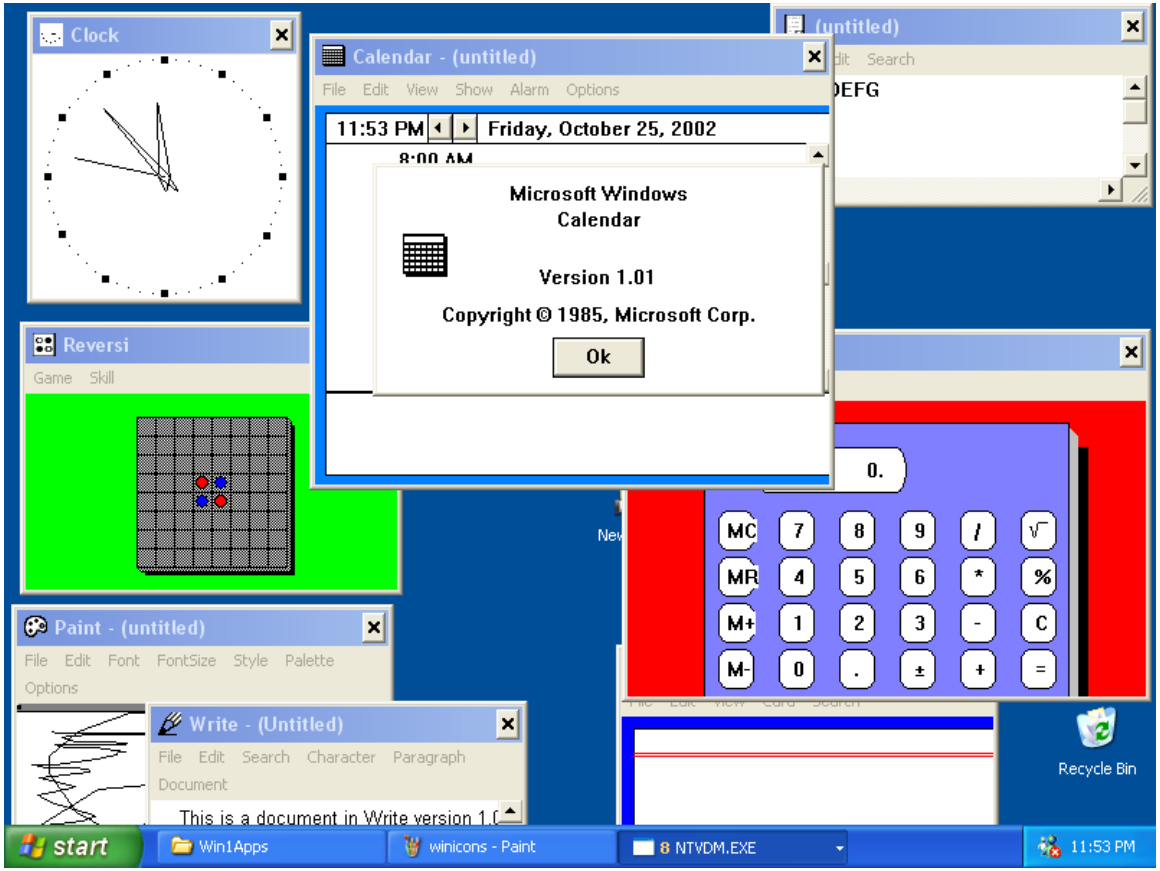
Java Class 16



Managing Fonts

The `Font` class represents a character font, which specify what characters look like when displayed

A font can be applied to a `Text` object or any control that displays text (such as a `Button` or `Label`)

A font is specifies:

- *font family* (Arial, Courier, Helvetica)
- *font size* (in units called points)
- *font weight* (boldness)
- *font posture* (italic or normal)

Managing Fonts

A `Font` object is created using either the `Font` constructor or by calling the static `font` method

The `Font` constructor can only take a font size, or a font family and size

To set the font weight or font posture, use the `font` method, which can specify various combinations of font characteristics

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.Group;
import javafx.scene.paint.Color;
import javafx.scene.text.Font;
import javafx.scene.text.FontPosture;
import javafx.scene.text.FontWeight;
import javafx.scene.text.Text;
import javafx.stage.Stage;

//*******************************************************************
//  FontDemo.java       Author: Lewis/Loftus
//
//  Demonstrates the creation and use of fonts.
//*******************************************************************

public class FontDemo extends Application
{
    //-----------------------------------------------------------------
    //  Displays three Text objects using various font styles.
    //-----------------------------------------------------------------
    public void start(Stage primaryStage)
    {
        Font font1 = new Font("Courier", 36);
        Font font2 = Font.font("Times", FontWeight.BOLD,
                FontPosture.ITALIC, 28);
        Font font3 = Font.font("Arial", FontPosture.ITALIC, 14);
```

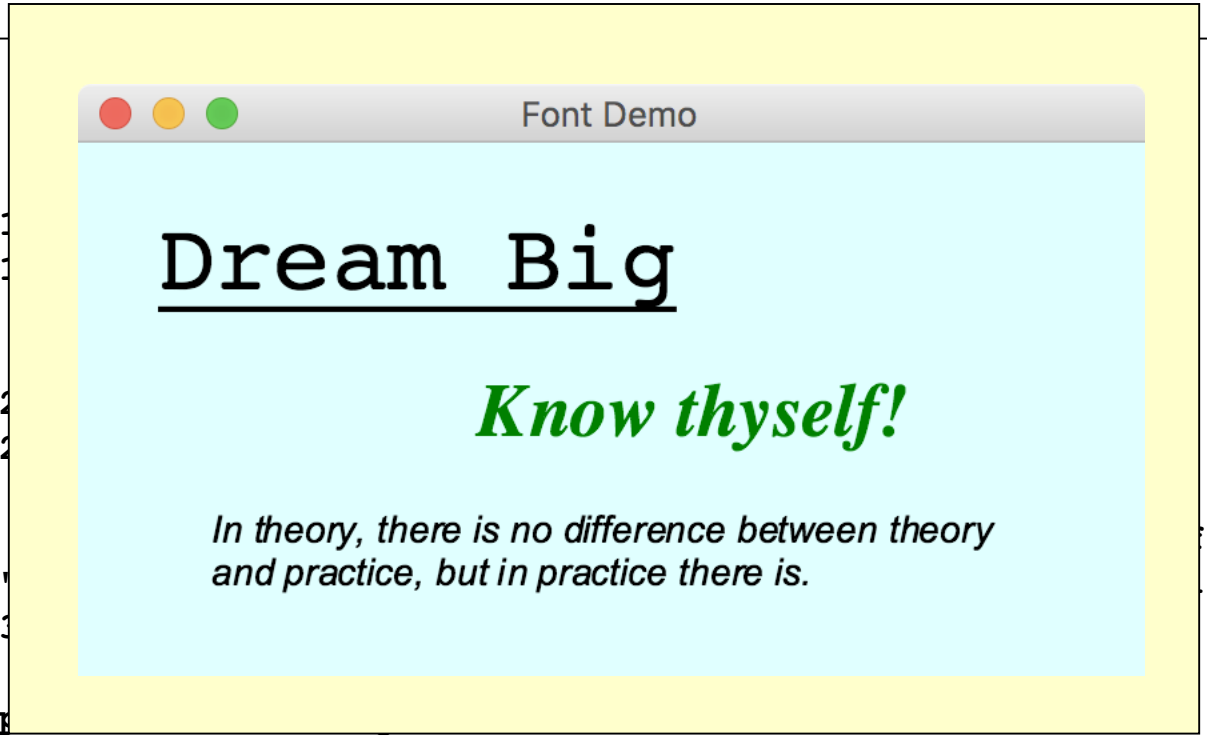
continue

continue

```
        Text text1 = new Text(30, 55, "Dream Big");
        text1.setFont(font1);
        text1.setUnderline(true);

        Text text2 = new Text(150, 110, "Know thyself!");
        text2.setFont(font2);
        text2.setFill(Color.GREEN);

        Text text3 = new Text(50, 150, "In theory, there is no difference " +
            "between theory\nand practice, but in practice there is.");
        text3.setFont(font3);

        Group root = new Group(text1, text2, text3);
        Scene scene = new Scene(root, 400, 200, Color.LIGHTCYAN);

        primaryStage.setTitle("Font Demo");
        primaryStage.setScene(scene);
        primaryStage.show();
    }
}
```

continue

```
        Text
        text1
        text1

        Text
        text2
        text2

        Text                                                    ference " +
                                                                s.");
        text3

        Group
        Scene scene = new Scene(root, 400, 200, Color.LIGHTCYAN);

        primaryStage.setTitle("Font Demo");
        primaryStage.setScene(scene);
        primaryStage.show();
    }
}
```



Managing Fonts

Note that setting the text color is not a function of the font applied

It's set through the `Text` object directly

The same is true for underlined text (or a "strike through" effect)

Check Boxes

A *check box* is a button that can be toggled on or off

It is represented by the JavaFX `CheckBox` class

Checking or unchecking a check box produces an action event

```
import javafx.application.Application;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.stage.Stage;

//*****************************************************************
//  StyleOptions.java       Author: Lewis/Loftus
//
//  Demonstrates the use of check boxes.
//*****************************************************************

public class StyleOptions extends Application
{
    //-----------------------------------------------------------------
    //  Creates and presents the program window.
    //-----------------------------------------------------------------
    public void start(Stage primaryStage)
    {
        StyleOptionsPane pane = new StyleOptionsPane();
        pane.setAlignment(Pos.CENTER);
        pane.setStyle("-fx-background-color: skyblue");

        Scene scene = new Scene(pane, 400, 150);

        primaryStage.setTitle("Style Options");
        primaryStage.setScene(scene);
        primaryStage.show();
    }
}
```

```
import javafx
import javafx
import javafx
import javafx

//***********                                                    *********
//   StyleOpti
//
//   Demonstra
//***********                                                    *********

public class StyleOptions extends Application
{
    //-----------------------------------------------------------------
    //  Creates and presents the program window.
    //-----------------------------------------------------------------
    public vo
    {
        Style
        pane.
        pane.

        Scene

        prima
        prima
        prima
    }
}
```

```
import javafx.event.ActionEvent;
import javafx.geometry.Pos;
import javafx.scene.control.CheckBox;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.scene.text.Text;
import javafx.scene.text.Font;
import javafx.scene.text.FontPosture;
import javafx.scene.text.FontWeight;

//*****************************************************************
//  StyleOptionsPane.java       Author: Lewis/Loftus
//
//  Demonstrates the use of check boxes.
//*****************************************************************

public class StyleOptionsPane extends VBox
{
    private Text phrase;
    private CheckBox boldCheckBox, italicCheckBox;
```

continue

continue

```
    //-----------------------------------------------------------------
    //  Sets up this pane with a Text object and check boxes that
    //  determine the style of the text font.
    //-----------------------------------------------------------------
    public StyleOptionsPane()
    {
        phrase = new Text("Say it with style!");
        phrase.setFont(new Font("Helvetica", 36));

        boldCheckBox = new CheckBox("Bold");
        boldCheckBox.setOnAction(this::processCheckBoxAction);
        italicCheckBox = new CheckBox("Italic");
        italicCheckBox.setOnAction(this::processCheckBoxAction);

        HBox options = new HBox(boldCheckBox, italicCheckBox);
        options.setAlignment(Pos.CENTER);
        options.setSpacing(20);  // between the check boxes

        setSpacing(20);  // between the text and the check boxes
        getChildren().addAll(phrase, options);
    }
```

continue

continue

```
    //-----------------------------------------------------------------
    //  Updates the font style of the displayed text.
    //-----------------------------------------------------------------
    public void processCheckBoxAction(ActionEvent event)
    {
        FontWeight weight = FontWeight.NORMAL;
        FontPosture posture = FontPosture.REGULAR;

        if (boldCheckBox.isSelected())
            weight = FontWeight.BOLD;

        if (italicCheckBox.isSelected())
            posture = FontPposture.ITALIC;

        phrase.setFont(Font.font("Helvetica", weight, posture, 36));
    }
}
```

```
import javafx.a
import javafx.
import javafx.
import javafx.

//************
//  StyleOptio
//
//  Demonstrat
//************

public class StyleOptions extends Application
{
    //-----------------------------------------------------------------
    //  Creates and presents the program window.
    //-----------------------------------------------------------------
    public voi
    {
        StyleO
        pane.s
        pane.s

        Scene

        primar
        primar
        primar
    }
}
```

Radio Buttons

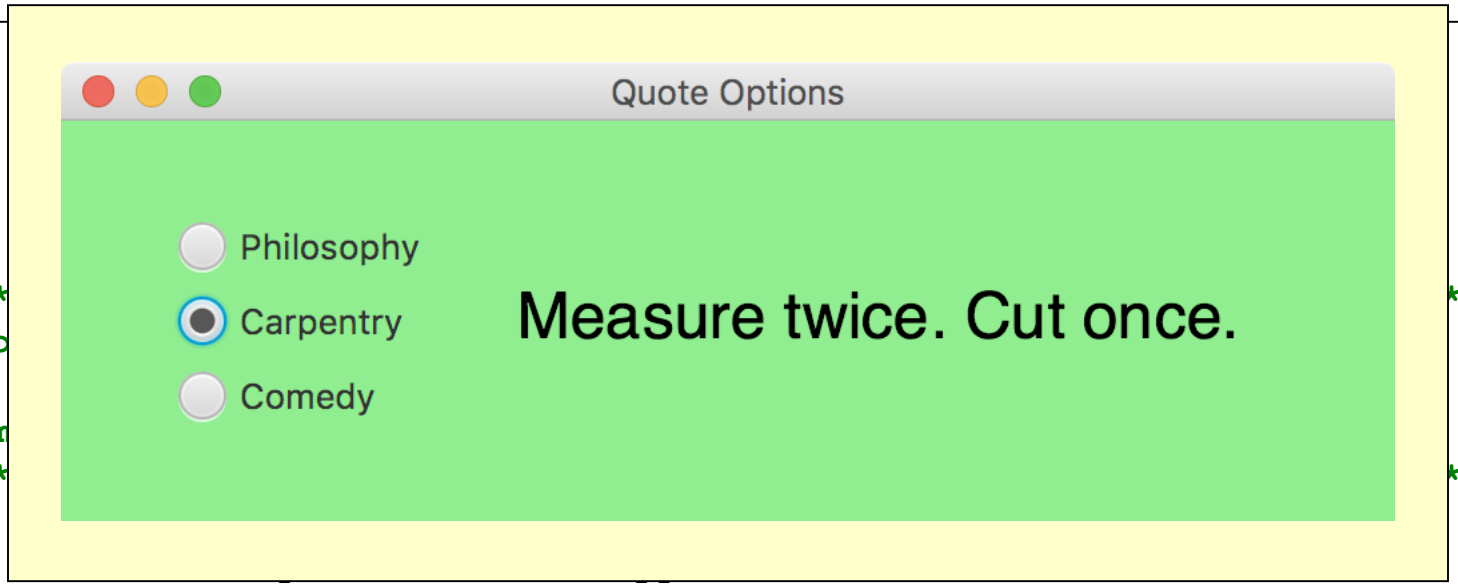
Let's look at a similar example that uses *radio buttons*

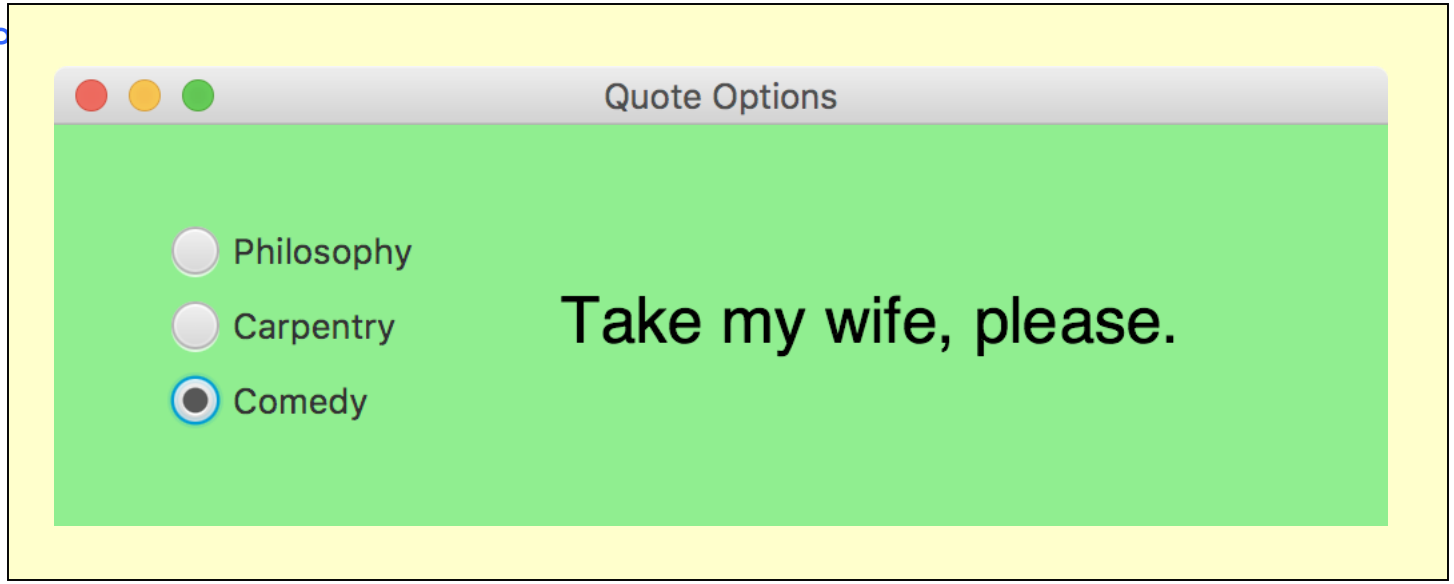
A group of radio buttons represents a set of mutually exclusive options – only one button can be selected at any given time

When a radio button from a group is selected, the button that is currently "on" in the group is automatically toggled off

```java
import javafx.application.Application;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.stage.Stage;

//*****************************************************************
//  QuoteOptions.java       Author: Lewis/Loftus
//
//  Demonstrates the use of radio buttons.
//*****************************************************************

public class QuoteOptions extends Application
{
    //-----------------------------------------------------------------
    //  Creates and presents the program window.
    //-----------------------------------------------------------------
    public void start(Stage primaryStage)
    {
        QuoteOptionsPane pane = new QuoteOptionsPane();
        pane.setAlignment(Pos.CENTER);
        pane.setStyle("-fx-background-color: lightgreen");

        Scene scene = new Scene(pane, 500, 150);

        primaryStage.setTitle("Quote Options");
        primaryStage.setScene(scene);
        primaryStage.show();
    }
}
```

```
import
import
import
import

//*****
//   Quo
//
//   Dem
//*****

public
{
    //-----------------------------------------------------------------
    //  Creates and presents the program window.
    //-----------------------------------------------------------------
    pub
    {

    }
}
```





```
import javafx.event.ActionEvent;
import javafx.geometry.Pos;
import javafx.scene.control.RadioButton;
import javafx.scene.control.ToggleGroup;
import javafx.scene.layout.HBox;
import javafx.scene.layout.StackPane;
import javafx.scene.layout.VBox;
import javafx.scene.text.Text;
import javafx.scene.text.Font;

//*****************************************************************
//  QuoteOptionsPane.java       Author: Lewis/Loftus
//
//  Demonstrates the use of radio buttons.
//*****************************************************************

public class QuoteOptionsPane extends HBox
{
    private Text quote;
    private String philosophyQuote, carpentryQuote, comedyQuote;
    private RadioButton philosophyButton, carpentryButton, comedyButton;
```

continue

continue

```
    //-----------------------------------------------------------------
    //  Sets up this pane with a Text object and radio buttons that
    //  determine which phrase is displayed.
    //-----------------------------------------------------------------
    public QuoteOptionsPane()
    {
        philosophyQuote = "I think, therefore I am.";
        carpentryQuote = "Measure twice. Cut once.";
        comedyQuote = "Take my wife, please.";

        quote = new Text(philosophyQuote);
        quote.setFont(new Font("Helvetica", 24));

        StackPane quotePane = new StackPane(quote);
        quotePane.setPrefSize(300, 100);

        ToggleGroup group = new ToggleGroup();

        philosophyButton = new RadioButton("Philosophy");
        philosophyButton.setSelected(true);
        philosophyButton.setToggleGroup(group);
        philosophyButton.setOnAction(this::processRadioButtonAction);
```

continue

continue

```
        carpentryButton = new RadioButton("Carpentry");
        carpentryButton.setToggleGroup(group);
        carpentryButton.setOnAction(this::processRadioButtonAction);

        comedyButton = new RadioButton("Comedy");
        comedyButton.setToggleGroup(group);
        comedyButton.setOnAction(this::processRadioButtonAction);

        VBox options = new VBox(philosophyButton, carpentryButton,
            comedyButton);
        options.setAlignment(Pos.CENTER_LEFT);
        options.setSpacing(10);

        setSpacing(20);
        getChildren().addAll(options, quotePane);
    }
```

continue

continue

```
    //-----------------------------------------------------------------
    //  Updates the content of the displayed text.
    //-----------------------------------------------------------------
    public void processRadioButtonAction(ActionEvent event)
    {
        if (philosophyButton.isSelected())
            quote.setText(philosophyQuote);
        else if (carpentryButton.isSelected())
            quote.setText(carpentryQuote);
        else
            quote.setText(comedyQuote);
    }
}
```

Radio Buttons

To establish a set of mutually exclusive options, the radio buttons that work together as a group are added to a `ToggleGroup` object

The `setToggleGroup` method is used to specify which toggle group a button belongs to

The `isSelected` method of a radio button returns true if that button is currently "on"

Graphic Transformations

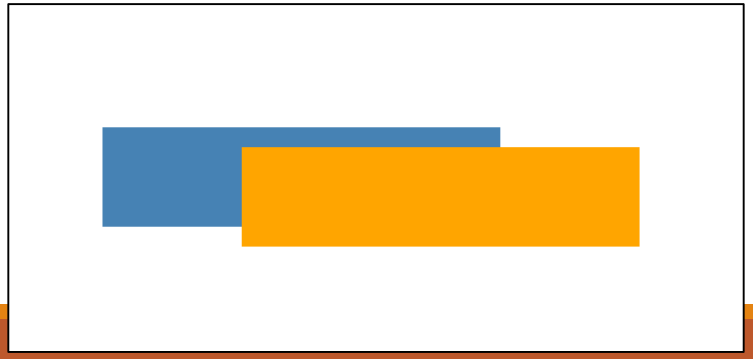
A JavaFX *transformation* changes the way a node is presented visually

- *translation* – shifts the position along the x or y axis
- *scaling* – causes the node to appear larger or smaller
- *rotation* – rotates the node around its center point
- *shearing* – rotates one axis so that the x and y axes are no longer perpendicular

Translation

The following creates two rectangles in the same position, then shifts the second one:

```
Rectangle rec1 = new Rectangle(100, 100, 200, 50);
rec1.setFill(Color.STEELBLUE);

Rectangle rec2 = new Rectangle(100, 100, 200, 50);
rec2.setFill(Color.ORANGE);
rec2.setTranslateX(70);
rec2.setTranslateY(10);
```



Scaling

The following displays two `ImageView` objects, the second scaled to 70%:

```
Image img = new Image("water lily.jpg");
ImageView imgView1 = new ImageView(img);

ImageView imgView2 = new ImageView(img);
imgView2.setX(300);
imgView2.setScaleX(0.7);
imgView2.setScaleY(0.7);
```

Rotation

The parameter to `setRotate` determines how many degrees the node is rotated

If the parameter positive, the node is rotated clockwise

If the parameter is negative, the node is rotated counterclockwise

Rotation

```
Rectangle rec = new Rectangle(50, 100, 200, 50);
rec.setFill(Color.STEELBLUE);
rec.setRotate(40);

Text text = new Text(270, 125, "Tilted Text!");
text.setFont(new Font("Courier", 24));
text.setRotate(-15);
```

Rotation

To rotate a node around a point other than its center point, create a `Rotate` object and add it to the node's list of transformations

The following rotates a node 45 degrees around the point (70, 150):

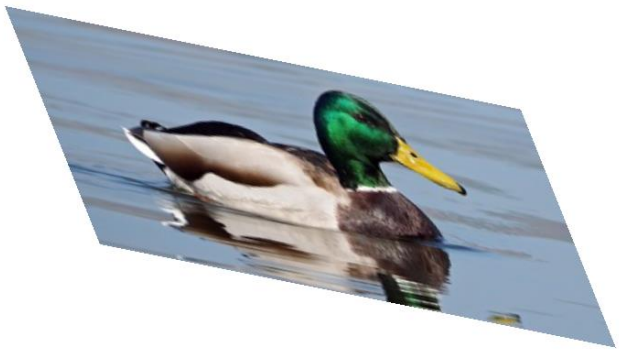
```
node.getTransforms().add(new Rotate(45, 70, 150));
```

Shearing

Shearing is accomplished by creating a `Shear` object and adding it to this list of transformations

The following applies a shear of 40% on the x axis and 20% on the y axis to an `ImageView` object:

```
Image img = new Image("duck.jpg");
ImageView imgView = new ImageView(img);
imgView.getTransforms().add(new Shear(0.4, 0.2));
```



Transformations on Groups

Transformations can be applied to any JavaFX nodes

- shapes, images, controls
- groups and panes

When applied to a group or pane, the transformation is applied to each node it contains

```
import javafx.scene.Group;
import javafx.scene.paint.Color;
import javafx.scene.shape.Rectangle;

//********************************************************************
//  RobotFace.java       Author: Lewis/Loftus
//
//  Presents the face of a robot.
//********************************************************************

public class RobotFace extends Group
{
    //-----------------------------------------------------------------
    //  Sets up the elements that make up the robots face, positioned
    //  in the upper left corner of the coordinate system.
    //-----------------------------------------------------------------
    public RobotFace()
    {
        Rectangle head = new Rectangle(5, 0, 100, 70);
        head.setFill(Color.SILVER);
        head.setArcHeight(10);
        head.setArcWidth(10);

        Rectangle ears = new Rectangle(0, 20, 110, 30);
        ears.setFill(Color.DARKBLUE);
        ears.setArcHeight(10);
        ears.setArcWidth(10);
```

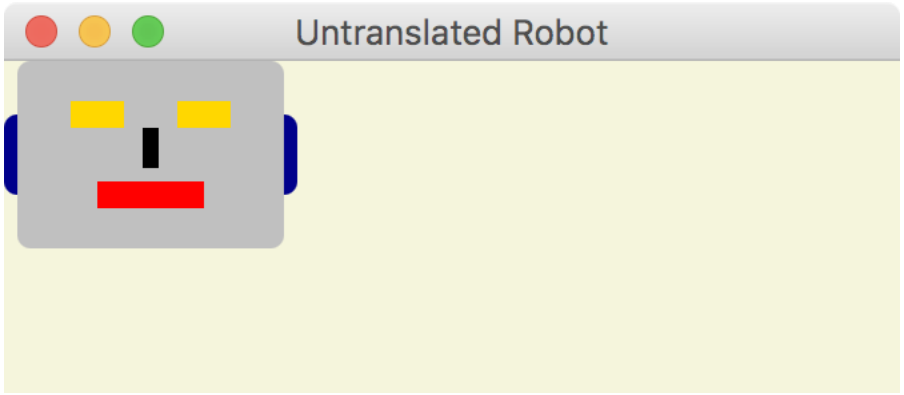
continue

continue

```
        Rectangle eye1 = new Rectangle(25, 15, 20, 10);
        eye1.setFill(Color.GOLD);

        Rectangle eye2 = new Rectangle(65, 15, 20, 10);
        eye2.setFill(Color.GOLD);

        Rectangle nose = new Rectangle(52, 25, 6, 15);
        nose.setFill(Color.BLACK);

        Rectangle mouth = new Rectangle(35, 45, 40, 10);
        mouth.setFill(Color.RED);

        getChildren().addAll(ears, head, eye1, eye2, nose, mouth);
    }
}
```

Transformations on Groups

If presented as defined, the robot face would be displayed in the upper left corner:



```
import javafx.application.Application;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.paint.Color;
import javafx.stage.Stage;

//********************************************************************
//  Robots.java       Author: Lewis/Loftus
//
//  Demonstrates graphical transformations.
//********************************************************************

public class Robots extends Application
{
    //-----------------------------------------------------------------
    //  Displays three robot faces, applying various transformations.
    //-----------------------------------------------------------------
    public void start(Stage primaryStage)
    {
        RobotFace robot1 = new RobotFace();
        robot1.setTranslateX(70);
        robot1.setTranslateY(40);

        RobotFace robot2 = new RobotFace();
        robot2.setTranslateX(300);
        robot2.setTranslateY(40);
        robot2.setRotate(20);
```

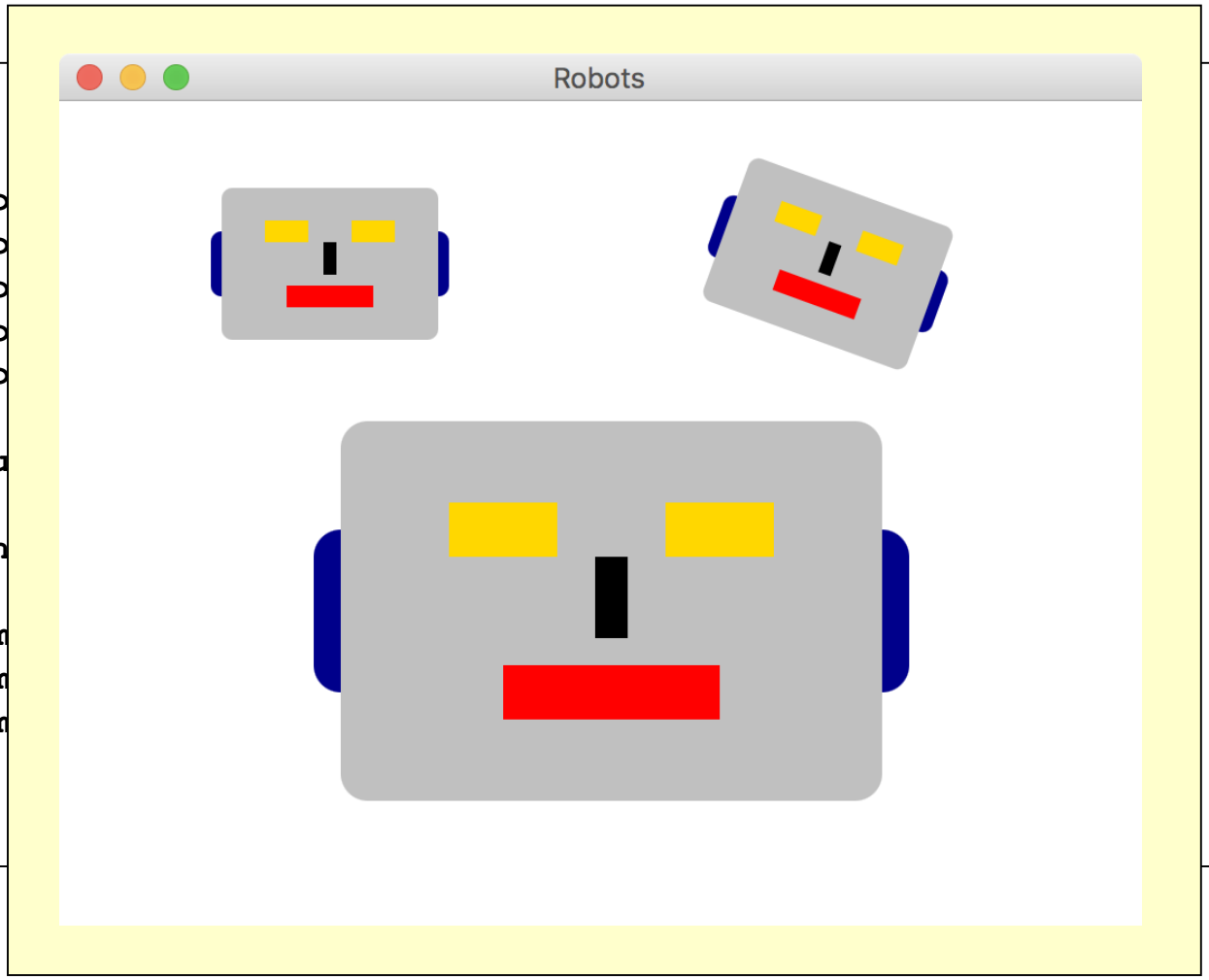
continue

continue

```
        RobotFace robot3 = new RobotFace();
        robot3.setTranslateX(200);
        robot3.setTranslateY(200);
        robot3.setScaleX(2.5);
        robot3.setScaleY(2.5);

        Group root = new Group(robot1, robot2, robot3);

        Scene scene = new Scene(root, 500, 380, Color.WHITE);

        primaryStage.setTitle("Robots");
        primaryStage.setScene(scene);
        primaryStage.show();
    }
}
```

continue

```
            Robo
            robo
            robo
            robo
            robo

            Grou

            Scen

            prim
            prim
            prim
        }
}
```



Note: this is an individual assignment!

Continue with Programming Assignment #2

Copy the 27 fruit pictures on D2L onto your computer.
Add the apple/apple/apple picture (aaa) to a new label.
Add the label to newSlotsPanel.
Put a switch statement in your ButtonListener class that will display the appropriate picture when the *spin* button is pushed (use a random number generator).

Example code:

```
switch (line)
 {
    case 0:
    ImageIcon icon = new ImageIcon ("C:/Users/Eller/Pictures/2011-06-06/aaa.png");
    imageLabel.setIcon (icon);
    break;

    case 1:
    ImageIcon icon1 = new ImageIcon ("C:/Users/eller/Pictures/2011-06-06/aao.png");
    imageLabel.setIcon(icon1);
    break;
```

Remember you can use the statement if (event.getSource() == *button*) to determine which button is pushed.

Note: you do not have to implement the play again feature.

Assignment for Class 17

Review FontDemo, StyleOptions, StyleOptionsPane, QuoteOptions, QuoteOptionsPane, RobotFace, Robots

Read 7.10, 7.11, 7.12