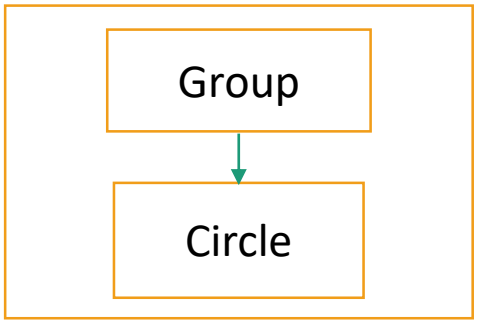
Handling Events

- Register an event handler for an object
  - Implements the EventHandler<T extends Event> interface and create an object for the event handler
  - Set the event handler of the object as the event handler object
  - An object often has a number of convenient setter methods
    - setOnMouseClicked, setOnMouseEntered, seOnMouseExited …
    - More generic method: addEventHandler

Write the First JavaFX Application from Scratch: Demo

- Can we have multiple scenes?
- How do we improve readability?
  - Use named constants
- Can we add more children to a scene graph?
- Can we have multiple stages (windows)?





Questions?

- Wrote the first JavaFX application from scratch (how?)
  - How to read Java API documentation
  - Windows concept: components, styles, and modality
  - Graphics concept: colors, local coordinate system, strokes & fill
    - Example: in the javafx.scence.Node class
      - setStroke(Paint value)
      - setFill(Paint value)
  - Design pattern: static factory method
    - Example: in the javafx.scene.paint.Color class
      - static Color rgb(int red, int green, int blue)
      - static Color hsv(double hue, double saturation, double brightness)

Write Larger JavaFX Applications

- UI controls: prebuilt user interface controls
- Use texts
- Layout containers: prebuilt layouts for UI controls and more
- Handle user interactions and other events
- Use graphics, transformation, and effects, charts, multimedia, and web viewers
- Style user interface using CSS and themes
- Design user interface with provided tools and libraries
  - FXML
  - Model-View-Controller (MVC) design pattern
- Concurrency and other features

User Interface Components

- UI controls
- Layouts
- Text
- Charts
- HTML content & embedded web browser
- Shapes (2-D and 3-D)
- Images
- Groups

Use Build-in UI Controls and Layouts

- UI controls: prebuilt user interface controls
- Use texts
- Layout containers: prebuilt layouts for UI controls and more
- Use 2D graphics
- Handle user interactions with simple event handlers

UI Controls

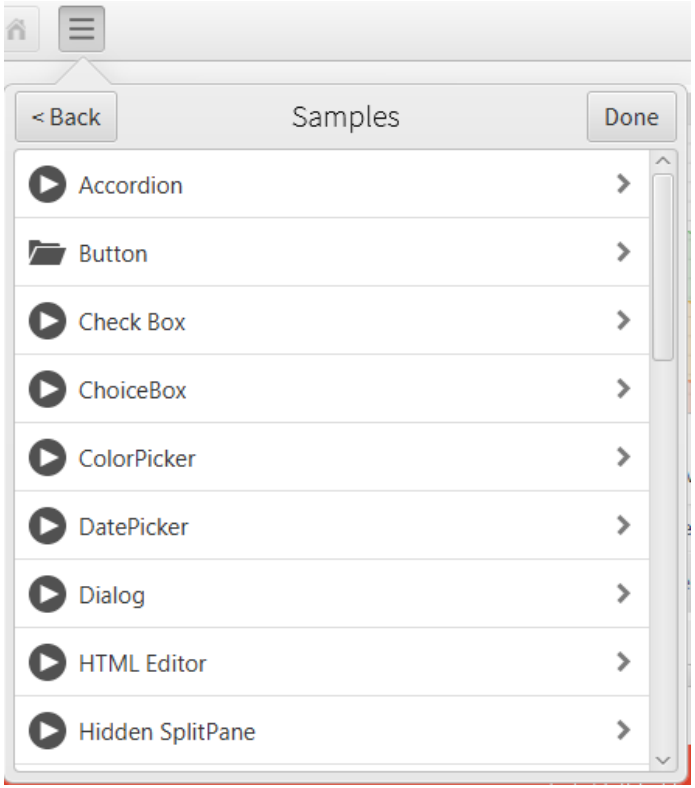
- Packaged in javafx.scene.control

- Label
- Button
- Radio Button
- Toggle Button
- Checkbox
- Choice Box
- Text Field
- Password Field
- Scroll Bar
- Scroll Pane
- List View
- Table View
- Tree View
- Tree Table View
- ComboBox
- Separator
- Slider
- Progress Bar
- Progress Indicator
- Hyperlink
- Tooltip
- HTML Editor
- Titled Pane
- Accordion
- Menu
- Color Picker
- Date Picker
- Pagination Control
- File Chooser

A Gallery of Selected UI Controls

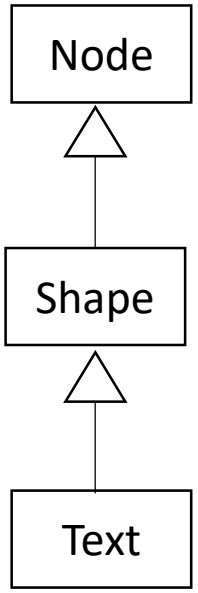
Explore UI Controls

- Using the JavaFX Ensemble 8 sample application
  - Ensemble 8 is in the "sampleprograms" repository
    - Open it as a Maven project
  - Run the ensemble.EnsembleApp class



Text

- Packaged in javafx.scene.text.Text



- Text class inherits from the Shape class, and the Shape class inherits from the Node class
  - You can apply effects, animation, and transformations to text nodes in the same way as to any other Nodes.
  - you can set a stroke or apply a fill setting to text nodes in the same way as to any other Shapes.

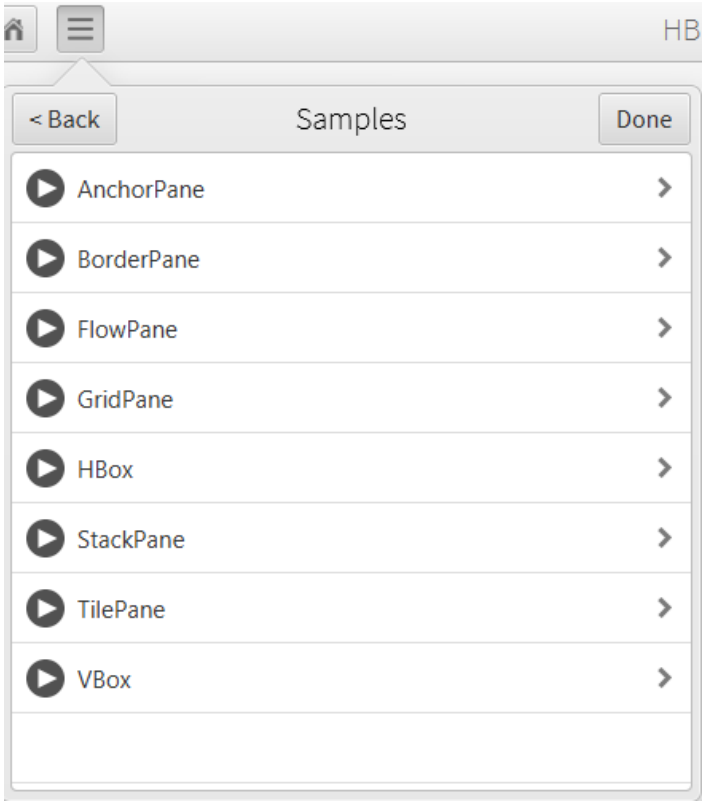
Layout Containers (Panes)

- Packaged in javafx.scene.layout
- Arrangements of the UI controls within a scene graph
- Provide the following common layout models

  - BorderPane
  - HBox
  - VBox
  - StackPane
  - GridPane
  - FlowPane
  - TilePane
  - AnchorPane

Explore Layouts

- Using the JavaFX Ensemble 8 sample application
  - Run the ensemble.EnsembleApp class



2-D Graphics

- Draw images on Canvas
  - Canvas
    - javafx.scene.canvas.Canvas
- Using a set of graphics commands provided by a GraphicsContext.
  - GraphicsContext
    - javafx.scene.canvas.GraphicsContext

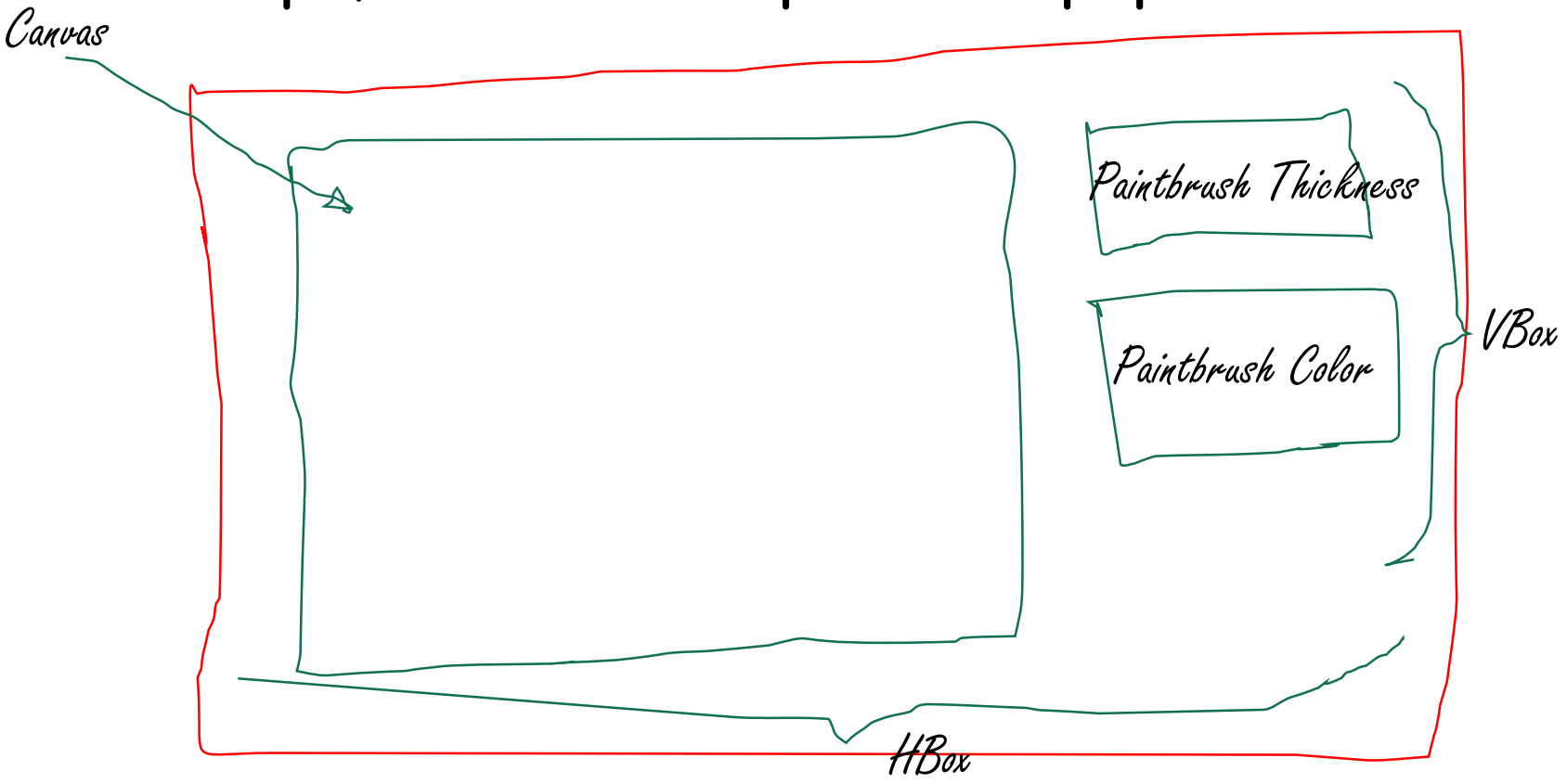
```
Canvas canvas = new Canvas(WIDTH, HEIGHT);
GraphicsContext gc = canvas.getGraphicsContext2D();
```

Use Build-in UI Controls and Layouts: Demo

- Write a JavaFX application with prebuilt UI controls and layouts

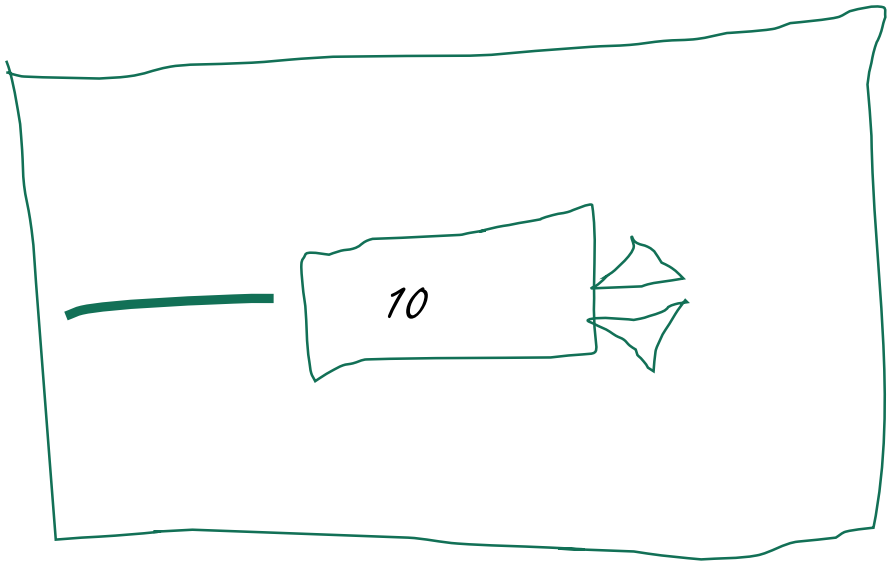
UI Design: Main Scene

- Perhaps, sketch on a piece of paper



UI Design: Brush Thickness

- Perhaps, sketch on a piece of paper



Demonstrate JavaFX in Sample Applications

- The applications are in the "sampleprograms" repository
  - JavaFX Ensemble 8
  - Modena
  - MandelbrotSet
  - 3D Viewer
- In addition to build-in UI controls and layouts, you should explore the following features …

Effects, Animation, and Media

- Visual effects
- 2D and 3D transformations
- Transitions and animation
- Incorporate media

Visual Effects

- Packaged in javafx.scene.effect
  - Drop shadow
  - Reflecting
  - Lighting

2-D and 3-D Transformations

- Packaged in javafx.scene.transform
  - Translate
  - Scale
  - Shear
  - Rotate
  - Affine

Assignments

- Via CUNY Blackboard
  - Practice assignment
  - Project 3

Questions?

- JavaFX build-in components
  - UI controls
  - Text
  - Layouts
  - UI design
- Sample applications for exploring JavaFX features
- Assignments