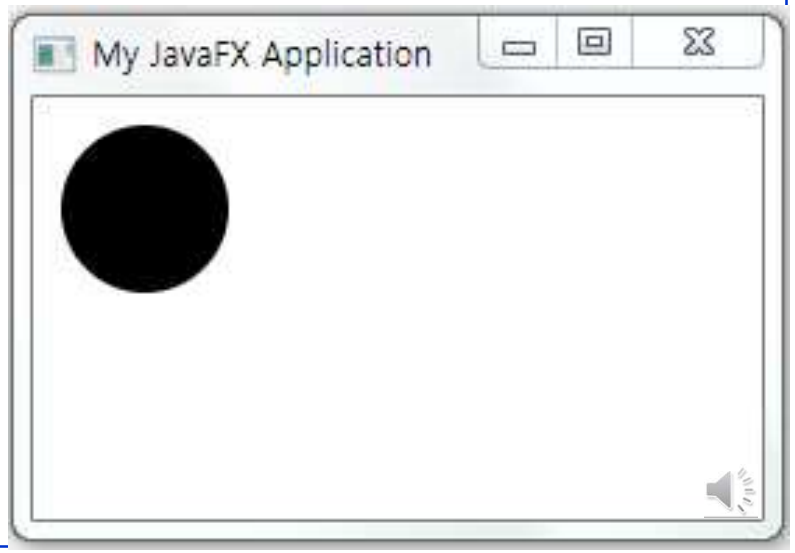
5.4 (optional) Graphics Supplement: Outline

- The GraphicsContext Class
- Adding Labels to a JavaFX Application

- JavaFX
  - [redacted]을 대체하기 위해 만들어짐
  - Swing보다 가볍고, 그래픽을 지원(그래프, 이미지, 영상, 사운드 등의 멀티미디어)하며, Web에서 호환 가능
  - Linux, Windows, OS X 등 Java를 설치할 수 있는 플랫폼이라면 어디서든지 사용 가능

```
import javafx.application.Application;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.shape.Circle;
import javafx.stage.Stage;

public class MyApplicationExample extends Application {
    public void start(Stage stage) {
        Circle circ = new Circle(40, 40, 30);
        Group root = new Group(circ);
        Scene scene = new Scene(root, 400, 300);

        stage.setTitle("My JavaFX Application");
        stage.setScene(scene);
        stage.show();
    }
   public static void main(String[] args)
   {
     launch(args);
   }
}
```



- Application
  - public [redacted] class Application extends Object
  - Application class from which JavaFX applications extend.
  - JavaFX creates an application thread for running the application start method, processing input events, and running animation timelines

- public void start(Stage primaryStage) throws Exception
  - abstract class Application 안에 있는 abstract
  - The main entry point for all JavaFX applications. The start method is called after the method has returned, and after the system is ready for the application to begin running.
  - NOTE: This method is called on the JavaFX Application Thread.

```
public void start(Stage stage) {
    Circle circ = new Circle(40, 40, 30);
    Group root = new Group(circ);
    Scene scene = new Scene(root, 400, 300);

    stage.setTitle("My JavaFX Application");
    stage.setScene(scene);
    stage.show();
}
```

- Stage
  - the top level JavaFX container.
  - The primary Stage is constructed by the platform.
  - Additional Stage objects may be constructed by the application.
  - Stage objects must be constructed and modified on the JavaFX Application Thread.

- [redacted]
  - "최상위 컨테이너“
  - GUI 기본 토대가 되는 컨테이너
  - AWT/Swing에서는 Frame과 JFrame 같은 클래스에서 윈도우를 만들고 표시
  - JavaFX의 경우 이 Stage를 사용하여 윈도우를 구축하는 것이 기본
  - "GUI를 통합하는 윈도우 본체는 JavaFX 측에서 준비되어 있다“
  - 응용 프로그램을 시작하면 처음에 표시되는 응용 프로그램의 윈도우로 Stage 인스턴스가 start에 전달
  - 프로그래머는 단지 그 전달된 Stage를 이용하여 GUI를 구축

```
import javafx.application.Application;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.text.Font;
import javafx.scene.text.Text;
import javafx.stage.Stage;

public class StageExample extends Application {

    @Override
        public void start(Stage stage) {
        Text text = new Text(10, 40, "Hello World!");
        text.setFont(new Font(40));
        Scene scene = new Scene(new Group(text));

        stage.setTitle("Welcome to JavaFX!");
        stage.setScene(scene);
        stage.sizeToScene();
        stage.show();
    }

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

- public void [redacted]()
  - Set the width and height of this Window to match the size of the content of this Window's Scene.

```
public void start(Stage stage) {
    Text text = new Text(10, 40, "Hello World!");
    text.setFont(new Font(40));
    Scene scene = new Scene(new Group(text));

    stage.setTitle("Welcome to JavaFX!");
    stage.setScene(scene);
    stage.sizeToScene();
    stage.show();
}
```

```
public void start(Stage stage) {
Text text = new Text(10, 40, "Hello World!");
text.setFont(new Font(40));
Scene scene = new Scene(new Group(text));

stage.setTitle("Welcome to JavaFX!");
stage.setScene(scene);
stage.sizeToScene();
stage.show();
}
```

- ()
  - start 메소드에서는 인자로 전달된 Stage 인스턴스의 "" 메소드를 호출
  - 그 Stage에서 구축된 윈도우를 화면에 표시하는 것
  - 반대로 윈도우를 숨기는 "hide"라는 메소드도 포함

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

- Launch()
  - [redacted] Class의 method
  - public [redacted] void launch(String... args)
  - Launch a standalone application.
  - This method is typically called from the main method().
  - It must not be called more than [redacted] or an exception will be thrown

```
public void start(Stage stage) {
        Text text = new Text(10, 40, "Hello World!");
        text.setFont(new Font(40));
        Scene scene = new Scene(new Group(text));

        stage.setTitle("Welcome to JavaFX!");
        stage.setScene(scene);
        stage.sizeToScene();
        stage.show();
    }
```

- JavaFX applications
  - use the metaphor of a **Stage** and **Scene**
- The object
  - the container for all content in a scene graph
  - contains a set of **Nodes** called a **scene graph** that desc ribes a scene of the applications, just like the script , actors, and props describe a scene in a play or m ovie
- Group
  - a special type of Node called a **Group** that contains other Nodes and can auto-size based on the nodes within the group

Scene

```
public Scene(Parent root,
             double width,
             double height)
```

Creates a Scene for a specific root Node with a specific size.

Parameters:

`root` - The root node of the scene graph

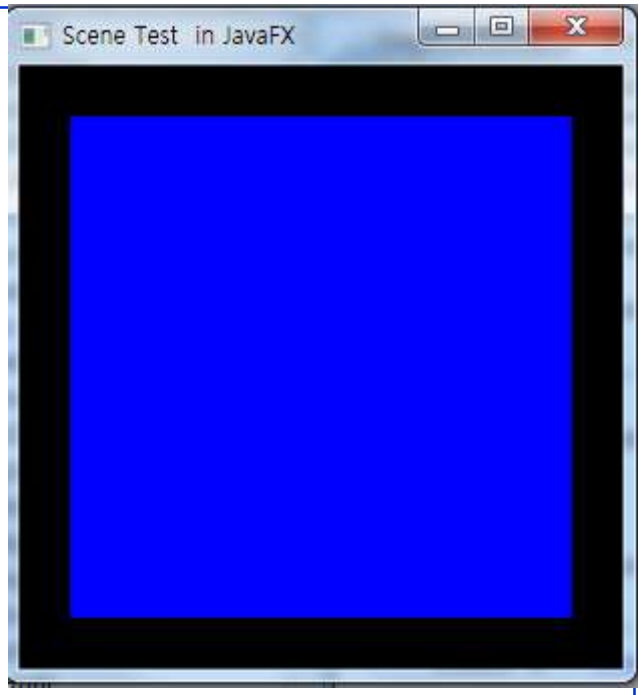
`width` - The width of the scene

`height` - The height of the scene

Throws:

`NullPointerException` - if root is null

```
import javafx.application.Application;
import javafx.scene.*;
import javafx.scene.paint.*;
import javafx.scene.shape.*;
import java.lang.Math;
import javafx.stage.Stage;

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

   @Override
   public void start(Stage primaryStage) throws Exception
   {
    Group root = new Group();
    Scene s = new Scene(root, 300, 300, Color.BLACK);

    Rectangle r = new Rectangle(25,25,250,250);
    r.setFill(Color.BLUE);
    root.getChildren().add(r);

            primaryStage.setTitle("Scene Test  in JavaFX");
            primaryStage.setScene(s);
            primaryStage.show();

   }
}
```



getChildren

```
public ObservableList<Node> getChildren()
```

Gets the list of children of this `Group`.

Overrides:

`getChildren` in class `Parent`

Returns:

the list of children of this `Group`.

```
import javafx.application.Application;
import javafx.scene.*;
import javafx.scene.paint.*;
import javafx.scene.shape.*;
import java.lang.Math;
import javafx.stage.Stage;

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

   @Override
   public void start(Stage primaryStage) throws Exception
   {
    Group g = new Group();
    Scene s = new Scene(g, 300, 300, Color.BLACK);

    for (int i = 0; i < 5; i++) {
         Rectangle r = new Rectangle();
         r.setY(i * 20);
         r.setWidth(100);
         r.setHeight(10);
         r.setFill(Color.RED);
         g.getChildren().add(r);
     }
                 primaryStage.setTitle("Group Test  in JavaFX");
                  primaryStage.setScene(s);
                  primaryStage.show();
   }
}
```

The [redacted] Class

- An object of the [redacted] class
  - represents an area of the screen
- [redacted]
  - a class that allows us to draw figures and write text in an area of the application
  - used to issue draw calls to a Canvas using a buffer.
  - A Canvas only contains [redacted] GraphicsContext, and only one buffer

The **GraphicsContext** Class

- Figure 5.8a Some methods in class **GraphicsContext**

gc.`strokeOval`(*X*, *Y*, *Width*, *Height*) Draws the outline of an oval having the specified width and height at the point (*X*, *Y*).
gc.`fillOval`(*X*, *Y*, *Width*, *Height*) Same as `strokeOval`, but the oval is filled in.
gc.`strokeArc`(*X*, *Y*, *Width*, *Height*, *StartAngle*, *ArcAngle*, *ArcType*) Draws an arc—that is, draws part of an oval. See the graphics supplement section of Chapter 1 for details.
gc.`fillArc`(*X*, *Y*, *Width*, *Height*, *StartAngle*, *ArcAngle*, *ArcType*) Same as `strokeArc`, but the visible portion of the oval is filled in.

The **GraphicsContext** Class

- Figure 5.8b Some methods in class **GraphicsContext**

gc.`strokeRect`(*X*, *Y*, *Width*, *Height*) Draws the outline of a rectangle having the specified width and height at the point (*X*, *Y*).
gc.`fillRect`(*X*, *Y*, *Width*, *Height*) Same as `strokeRect`, but the rectangle is filled in.
gc.`strokeLine`(*X1*, *Y1*, *X2*, *Y2*) Draws a line between points (*X1*, *Y1*) and (*X2*, *Y2*).
gc.`fillText`(*String*, *X*, *Y*) Draws the specified string at point (*X*, *Y*).

The GraphicsContext Class

- Figure 5.8c Some methods in class **GraphicsContext**

gc.`setLineWidth`(*Width*) Sets the current line width.
gc.`setFill`(*Attribute*) Sets the current fill paint attribute. We have been using color as the fill attribute to set the current color, but gradients or image patterns are other attributes.
gc.`setFont`(*Font*) Sets the current font.
gc.`drawImage`(*Image*, *X*, *Y*) Draws the specified image at point (*X*, *Y*).
gc.`setEffect`(*Effect*) Sets the effect to be applied to the next drawing call.

Drawing Images and Applying Effects

- the drawImage method
  - to draw an image on the screen and apply a variety of image effects
- View sample program, listing 5.23, fo r reading an image file and drawing it with scale and reflection

Listing 5.23 Drawing Images and Applying Effects

```
import javafx.scene.Group;
import javafx.stage.Stage;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.shape.ArcType;
import javafx.scene.paint.Color;
import javafx.scene.image.Image;
import javafx.scene.effect.Reflection;

public class ImageExample extends Application
{
  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);
```

```
        // Java looks for " hgulogo.png" in the default folder
        Image img = new Image("hgulogo.png");
        Canvas canvas = new Canvas(400, 400);
        GraphicsContext gc = canvas.getGraphicsContext2D();
        // Draw image in normal scaling at (1,1)
        gc.drawImage(img, 1, 1);
        // Draw image twice as large to the right of
        // previous image
        gc.drawImage(img, img.getWidth() + 10, 1,
                img.getWidth() * 2, img.getHeight() * 2);

        // Draw image below the first with a reflection effect
        gc.setEffect(new Reflection());
        gc.drawImage(img, 1, img.getHeight() *2);

        root.getChildren().add(canvas);
        primaryStage.setTitle("Drawing Images in JavaFX");
        primaryStage.setScene(scene);
        primaryStage.show();
    }
}
```

javafx.scene.canvas

Class Canvas

java.lang.Object
    javafx.scene.Node
        javafx.scene.canvas.Canvas

All Implemented Interfaces:

Styleable, EventTarget

```
public class Canvas
extends Node
```

Canvas is an image that can be drawn on using a set of graphics commands provided by a GraphicsContext.

A Canvas node is constructed with a width and height that specifies the size of the image into which the canvas drawing commands are rendered. All drawing operations are clipped to the bounds of that image.

getGraphicsContext2D

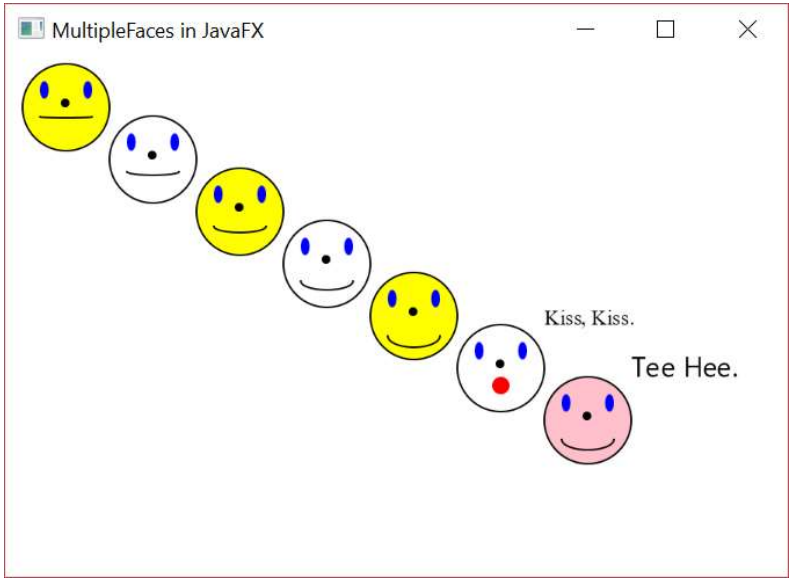
```
public GraphicsContext getGraphicsContext2D()
```

returns the GraphicsContext associated with this Canvas.

Programming Example

- Multiple faces – using a Helping method
- View sample code, listing 5.24

```
class MultipleFaces
```



Sample screen output

LISTING 5.24 Using a Method for a Recurrent Subtask

```
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;
import javafx.scene.text.Font;

public class MultipleFaces extends Application
{
   public static final int WINDOW_WIDTH = 450;
   public static final int WINDOW_HEIGHT = 300;

   public static final int FACE_DIAMETER = 50;
   public static final int X_FACE0 = 10;
   public static final int Y_FACE0 = 5;

   public static final int EYE_WIDTH = 5;
   public static final int EYE_HEIGHT = 10;
   public static final int X_RIGHT_EYE0 = 20;
   public static final int Y_RIGHT_EYE0 = 15;
   public static final int X_LEFT_EYE0 = 45;
   public static final int Y_LEFT_EYE0 = Y_RIGHT_EYE0;

   public static final int NOSE_DIAMETER = 5;
   public static final int X_NOSE0 = 32;
   public static final int Y_NOSE0 = 25;

   public static final int MOUTH_WIDTH = 30;
   public static final int MOUTH_HEIGHT0 = 0;
   public static final int X_MOUTH0 = 20;
   public static final int Y_MOUTH0 = 35;
   public static final int MOUTH_START_ANGLE = 180;
   public static final int MOUTH_EXTENT_ANGLE = 180;
```

```java
public static void main(String[] args)
{
   launch(args);
}
/**
gc is the drawing area. pos indicates the position of the
face. As pos increases, the face is drawn lower and further
to the right.
*/
private void drawFaceSansMouth(GraphicsContext gc, int pos)
{
   // Draw face oval
   gc.setFill(Color.BLACK);
   gc.strokeOval(X_FACE0 + 50 * pos, Y_FACE0 + 30 * pos,
                 FACE_DIAMETER, FACE_DIAMETER);
   // Draw eyes
   gc.setFill(Color.BLUE);

   gc.fillOval(X_RIGHT_EYE0 + 50 * pos, Y_RIGHT_EYE0 + 30 * pos,
               EYE_WIDTH, EYE_HEIGHT);
   gc.fillOval(X_LEFT_EYE0 + 50 * pos, Y_LEFT_EYE0 + 30 * pos,
               EYE_WIDTH, EYE_HEIGHT);
   //Draw nose
   gc.setFill(Color.BLACK);
   gc.fillOval(X_NOSE0 + 50 * pos, Y_NOSE0 + 30 * pos,
               NOSE_DIAMETER, NOSE_DIAMETER);
}
@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();

    int i;   //Want i to exist after the loop ends
```

```
for (i = 0; i <= 4; i++)
{//Draw one face:
   if (i % 2 == 0)//if i is even
   {//Make face yellow
       gc.setFill(Color.YELLOW);
       gc.fillOval(X_FACE0 + 50 * i, Y_FACE0 + 30 * i,
                                FACE_DIAMETER, FACE_DIAMETER);
   }
   drawFaceSansMouth(gc, i);
   //Draw mouth:
   gc.setFill(Color.RED);
   gc.strokeArc(X_MOUTH0 + 50 * i, Y_MOUTH0 + 30 * i,
                MOUTH_WIDTH, MOUTH_HEIGHT0 + 3 * i + 1,
                MOUTH_START_ANGLE, MOUTH_EXTENT_ANGLE,
                ArcType.OPEN);
}
//i is 5 when the previous loop ends

//Draw kissing face:
drawFaceSansMouth(gc, i);
//Draw mouth in shape of a kiss:
gc.setFill(Color.RED);
gc.fillOval(X_MOUTH0 + 50 * i + 10, Y_MOUTH0 + 30 * i,
              MOUTH_WIDTH - 20, MOUTH_WIDTH - 20);
//Add text in Times New Roman, 12 point:
gc.setFont(Font.font("Times New Roman", 12));
```

```
        gc.setFill(Color.BLACK);
        gc.fillText("Kiss, Kiss.",
               X_FACE0 + 50 * i + FACE_DIAMETER, Y_FACE0 + 30 * i);

        //Draw blushing face:
        i++;
        //Draw face circle:
        gc.setFill(Color.PINK);
        gc.fillOval(X_FACE0 + 50*i, Y_FACE0 + 30*i,
                      FACE_DIAMETER, FACE_DIAMETER);
        drawFaceSansMouth(gc, i);
        gc.setFill(Color.RED);

        //Draw mouth:
        gc.setFill(Color.BLACK);
        gc.strokeArc(X_MOUTH0 + 50*i, Y_MOUTH0 + 30*i, MOUTH_WIDTH,
                     MOUTH_HEIGHT0 + 3 * (i - 2),
                     MOUTH_START_ANGLE, MOUTH_EXTENT_ANGLE, ArcType.OPEN);
        //Add text in Courier New Font, 16 point:
        gc.setFont(Font.font("Courier New ", 16));
        gc.fillText("Tee Hee.",
              X_FACE0 + 50*i + FACE_DIAMETER, Y_FACE0 + 30*i);

      root.getChildren().add(canvas);
      primaryStage.setTitle("MultipleFaces in JavaFX");
      primaryStage.setScene(scene);
      primaryStage.show();
    }
}
```

Program Output

The drawing produced is identical to the one show in Listing 4.9 except for some of the colors used to draw the faces.

Adding Labels to a JavaFX Application

- Provides a way to add text to an application
- More flexible than using **`fillText`**
- Add a **`Label`** in conjunction with a [redacted] which specifies how components will be arranged visually in the window
  - Here we use a simple vertical box layout, [redacted]

Listing 5.25 Adding Labels to a JavaFX Application

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.stage.Stage;
import javafx.scene.text.Font;
import javafx.scene.layout.VBox;
import javafx.scene.control.Label;

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

```
@Override
  public void start(Stage primaryStage) throws Exception
  {
        VBox root = new VBox();
        Label label1, label2;
        label1 = new Label("Hello");
        label1.setFont(Font.font("Times New Roman", 24));
        label2 = new Label("Out there!");
        label2.setFont(Font.font("Courier New", 36));

        root.getChildren().add(label1);
        root.getChildren().add(label2);

        Scene scene = new Scene(root, 300, 100);

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

끝