Lecture 5 - Java Graphical User Interface (GUI): JavaFX - Part II

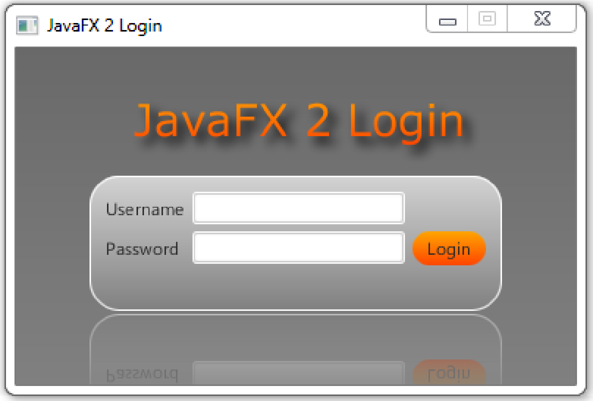
Guiliang Liu

The Chinese University of Hong Kong, Shenzhen

CSC-1004: Computational Laboratory Using Java
Course Page: [Click]

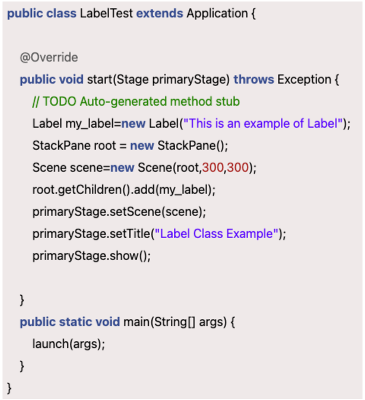
JavaFX UI Controls

JavaFX UI controls are the visual elements that form the building blocks of a JavaFX application's user interface. These controls are pre-built components that developers can use to construct the vinteractive parts of a GUI (Graphical User Interface).

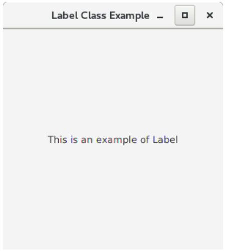


JavaFX UI Controls

- Label is a component that is used to define a simple text on the screen.



```
public class LabelTest extends Application {

  @Override
  public void start(Stage primaryStage) throws Exception {
    // TODO Auto-generated method stub
    Label my_label=new Label("This is an example of Label");
    StackPane root = new StackPane();
    Scene scene=new Scene(root,300,300);
    root.getChildren().add(my_label);
    primaryStage.setScene(scene);
    primaryStage.setTitle("Label Class Example");
    primaryStage.show();

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

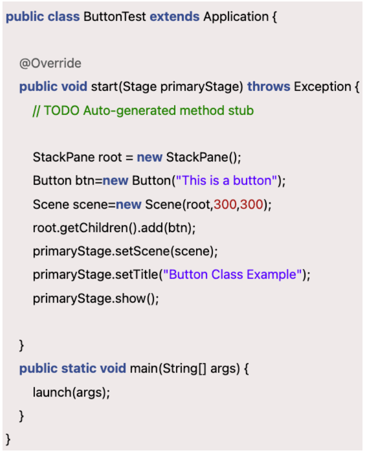


Label Class Example

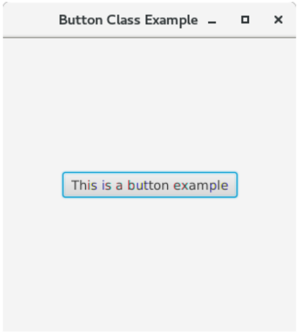
This is an example of Label

JavaFX UI Controls

- **Button** is a component that bluecontrols the function of the application. Button class is used to create a labeled button.

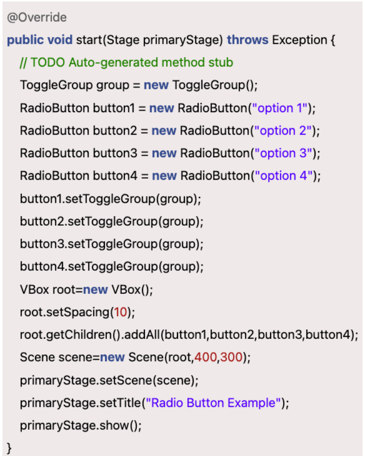


```
public class ButtonTest extends Application {

  @Override
  public void start(Stage primaryStage) throws Exception {
    // TODO Auto-generated method stub

    StackPane root = new StackPane();
    Button btn=new Button("This is a button");
    Scene scene=new Scene(root,300,300);
    root.getChildren().add(btn);
    primaryStage.setScene(scene);
    primaryStage.setTitle("Button Class Example");
    primaryStage.show();

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

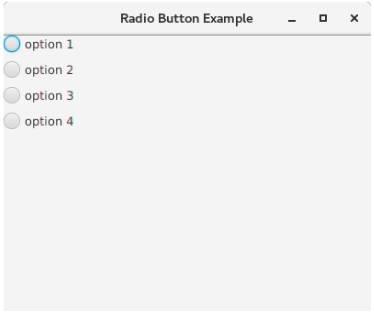


JavaFX UI Controls

- **The Radio Button** is used to provide various options to the user. The user can choose one option from all. A radio button is either selected or deselected.

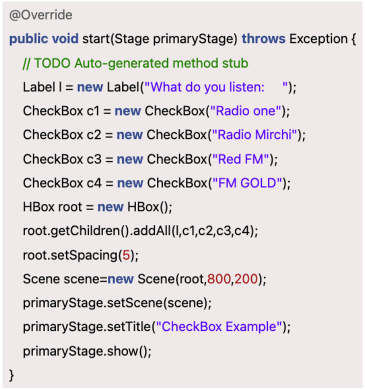


```
@Override
public void start(Stage primaryStage) throws Exception {
    // TODO Auto-generated method stub
    ToggleGroup group = new ToggleGroup();
    RadioButton button1 = new RadioButton("option 1");
    RadioButton button2 = new RadioButton("option 2");
    RadioButton button3 = new RadioButton("option 3");
    RadioButton button4 = new RadioButton("option 4");
    button1.setToggleGroup(group);
    button2.setToggleGroup(group);
    button3.setToggleGroup(group);
    button4.setToggleGroup(group);
    VBox root=new VBox();
    root.setSpacing(10);
    root.getChildren().addAll(button1,button2,button3,button4);
    Scene scene=new Scene(root,400,300);
    primaryStage.setScene(scene);
    primaryStage.setTitle("Radio Button Example");
    primaryStage.show();
}
```

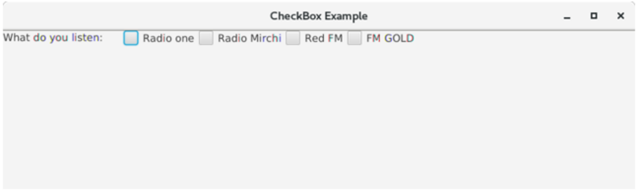


JavaFX UI Controls

- Check Box is used to get the kind of information from the user which contains various choices. The user marked the checkbox either on (true) or off(false).



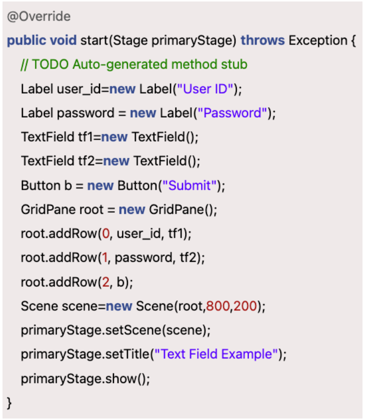
```
@Override
public void start(Stage primaryStage) throws Exception {
   // TODO Auto-generated method stub
   Label l = new Label("What do you listen:     ");
   CheckBox c1 = new CheckBox("Radio one");
   CheckBox c2 = new CheckBox("Radio Mirchi");
   CheckBox c3 = new CheckBox("Red FM");
   CheckBox c4 = new CheckBox("FM GOLD");
   HBox root = new HBox();
   root.getChildren().addAll(l,c1,c2,c3,c4);
   root.setSpacing(5);
   Scene scene=new Scene(root,800,200);
   primaryStage.setScene(scene);
   primaryStage.setTitle("CheckBox Example");
   primaryStage.show();
}
```



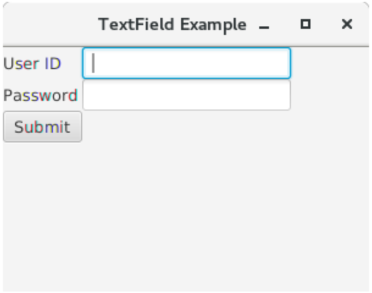
香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

JavaFX UI Controls

- Text Field is basically used to get input from the user in the form of text.



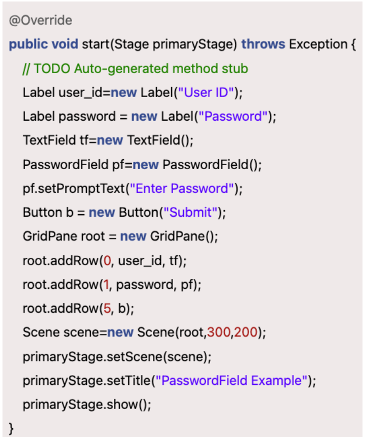
```
@Override
public void start(Stage primaryStage) throws Exception {
    // TODO Auto-generated method stub
    Label user_id=new Label("User ID");
    Label password = new Label("Password");
    TextField tf1=new TextField();
    TextField tf2=new TextField();
    Button b = new Button("Submit");
    GridPane root = new GridPane();
    root.addRow(0, user_id, tf1);
    root.addRow(1, password, tf2);
    root.addRow(2, b);
    Scene scene=new Scene(root,800,200);
    primaryStage.setScene(scene);
    primaryStage.setTitle("Text Field Example");
    primaryStage.show();
}
```



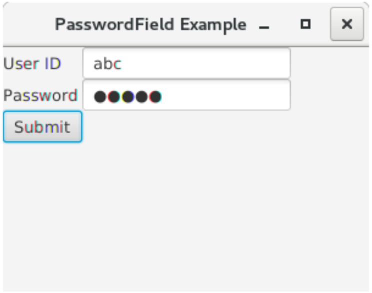
香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

JavaFX UI Controls

- PasswordField is used to get the user's password. Whatever is typed in the password field is not shown on the screen to anyone.

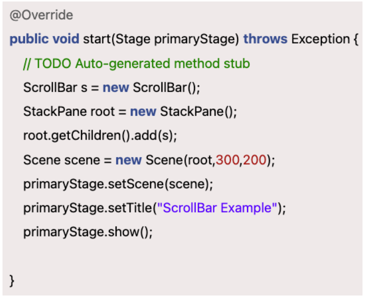


```
@Override
public void start(Stage primaryStage) throws Exception {
    // TODO Auto-generated method stub
    Label user_id=new Label("User ID");
    Label password = new Label("Password");
    TextField tf=new TextField();
    PasswordField pf=new PasswordField();
    pf.setPromptText("Enter Password");
    Button b = new Button("Submit");
    GridPane root = new GridPane();
    root.addRow(0, user_id, tf);
    root.addRow(1, password, pf);
    root.addRow(5, b);
    Scene scene=new Scene(root,300,200);
    primaryStage.setScene(scene);
    primaryStage.setTitle("PasswordField Example");
    primaryStage.show();
}
```

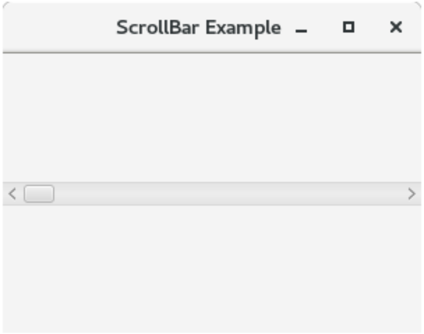


JavaFX UI Controls

- ScrollBar is used to provide a scroll bar to the user so that the user can scroll down the application pages.

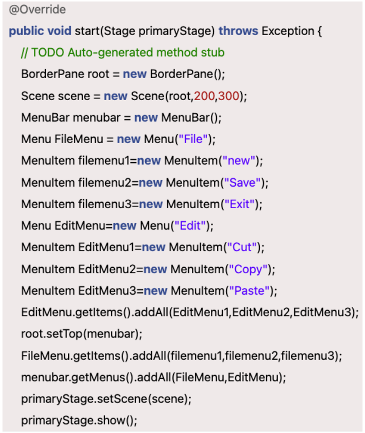


```
@Override
public void start(Stage primaryStage) throws Exception {
    // TODO Auto-generated method stub
    ScrollBar s = new ScrollBar();
    StackPane root = new StackPane();
    root.getChildren().add(s);
    Scene scene = new Scene(root,300,200);
    primaryStage.setScene(scene);
    primaryStage.setTitle("ScrollBar Example");
    primaryStage.show();

}
```

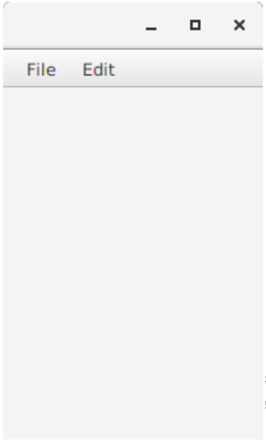


JavaFX UI Controls

- Menu implement menus. The menu is the main component of any application.

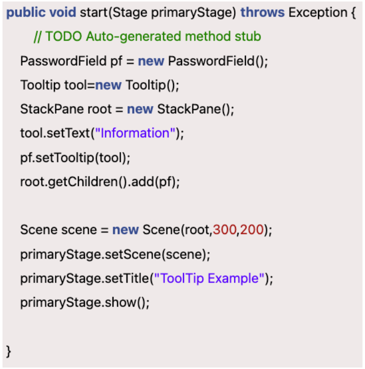


```
@Override
public void start(Stage primaryStage) throws Exception {
  // TODO Auto-generated method stub
  BorderPane root = new BorderPane();
  Scene scene = new Scene(root,200,300);
  MenuBar menubar = new MenuBar();
  Menu FileMenu = new Menu("File");
  MenuItem filemenu1=new MenuItem("new");
  MenuItem filemenu2=new MenuItem("Save");
  MenuItem filemenu3=new MenuItem("Exit");
  Menu EditMenu=new Menu("Edit");
  MenuItem EditMenu1=new MenuItem("Cut");
  MenuItem EditMenu2=new MenuItem("Copy");
  MenuItem EditMenu3=new MenuItem("Paste");
  EditMenu.getItems().addAll(EditMenu1,EditMenu2,EditMenu3);
  root.setTop(menubar);
  FileMenu.getItems().addAll(filemenu1,filemenu2,filemenu3);
  menubar.getMenus().addAll(FileMenu,EditMenu);
  primaryStage.setScene(scene);
  primaryStage.show();
```

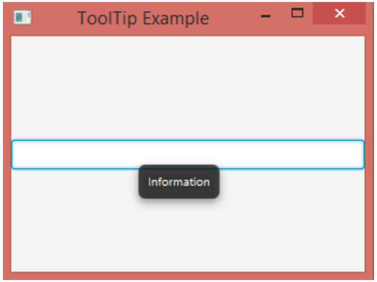


JavaFX UI Controls

- **ToolTip** is used to provide hint to the user about any component. It is used to provide hints about the text fields or password fields being used in the application.

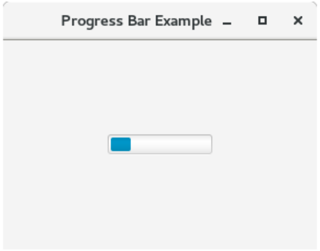


```
public void start(Stage primaryStage) throws Exception {
    // TODO Auto-generated method stub
  PasswordField pf = new PasswordField();
  Tooltip tool=new Tooltip();
  StackPane root = new StackPane();
  tool.setText("Information");
  pf.setTooltip(tool);
  root.getChildren().add(pf);

  Scene scene = new Scene(root,300,200);
  primaryStage.setScene(scene);
  primaryStage.setTitle("ToolTip Example");
  primaryStage.show();

}
```

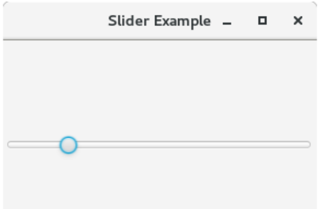


JavaFX UI Controls

- **Progress Bar** is used to show the work progress to the user.

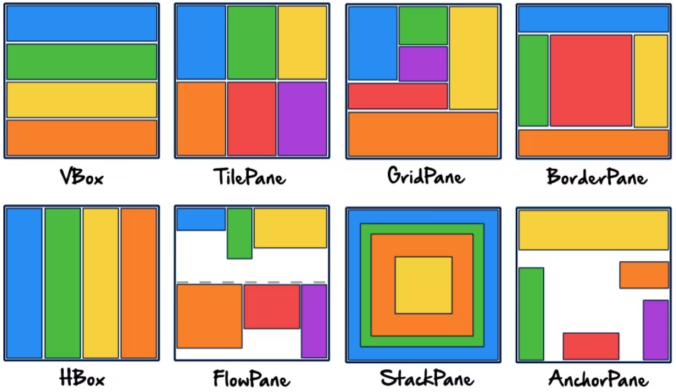


- **Slider** is where users move a slider over the range of values to select one of them.



JavaFX Layouts

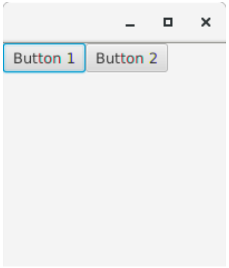
- Layouts are the top-level container classes that define the UI styles for scene graph objects. The layout can be seen as the parent node to all the other nodes.
- JavaFX provides various layout panes that support different styles of layouts.



JavaFX Layouts

- **JavaFX HBox**: HBox layout pane arranges the nodes in a single row.

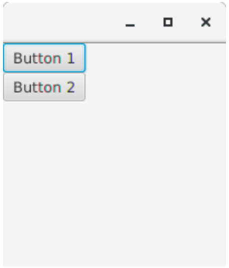
```
package application;
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.HBox;
import javafx.stage.Stage;
public class Label_Test extends Application {

@Override
public void start(Stage primaryStage) throws Exception {
Button btn1 = new Button("Button 1");
Button btn2 = new Button("Button 2");
HBox root = new HBox();
Scene scene = new Scene(root,200,200);
root.getChildren().addAll(btn1,btn2);
primaryStage.setScene(scene);
primaryStage.show();
}
public static void main(String[] args) {
      launch(args);
}
```



JavaFX Layouts

- **JavaFX VBox**: This layout Pane arranges the nodes in a single vertical column.

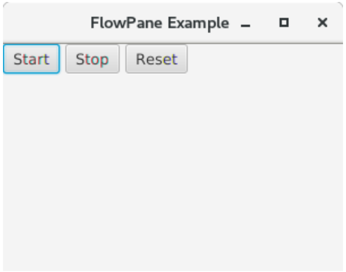
```
package application;
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;
public class Label_Test extends Application {

  @Override
  public void start(Stage primaryStage) throws Exception {
    Button btn1 = new Button("Button 1");
    Button btn2 = new Button("Button 2");
    VBox root = new VBox();
    Scene scene = new Scene(root,200,200);
    root.getChildren().addAll(btn1,btn2);
    primaryStage.setScene(scene);
    primaryStage.show();
  }
  public static void main(String[] args) {
    launch(args);
  }
```



JavaFX Layouts

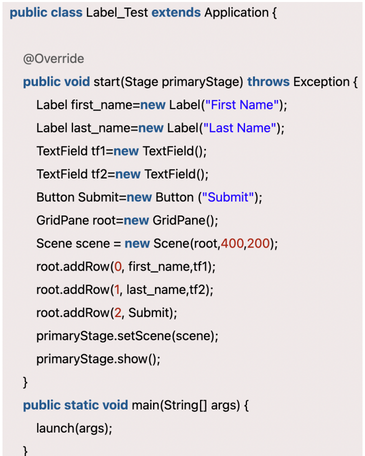
- **JavaFX FlowPane**: FlowPane layout pane organizes the nodes in a flow that are wrapped at the FlowPane's boundary.

```
public class FlowPaneTest extends Application {

  @Override
  public void start(Stage primaryStage) {
    primaryStage.setTitle("FlowPane Example");
    FlowPane root = new FlowPane();
    root.setVgap(6);
    root.setHgap(5);
    root.setPrefWrapLength(250);
    root.getChildren().add(new Button("Start"));
    root.getChildren().add(new Button("Stop"));
    root.getChildren().add(new Button("Reset"));
    Scene scene = new Scene(root,300,200);

    primaryStage.setScene(scene);
    primaryStage.show();
  }

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

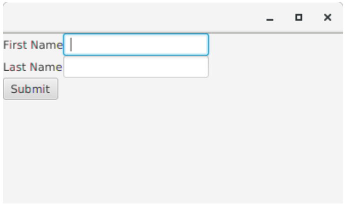


JavaFX Layouts

- **JavaFX GridPane**: This layout pane provides a flexible grid of rows and columns where nodes can be placed in any cell of the grid.

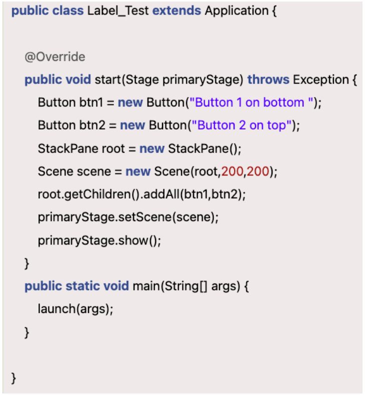


```
public class Label_Test extends Application {

  @Override
  public void start(Stage primaryStage) throws Exception {
    Label first_name=new Label("First Name");
    Label last_name=new Label("Last Name");
    TextField tf1=new TextField();
    TextField tf2=new TextField();
    Button Submit=new Button ("Submit");
    GridPane root=new GridPane();
    Scene scene = new Scene(root,400,200);
    root.addRow(0, first_name,tf1);
    root.addRow(1, last_name,tf2);
    root.addRow(2, Submit);
    primaryStage.setScene(scene);
    primaryStage.show();
  }
  public static void main(String[] args) {
    launch(args);
  }
```

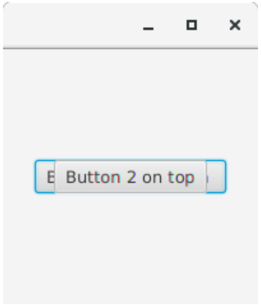


JavaFX Layouts

- **JavaFX StackPane**: This layout places all the nodes into a single stack where every new node gets placed on top of the previous node.



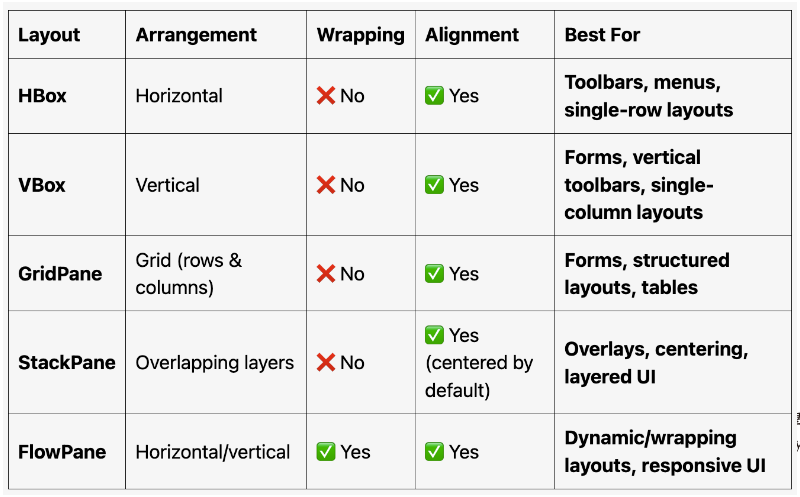
```
public class Label_Test extends Application {

  @Override
  public void start(Stage primaryStage) throws Exception {
    Button btn1 = new Button("Button 1 on bottom ");
    Button btn2 = new Button("Button 2 on top");
    StackPane root = new StackPane();
    Scene scene = new Scene(root,200,200);
    root.getChildren().addAll(btn1,btn2);
    primaryStage.setScene(scene);
    primaryStage.show();
  }
  public static void main(String[] args) {
    launch(args);
  }

}
```



香 港 中 文 大 學 (深 圳)
The Chinese University of Hong Kong, Shenzhen

JavaFX Layouts

- The Summary of JavaFx Layouts.



Layout	Arrangement	Wrapping	Alignment	Best For
HBox	Horizontal	❌ No	✅ Yes	**Toolbars, menus, single-row layouts**
VBox	Vertical	❌ No	✅ Yes	**Forms, vertical toolbars, single-column layouts**
GridPane	Grid (rows & columns)	❌ No	✅ Yes	**Forms, structured layouts, tables**
StackPane	Overlapping layers	❌ No	✅ Yes (centered by default)	**Overlays, centering, layered UI**
FlowPane	Horizontal/vertical	✅ Yes	✅ Yes	**Dynamic/wrapping layouts, responsive UI**

JavaFX Layouts

- "Can I put one layout inside / on the top of another layout ?"

Yes, you can have a hierarchy of layouts!

Check the overlapping layouts in the example code.

Question and Answering (Q&A)

香 港 中 文 大 學 (深 圳)
The Chinese University of Hong Kong, Shenzhen