Specifying a Drawing Color

When drawing shapes with methods such as **strokeOval** we can change colors. The method **setFill**, which is in the class **GraphicsContext**, will change the color of your pen.

Note: The setColor method sets the color of the pen

For example, the happy face that is drawn in using **Example below** is a yellow face with blue eyes and red lips. Aside from the color, the face is basically the same as the one shown in **example in Lesson 2**, except that we have now added a nose.

```
import javafx.application.Application;
import javafx.scene.canvas.Canvas;
import javafx.scene.Scene;
import javafx.scene.Group;
import javafx.stage.Stage;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.shape.ArcType;
import javafx.scene.paint.Color;
public class YellowFace extends Application
{
   public static final int WINDOW_WIDTH = 400;
   public static final int WINDOW_HEIGHT = 300;
   public static final int FACE_DIAMETER = 200;
   public static final int X_FACE = 100;
   public static final int Y_FACE = 50;
   public static final int EYE_WIDTH = 10;
   public static final int EYE_HEIGHT = 20;
   public static final int X_RIGHT_EYE = 155;
   public static final int Y_RIGHT_EYE = 100;
   public static final int X_LEFT_EYE = 230;
   public static final int Y_LEFT_EYE = Y_RIGHT_EYE;
   public static final int NOSE_DIAMETER = 10;
   public static final int X_NOSE = 195; // Center of nose at 200
   public static final int Y_NOSE = 135;
   public static final int MOUTH_WIDTH = 100;
   public static final int MOUTH_HEIGHT = 50;
   public static final int X_MOUTH = 150;
   public static final int Y_MOUTH = 160;
   public static final int MOUTH_START_ANGLE = 180;
   public static final int MOUTH_DEGREES_SHOWN = 180;
   public static void main(String[] args)
   {
      launch(args);
   }
   @Override
   public void start(Stage primaryStage) throws Exception
   {
      Group root = new Group();
      Scene scene = new Scene(root);
      Canvas canvas = new Canvas(WINDOW_WIDTH, WINDOW_HEIGHT);
      GraphicsContext gc = canvas.getGraphicsContext2D();
      // Draw face interior in yellow and outline in black
      gc.setFill(Color.YELLOW);
      gc.fillOval(X_FACE, Y_FACE, FACE_DIAMETER, FACE_DIAMETER);
      gc.setFill(Color.BLACK);
      gc.strokeOval(X_FACE, Y_FACE, FACE_DIAMETER, FACE_DIAMETER);
      // Draw eyes
      gc.setFill(Color.BLUE);
      gc.fillOval(X_RIGHT_EYE, Y_RIGHT_EYE, EYE_WIDTH, EYE_HEIGHT);
      gc.fillOval(X_LEFT_EYE, Y_LEFT_EYE, EYE_WIDTH, EYE_HEIGHT);
      // Draw nose
      gc.setFill(Color.BLACK);
      gc.setFill(Color.BLACK);
      gc.fillOval(X_NOSE, Y_NOSE, NOSE_DIAMETER, NOSE_DIAMETER);
      // Draw mouth
      gc.setFill(Color.RED);
      gc.strokeArc(X_MOUTH, Y_MOUTH, MOUTH_WIDTH, MOUTH_HEIGHT,
                   MOUTH_START_ANGLE, MOUTH_DEGREES_SHOWN, ArcType.OPEN);
      root.getChildren().add(canvas);
      primaryStage.setTitle("HappyFace in JavaFX");
      primaryStage.setScene(scene);
      primaryStage.show();
   }
}
```

The statement

gc.setFill(Color.YELLOW);

sets the color of the pen to yellow. So now the statement

gc.fillOval(X_FACE, Y_FACE, FACE_DIAMETER, FACE_DIAMETER);

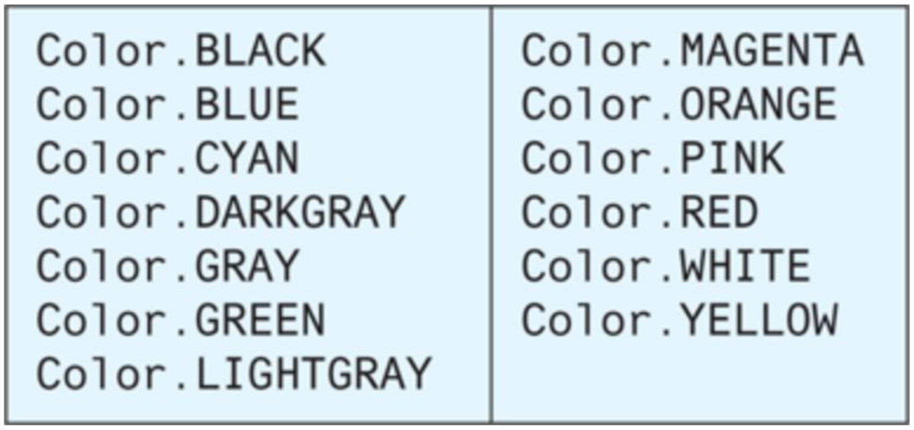
draws a circle for the face that is filled in with yellow.

The order in which you draw the components of the face affects the final outcome. Note that the solid yellow circle is drawn first, so that the other drawings, such as the eyes, will be on top of the yellow. As when using an actual pen or brush, the drawings are done one on top of the other in the order of the code in the paint method. If we had instead drawn the yellow circle last, it would be on top of the eyes, nose, and mouth, and only the yellow circle would be visible. Unlike drawing with an actual pen, however, placing one color over another does not blend the two colors. The newest drawing hides—actually, replaces—any earlier drawing made in the same spot. You are simply setting the state and color of pixels, replacing their earlier values.

Note: The order in which you draw affects the outcome

Certain colors are already defined for you as **public named constants** in the class **Color** which you import from **javafx.scene.paint.Color**

Some Predefined Colors for the setFill Method:



Color.BLACK	Color.MAGENTA
Color.BLUE	Color.ORANGE
Color.CYAN	Color.PINK
Color.DARKGRAY	Color.RED
Color.GRAY	Color.WHITE
Color.GREEN	Color.YELLOW
Color.LIGHTGRAY	

The setFill Method: When you draw using an object of the class **GraphicsContext**, you can set the color of the drawing by invoking the method **setFill**. You can later change the color by invoking **setFill** again, so a single drawing can have multiple colors.

Example

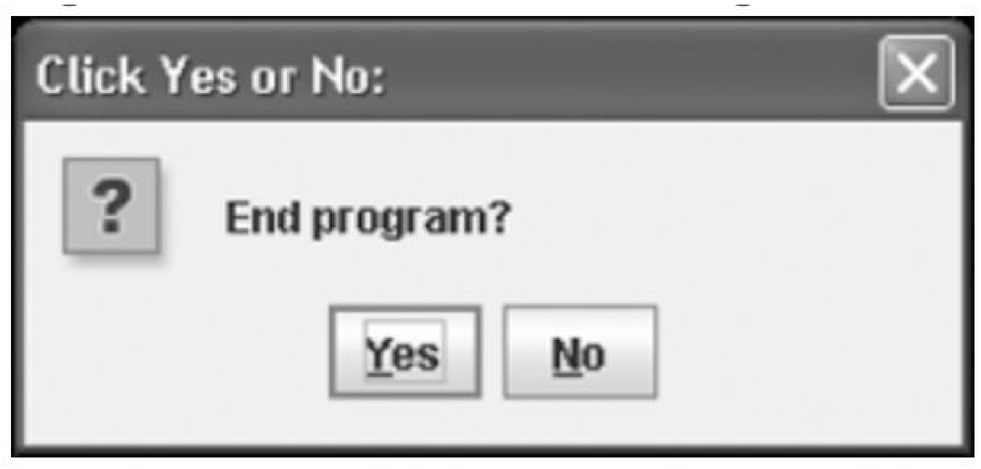
gc.setFill(Color.RED);

A Dialog Box for a Yes-or-No Question

Lesson 2 showed how to use the class **JOptionPane** to produce two kinds of windows called dialogs. One kind asked the user to **enter input** data that the program could read. The other kind of dialog **displayed a message** to the user. **JOptionPane provides yet another dialog box for asking yes-or-no questions of the user. The dialog has a title and contains the question you specify along with two buttons labeled Yes and No. For example,** the code:

```
int answer =
    JOptionPane.showConfirmDialog(null, "End program?",
            "Click Yes or No:", JOptionPane.YES_NO_OPTION);
if (answer == JOptionPane.YES_OPTION)
    System.exit(0);
else if (answer == JOptionPane.NO_OPTION)
    System.out.println("One more time");
else
    System.out.println("This is impossible");
```

produces the dialog box shown below.



Let’s describe the list of arguments for the method **showConfirmDialog** by considering the argument list in our example:

(null, "End program?", "Click Yes or No:", JOptionPane.YES_NO_OPTION)

The first argument has to do with where the dialog is placed on the screen, but we have not developed enough material to allow us to consider the possible options. So we will simply write **null** as the first argument, which gives us the default placement.

The second argument is a string that appears in the dialog containing the Yes and No buttons. Of course, the string should normally be a yes-or-no question. In our example, this argument is **"End program?"**.

The third argument is a string displayed as the title of the dialog. In our example, this argument is **"Click Yes or No:"**.

The last argument, JOptionPane.YES_NO_OPTION, indicates that we want a dialog containing Yes and No buttons. Other options are possible, but we will not discuss them here.

If the user clicks the **Yes** button, the method **showConfirmDialog** will return the **int value JOptionPane.YES_OPTION** and the window will disappear. In our example, the multi branch if-else statement will then invoke **System.exit(0)** to end the program. If the user clicks the **No** button, **the method showConfirmDialog will return the intvalue JOptionPane.NO_OPTION**, and then the multibranch if-else statement will invoke **System.out.println** to display One more time.

The class **JOptionPane** defines several named constants, including the int constants **YES_NO_OPTION, YES_OPTION, and NO_OPTION** that we have used here. **Note that to use these constants, you must precede each of their names with the name of the class and a dot.** What int values are actually named by these constants? It does not matter. Think of the value that **showConfirmDialog** returns as the answer to a yes-or-no question.

Dialog Boxes for Yes-or-No Questions: The class **JOptionPane** in the package **javax.swing** defines the method **show-ConfirmDialog**. You can use this method to create a dialog box to ask a yes-or-no question and get the user's response.

Syntax

Integer_Response **= JOptionPane.showConfirmDialog(null,** ***Question_String, Title_String, Option*****);**

The displayed dialog box is titled ***Title_String*** and contains the text ***Question_String*** and buttons as indicated by ***Option***. When *Option* is **JOptionPane.YES_NO_OPTION**, two buttons labeled **Yes** and **No** are displayed. The method returns the int constant

YES_OPTION if the user clicks the **Yes** button, or **NO_OPTION** if the user clicks the **No** button.

Example

```
int answer = JOptionPane.showConfirmDialog(null, "Done?",
                                    "Click a Button:", JOptionPane.YES_NO_OPTION);
```