JavaFX 2: Control, Event and Event Handler

*First, go to Lec2\_Classe&Method, bring up the power point slides for Chpt4: Using Classes and Objects. Start from slide#64: Arcs. Read all the slides from there to end and go over the examples. Then come to this lecture note.*

We are going to go over some examples here which will display the use of control elements like button, text field, etc; event and event-handler.

# Example 1:

Write a JavaFX program that allows the user to pick a set of pizza toppings using a set of check boxes. Assuming each toppings cost 50 cents, and a plain pizza costs $10, display the cost of the pizza.

**package** application;

**import** java.text.NumberFormat;

**import** javafx.application.Application;

**import** javafx.event.ActionEvent;

**import** javafx.geometry.HPos;

**import** javafx.geometry.Pos;

**import** javafx.scene.Scene;

**import** javafx.scene.control.CheckBox;

**import** javafx.scene.layout.GridPane;

**import** javafx.scene.layout.VBox;

**import** javafx.scene.text.Text;

**import** javafx.stage.Stage;

**import** javafx.scene.text.Font;

**public** **class** PizzaCost **extends** Application {

**private** **double** cost;

**private** Text costText;

**private** NumberFormat fmt = NumberFormat.*getCurrencyInstance*();

//Presents a set of check boxes representing pizza toppings and

//displays the total cost of the pizza.

**public** **void** start(Stage primaryStage) {

cost = 10.0; // cost of plain pizza

CheckBox extraCheese = **new** CheckBox("Extra Cheese");

extraCheese.setOnAction(**this**::processCheckBoxAction);

CheckBox pepperoni = **new** CheckBox("Pepperoni");

pepperoni.setOnAction(**this**::processCheckBoxAction);

CheckBox sausage = **new** CheckBox("Sausage");

sausage.setOnAction(**this**::processCheckBoxAction);

CheckBox greenPepper = **new** CheckBox("Green Pepper");

greenPepper.setOnAction(**this**::processCheckBoxAction);

CheckBox onion = **new** CheckBox("Onion");

onion.setOnAction(**this**::processCheckBoxAction);

CheckBox anchovies = **new** CheckBox("Anchovies");

anchovies.setOnAction(**this**::processCheckBoxAction);

VBox toppings1 = **new** VBox(extraCheese, pepperoni, sausage);

toppings1.setSpacing(5);

VBox toppings2 = **new** VBox(greenPepper, onion, anchovies);

toppings2.setSpacing(5);

costText = **new** Text("Pizza Cost: $10.00");

costText.setFont(**new** Font("Helvetica", 18));

GridPane.*setHalignment*(costText, HPos.***CENTER***);

GridPane root = **new** GridPane();

root.setAlignment(Pos.***CENTER***);

root.setHgap(20);

root.setVgap(30);

root.setStyle("-fx-background-color: cornsilk");

root.add(toppings1, 0, 0);

root.add(toppings2, 1, 0);

root.add(costText, 0, 1, 2, 1);

Scene scene = **new** Scene(root, 400, 150);

primaryStage.setTitle("Pizza Cost");

primaryStage.setScene(scene);

primaryStage.show();

}

//Updates cost based on a topping being added or removed.

**public** **void** processCheckBoxAction(ActionEvent event) {

CheckBox box = (CheckBox) event.getSource();

**if** (box.isSelected())

cost += 0.50;

**else**

cost -= 0.50;

costText.setText("Pizza Cost: " + fmt.format(cost));

}

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

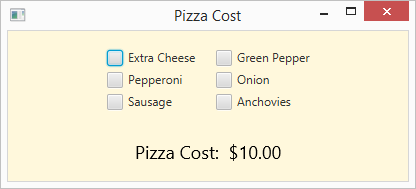
{

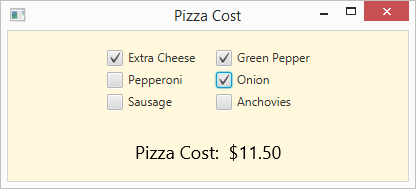
*launch*(args);

}

}

Run the program and we get the following:





## **Description:**

The following is a new set of code:

VBox toppings1 = **new** VBox(extraCheese, pepperoni, sausage);

VBox class is a layout pane that arranges a series of nodes vertically in one column. HBox arranges it horizontzlly in one row.

This is the package: javafx.scene.layout.VBox

VBox lays out its children in a single vertical column. If the vbox has a border and/or padding. When set, then the contents will be layed out within those insets.

VBox example:

VBox vbox = new VBox(8); // spacing = 8

vbox.getChildren().addAll(new Button("Cut"), new Button("Copy"), new Button("Paste"));

Other layout pane in JavaFX are: FlowPane, TilePane, StackPane, AnchorPane, BorderPane and GridPane.

GridPane root = **new** GridPane();

root.setStyle("-fx-background-color: cornsilk");

JavaFX has CSS support and extensions have been designed to allow JavaFX CSS styles sheets to be parsed cleanly by any compliant CSS parser, even though it might not support JavaFX extensions. This enables the mixing of CSS styles for JavaFX and for other purposes (such as for HTML pages) into a single style sheet. To this end, all JavaFX property names have been prefixed with a vendor extension of “-fx-“. Even properties that might seem to be compatible with standard HTML CSS have been pre-fixed, because JavaFX has somewhat different semantics for their values.

# Example 2:

Write a JavaFX program that allows the user to select a color out of five options provided by a set of radio buttons. Change the color of a displayed square accordingly.

**package** application;

**import** javafx.application.Application;

**import** javafx.event.ActionEvent;

**import** javafx.geometry.Pos;

**import** javafx.scene.Scene;

**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.paint.Color;

**import** javafx.scene.shape.Rectangle;

**import** javafx.stage.Stage;

**public** **class** Main **extends** Application {

**private** RadioButton red, green, orange, yellow, blue;

**private** Rectangle square;

//Presents a set of radio buttons providing color options, and a

//square filled with the selected color.

**public** **void** start(Stage primaryStage) {

ToggleGroup group = **new** ToggleGroup();

red = **new** RadioButton("Red");

red.setSelected(**true**);

red.setToggleGroup(group);

red.setOnAction(**this**::processRadioButtonAction);

green = **new** RadioButton("Green");

green.setToggleGroup(group);

green.setOnAction(**this**::processRadioButtonAction);

orange = **new** RadioButton("Orange");

orange.setToggleGroup(group);

orange.setOnAction(**this**::processRadioButtonAction);

yellow = **new** RadioButton("Yellow");

yellow.setToggleGroup(group);

yellow.setOnAction(**this**::processRadioButtonAction);

blue = **new** RadioButton("Blue");

blue.setToggleGroup(group);

blue.setOnAction(**this**::processRadioButtonAction);

VBox colors = **new** VBox(red, green, orange, yellow, blue);

colors.setAlignment(Pos.***CENTER\_LEFT***);

colors.setSpacing(10);

square = **new** Rectangle(100, 100);

square.setFill(Color.***RED***);

StackPane squarePane = **new** StackPane(square);

squarePane.setPrefSize(200, 100);

HBox root = **new** HBox(colors, squarePane);

root.setAlignment(Pos.***CENTER***);

root.setStyle("-fx-background-color: cornsilk");

Scene scene = **new** Scene(root, 400, 200);

primaryStage.setTitle("Square Color");

primaryStage.setScene(scene);

primaryStage.show();

}

//Updates the color of the square based on the selection.

**public** **void** processRadioButtonAction(ActionEvent event) {

**if** (red.isSelected())

square.setFill(Color.***RED***);

**else** **if** (green.isSelected())

square.setFill(Color.***GREEN***);

**else** **if** (orange.isSelected())

square.setFill(Color.***ORANGE***);

**else** **if** (yellow.isSelected())

square.setFill(Color.***YELLOW***);

**else**

square.setFill(Color.***BLUE***);

}

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

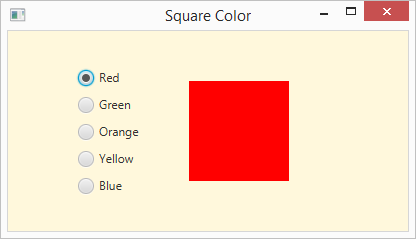
{

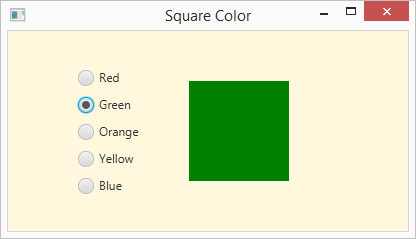
*launch*(args);

}

}

Run the program and we will get the following:





## **Description:**

This is new for you: ToggleGroup group = **new** ToggleGroup();

The class ToggleGroup contains a reference to all Toggles whose selected variables should be managed such that only a single Toggle within the ToggleGroup may be selected at any one time. This is similar to adding radio buttons to a group.

# Example 3:

Write a JavaFX application that presents four labelled text fields, allowing the user to enter values for name, age, favorite color, and hobby. Include a button labelled Print. When the button is pushed, the program should print the contents of all fields to the console window (standard output) using println statements.

**import** javafx.application.Application;

**import** javafx.event.ActionEvent;

**import** javafx.geometry.HPos;

**import** javafx.geometry.Pos;

**import** javafx.scene.Scene;

**import** javafx.scene.control.Button;

**import** javafx.scene.control.Label;

**import** javafx.scene.control.TextField;

**import** javafx.scene.layout.GridPane;

**import** javafx.scene.text.Font;

**import** javafx.stage.Stage;

**public** **class** ButtonInfoPrint **extends** Application {

**private** TextField nameField, ageField, colorField, hobbyField;

// Allows the user to enter information into text fields and print

// that information to the standard output stream with the press of a button.

**public** **void** start(Stage primaryStage) {

Font font = **new** Font(16);

Label nameLabel = **new** Label("Name:");

nameLabel.setFont(font);

GridPane.*setHalignment*(nameLabel, HPos.***RIGHT***);

Label ageLabel = **new** Label("Age:");

ageLabel.setFont(font);

GridPane.*setHalignment*(ageLabel, HPos.***RIGHT***);

Label colorLabel = **new** Label("Favorite Color:");

colorLabel.setFont(font);

GridPane.*setHalignment*(colorLabel, HPos.***RIGHT***);

Label hobbyLabel = **new** Label("Hobby:");

hobbyLabel.setFont(font);

GridPane.*setHalignment*(hobbyLabel, HPos.***RIGHT***);

nameField = **new** TextField();

nameField.setFont(font);

nameField.setMaxWidth(200);

ageField = **new** TextField();

ageField.setFont(font);

ageField.setMaxWidth(50);

colorField = **new** TextField();

colorField.setFont(font);

colorField.setMaxWidth(100);

hobbyField = **new** TextField();

hobbyField.setFont(font);

hobbyField.setMaxWidth(200);

Button printButton = **new** Button("Print");

GridPane.*setHalignment*(printButton, HPos.***CENTER***);

printButton.setOnAction(**this**::processButtonPress);

GridPane root = **new** GridPane();

root.setAlignment(Pos.***CENTER***);

root.setHgap(20);

root.setVgap(10);

root.setStyle("-fx-background-color: beige");

root.add(nameLabel, 0, 0);

root.add(nameField, 1, 0);

root.add(ageLabel, 0, 1);

root.add(ageField, 1, 1);

root.add(colorLabel, 0, 2);

root.add(colorField, 1, 2);

root.add(hobbyLabel, 0, 3);

root.add(hobbyField, 1, 3);

root.add(printButton, 0, 4, 2, 1);

Scene scene = **new** Scene(root, 500, 300);

primaryStage.setTitle("Button Info Print");

primaryStage.setScene(scene);

primaryStage.show();

}

// Converts and sets the new radius value of the circle.

**public** **void** processButtonPress(ActionEvent event) {

System.***out***.println(nameField.getText());

System.***out***.println(ageField.getText());

System.***out***.println(colorField.getText());

System.***out***.println(hobbyField.getText());

}

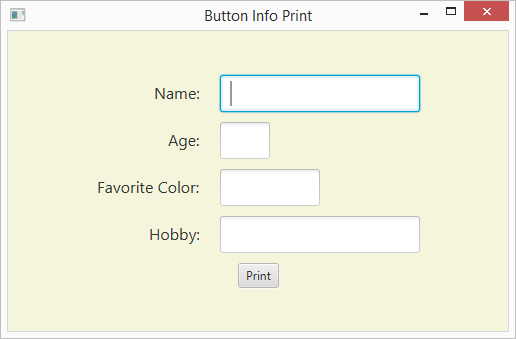
**public** **static** **void** main(String[] args) {

*launch*(args);

}

}

Run the program and we will get the following. Enter data and click on “Print”. The output will be displayed in the console.



## **Description:**

Note that the codes are not much different from that of previous examples. Note the multiple text boxes, their alignment and placements.