JavaFX Layout Containers

# Spacing and Padding

Spacing and padding are fields of the HBox and VBox layout containers.

Spacing

Spacing refers to the gaps between child nodes of these containers. Spacing is specified in pixels and can be assigned a value in the constructors of these classes.

When you pass a numerical value to the constructor of an HBox or VBox, this sets its padding.

HBox imageStrip = new HBox(20, image01, image02);

Alternatively, you can use the `.setSpacing()` method to update the container's padding:

imageStrip.setSpacing(15);

Padding

Padding refers to the space between the edges of the container and its children. It essentially creates an inner margin inside the container, pushing the children inward from the container's boundaries.

Padding is also specified in pixels. There is no parameter for padding in the constructors of these containers, so it must be set using the `setPadding()` method.

The `setPadding()` method takes an `Insets` object as an argument. The `Insets` class is defined in the `javafx.geometry` package and is used to define a space or margin around a component. The following are the two main constructors for creating an `Insets` object:

Insets insets = new Insets(double value); // This creates an `Insets` object with the same padding value on all four sides.

Insets insets = new Insets(double top, double right, double bottom, double left); // This allows you to specify the margins for all four sides starting at the top in clockwise order.

By default, the padding values for all JavaFX containers is 0.

The padding for a layout container can be set as follows:

HBox hbox = new HBox(10); // 10 pixels of spacing between children

hbox.setPadding(new Insets(10, 15, 10, 15);

# `GridPane`

The `GridPane` layout container arranges its children in columns and rows. A GridPane is divided into cells, like a spreadsheet. The columns and rows are identified by indexes, beginning at 0.

Controls can be added to a GridPane using the `add()` method:

GridPane gridPane = new GridPane();

gridPane.add(control, column, row);

By default, the grid lines in a GridPane are not displayed. However, they can be made visible using the following method:

gridPane.setGridLinesVisible(true);

By default, there is no spacing between the columns and rows in a GridPane. To add spacing, you can use the following methods:

gridPane.setHgap(10); // For horizontal spacing between columns in pixels

gridPane.setVgap(10); // For vertical spacing between rows in pixels

# Encapsulation of UI Element Creation

In small applications, it is common practice to include all UI elements and their setup directly within the `start` method. However, for more complex applications, it is best practice to encapsulate UI element creation and setup in separate methods to improve readability, maintainability and modularity.

This is done by creating a method that returns an object of the main `root` layout, such as a `StackPane`. This method would then be assigned to the root layout using code similar to the following:

public class Main extends Application{

@Override

public void start(Stage primaryStage){

StackPane root = getMainUI();

Scene scene = new Scene(root, 300, 250);

primaryStage.setScene(scene);

primaryStage.show();

}

private StackPane getMainUI(){

// Add all UI elements and their setup here.

}

public static void main(String[] args){

launch(args);

}

}

For any additional scenes other than the main one, it is best practice to encapsulate their creation in separate methods.