**고급 JAVA 프로그래밍**

11주차 과제

1826074 오현진

**<Bindings 클래스>**

JavaFX 속성은 다른 속성과 연결하는 속성 바인딩이 가능합니다.

이때 한 속성이 변경되면, 바인딩된 다른 속성도 자동으로 변경됩니다.

그러나 바인딩 과정에서, 바인딩되는 두 개의 속성이 항상 같은 값과 타입을 가질 수는 없습니다.

그렇기 때문에 속성값들의 단순 연산 후의 바인딩 뿐만 아니라, 두 속성값을 동일하게 하는 연산작업을 수행한 후 바인딩하기 위해 저희가 JavaFX의 Binding 클래스를 배운다고 저는 이해했습니다.

다음 예제는 강의 자료를 참고하여 같은 속성을 바인딩 해보았습니다.

**package** application;

**import** javafx.scene.control.\*;

**import** javafx.application.Application;

**import** javafx.beans.binding.Bindings;

**import** javafx.geometry.\*;

**import** javafx.scene.Scene;

**import** javafx.scene.layout.\*;

**import** javafx.stage.Stage;

**public** **class** AppMain **extends** Application {

@Override

**public** **void** start(Stage primaryStage) **throws** Exception

{

VBox root = **new** VBox();

root.setPadding(**new** Insets(10));

root.getChildren().add(**new** Label("textArea1"));

TextArea textArea1 = **new** TextArea();

root.getChildren().add(textArea1);

root.getChildren().add(**new** Label("textArea2"));

TextArea textArea2 = **new** TextArea();

root.getChildren().add(textArea2);

Bindings.*bindBidirectional*(textArea1.textProperty(), textArea2.textProperty());

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

primaryStage.setScene(scene);

primaryStage.show();

}

**public** **static** **void** main(String[] args) {

*launch*(args);

}

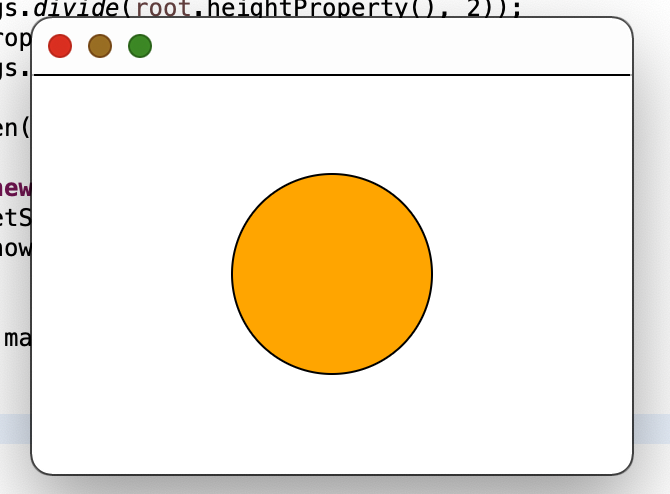
}

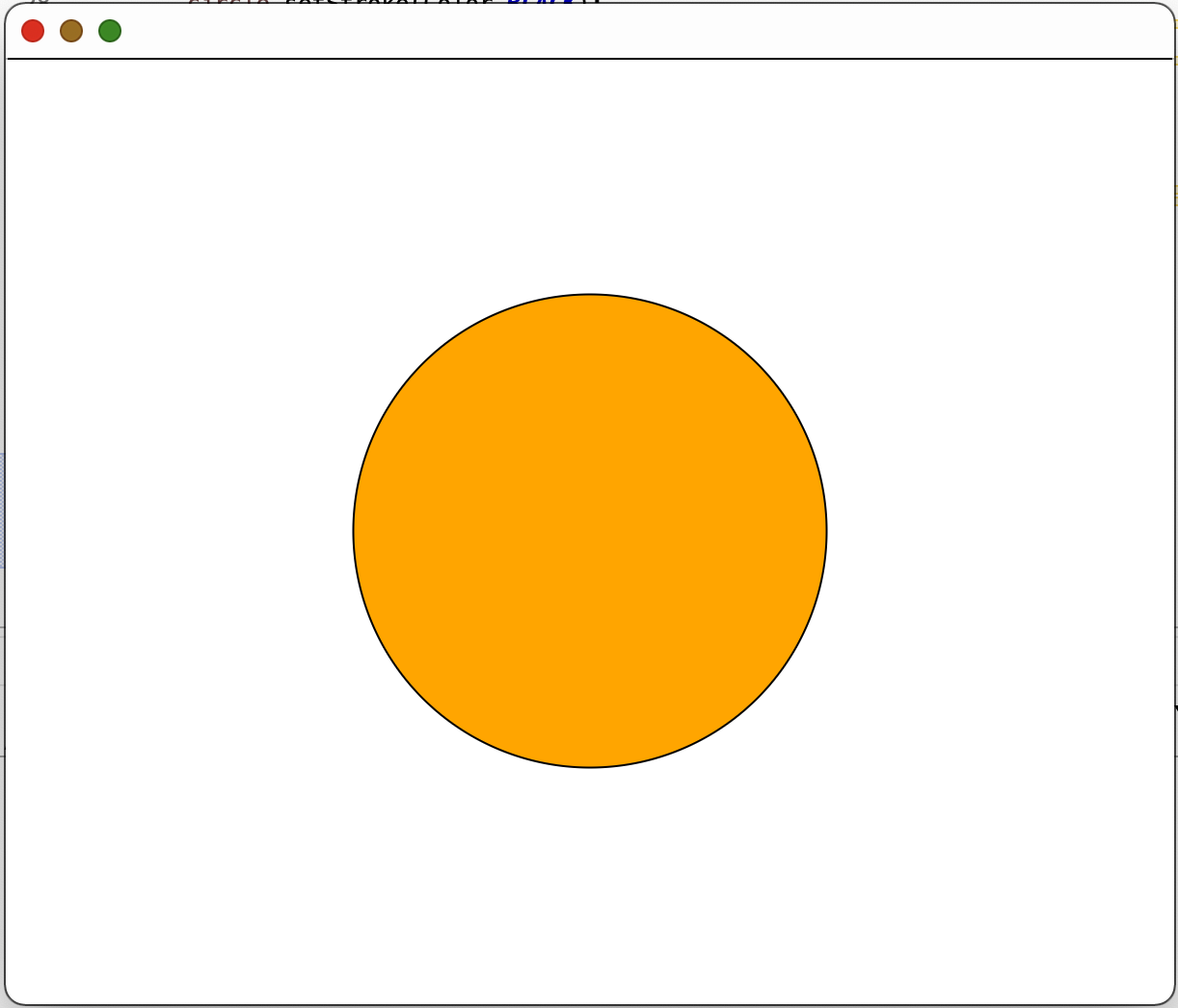
텍스트이(가) 표시된 사진

자동 생성된 설명

실행 예시에서 보이는 것과 같이, textArea1과 textArea2가 바인딩되어, textArea1에 텍스트를 입력하면, textArea2도 따라 텍스트가 입력되는 모습입니다.

다음 예제는 AnchorPane의 크기에 상관없이 무조건 중앙에 Circle을 표시하는 예제입니다. AnchorPane의 크기와 관계없이 항상 원의 중심이 AnchorPane의 가운데에 오도록, 원의 중심을 divide메소드를 이용하여, X와 Y좌표를 AnchorPane의 너비와 길이를 2로 나눈값에 바인딩하여, AnchorPane의 너비와 길이가 어떻게 변하던지, 바인딩에 의해 원의 중심 좌표도 항상 변한 너비와 길이의 1/2에 해당하는 값을 가지게 됩니다.





실행 예시에서 보이는 것과 같이 divide메소드를 이용해 AnchorPane의 중심에 원의 중심 좌표를 바인딩 했기 때문에, AnchorPane의 크기에 관계없이 원도 항상 AnchorPane의 중심에 위치하는 모습입니다.