# CS 351
# Design of Large Programs
# JavaFX

August 25, 2021

## Program is an Application

- Your program should extend javafx.application.Application
- In main, call the launch method
- Override the start method to set up your program.

## All the World's a Stage

- Top level display container is a *stage*
- User interface items are contained in a *scene*
- Only one scene at a time can be displayed on a stage.
- The Application start method takes a stage.
  - Set the title
  - Set the scene
  - Show the stage

## SceneGraph and Nodes

- The Scene contains a *scene graph*, containing all the components of the user interface.
- User interface objects are *nodes*, derived from `Node` class.
- Nodes are groups, layouts, controls, shapes, etc.

## Layouts are Nodes

- In Swing, you would create a JPanel, configure its LayoutManager, and add components to the panel.
- In JavaFX, a layout is a subclass of `Node` and contains a collection of other nodes.
- Many Layouts will feel familiar: BorderPane, FlowPane, GridPane, etc.

## Event Handling

- Like Swing, JavaFX uses event handlers to respond to user input.
- The `EventHandler` interface expects a type parameter for which type of event it handles.
- Add the handler object to a node using `addEventHandler` method.

# Hello World

```java
public class Hello extends Application {
  public static void main(String[] args) { launch(args); }

  @Override
  public void start(Stage primaryStage) {
    primaryStage.setTitle("Hello World!");
    Button btn = new Button();
    btn.setText("Say 'Hello World'");
    btn.setOnAction(new EventHandler<ActionEvent>() {

      @Override
      public void handle(ActionEvent event) {
        System.out.println("Hello World!");
      }
    });

    StackPane root = new StackPane();
    root.getChildren().add(btn);
    primaryStage.setScene(new Scene(root, 300, 250));
    primaryStage.show();
  }
}
```

## Lambda Syntax

Anonymous inner class in previous example

```
btn.setOnAction(new EventHandler<ActionEvent>() {
  public void handle(ActionEvent event) {
    System.out.println("Hello World!");
  }
});
```

could be replaced with lambda syntax.

```
btn.setOnAction(event -> {
  System.out.println("Hello World!");
});
```

Single statement body could leave out curly braces.

```
btn.setOnAction(event ->
    System.out.println("Hello World!"));
```

## Drawing on a Canvas

```
public class CanvasDemo extends Application {
  public static void main(String[] args) { launch(args); }

    @Override
    public void start(Stage stage) {
        stage.setTitle("Canvas Drawing Demo");

        Canvas canvas = new Canvas(300,200);

        GraphicsContext gc = canvas.getGraphicsContext2D();
        gc.setFill (Color.GOLD);
        gc.fillPolygon(
                new double[] {150, 300, 450},
                new double[] {150, 0, 150},
                3
        );
        gc.fillPolygon(
                new double[] {0, 150, 300},
                new double[] {300, 150, 300},
                3
        );
        gc.fillPolygon(
                new double[] {300, 450, 600},
                new double[] {300, 150, 300},
                3
        );

        stage.setScene(new Scene(new StackPane(canvas)));
        stage.show();
    }
}
```

## Mouse Events

```
public class MouseEventDemo extends Application {
  public static void main(String[] args) { launch(args); }

  @Override
  public void start(Stage stage) {
    stage.setTitle("Mouse Object Demo");
    Canvas canvas = new Canvas(700,500);

    canvas.addEventHandler(MouseEvent.MOUSE_PRESSED, event -> {
      System.out.println("pressed "
              + event.getX() + " " + event.getY());
    });

    canvas.setOnMouseMoved(event -> {
      System.out.println("moved "
              + event.getX() + " " + event.getY());
    });

    stage.setScene(new Scene(new StackPane(canvas)));
    stage.show();
  }
}
```

## AnimationTimer

- Can use as main program loop in JavaFX program.
- Implement `handle` method
- Argument is current time in nanoseconds
- Time between calls to `handle` may vary depending on other program events.

## Scanner and AnimationTimer example

```java
public class ScannerInput extends Application {
  public static void main(String[] args) { launch(args); }

  @Override
  public void start(Stage primaryStage) {
    primaryStage.setTitle("Scanner Input Demo");

    Label text = new Label("Initial String");
    primaryStage.setScene(new Scene(text, 200, 200));
    primaryStage.show();

    Scanner sc = new Scanner(System.in);
    AnimationTimer a = new AnimationTimer() {
      @Override
      public void handle(long now) {
        String line = sc.nextLine();
        text.setText(line);
      }
    };
    a.start();
  }
}
```

# Elapsed Time with java.time.Duration

```java
public class TimerDisplay extends Application {
    public static void main(String[] args) { launch(args); }

    @Override
    public void start(Stage primaryStage) {
        primaryStage.setTitle("Time Display Demo");
        Label text = new Label();
        primaryStage.setScene(new Scene(text, 200, 100));
        primaryStage.show();

        AnimationTimer a = new AnimationTimer() {
            private long startTime = -1;

            @Override
            public void handle(long now) {
                if(startTime < 0) {
                    startTime = now;
                }

                Duration elapsed = Duration.ofNanos(now - startTime);
                long minutes = elapsed.toMinutes();
                long seconds = elapsed.getSeconds() - minutes*60;
                String str = String.format("elapsed time %2d:%02d",
                                           minutes, seconds);
                text.setText(str);
            }
        };
        a.start();
    }
}
```

## Encapulating Mouse State – 1/2

```
public class MouseStateInfo {
  private double x;
  private double y;

  public EventHandler<MouseEvent> getMouseHandler() {
    return event -> {
      x = event.getX();
      y = event.getY();
    };
  }

  public double getX() { return x; }
  public double getY() { return y; }
}
```

You could imagine a single object that holds information gathered from multiple event handlers, simulating a low level mouse state object.

## Encapulating Mouse State – 2/2

```
public class MouseStateObjectDemo extends Application {
  public static void main(String[] args) { launch(args); }

  @Override
  public void start(Stage stage) {
    stage.setTitle("Mouse Object Demo");
    Canvas canvas = new Canvas(700,500);

    MoveStateInfo info = new MouseStateInfo();
    canvas.setOnMouseMoved(info.getMouseHandler());
    stage.setScene(new Scene(new StackPane(canvas)));
    stage.show();

    // Print out mouse location every second
    AnimationTimer a = new AnimationTimer() {
      private long nextTime = 0;

      @Override
      public void handle(long now) {
        if(now > nextTime) {
          System.out.println("mouse at " + info.getX() + " " + info.getY());
          nextTime = now + Duration.ofSeconds(1).toNanos();
        }
      }
    };
    a.start();
  }
}
```