BIRZEIT UNIVERSITY

Advanced JavaFX

By: Mamoun Nawahdah (Ph.D.)

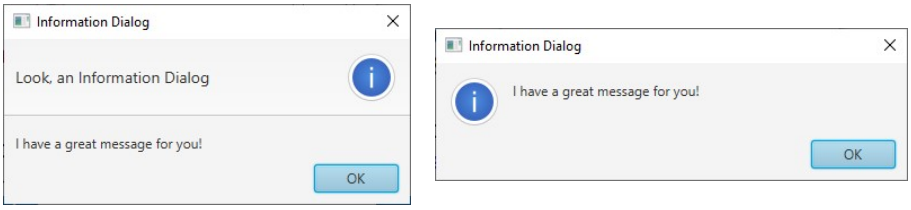
2022/2023

Objectives

- To create **Dialogs** and **Alerts**
- To create a **FileChooser**
- To create tab panes using the **TabPane** control
- To create and display tables using the **TableView** and **TableColumn** classes

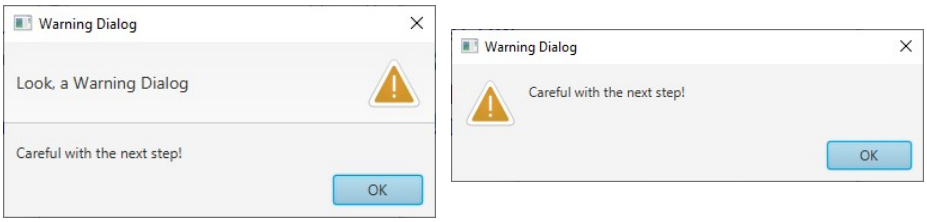
Information Dialog

Alert alert = **new Alert(AlertType.*INFORMATION);***
alert.setTitle("Information Dialog");
alert.setHeaderText("Look, an Information Dialog");
alert.setContentText("I have a great message for you!");
alert.showAndWait();



Warning Dialog

Alert alert = **new Alert(AlertType.*WARNING);***
alert.setTitle("Warning Dialog");
alert.setHeaderText("Look, a Warning Dialog");
alert.setContentText("Careful with the next step!");
alert.showAndWait();

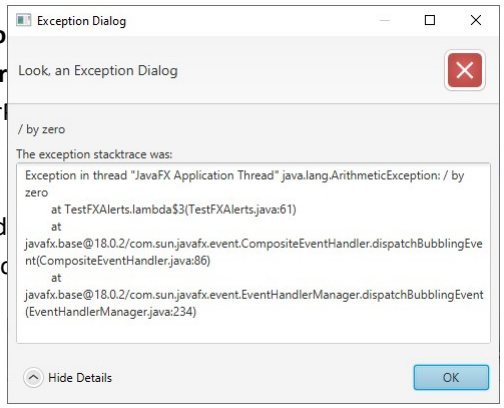


Error Dialog

```
Alert alert = new Alert(AlertType.ERROR);
alert.setTitle("Error Dialog");
alert.setHeaderText(null);
alert.setContentText("Ooops, there was an error!");
alert.showAndWait();
```

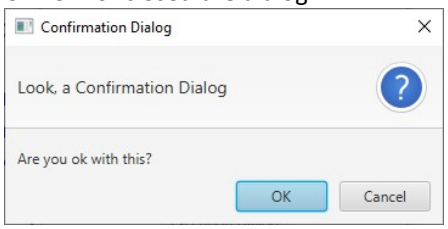
Exception Dialog

```
Alert alert = new Alert(AlertType.ERROR);
alert.setTitle("Exception Dialog");
alert.setHeaderText("Look, an Exception Dialog");
alert.setContentText("/ by zero");

Label label = new Label("The exceptio
TextArea textArea = new TextArea("Ar
BorderPane expContent = new Border
expContent.setTop(label);
expContent.setCenter(textArea);
// Set expandable Exception into the d
alert.getDialogPane().setExpandableCo
alert.showAndWait();
```



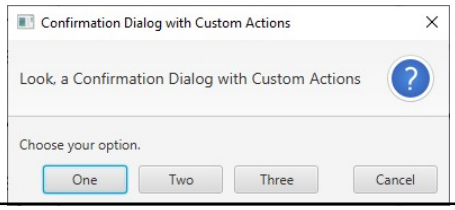
Confirmation Dialog

```
Alert alert = new Alert(AlertType.CONFIRMATION);
alert.setTitle("Confirmation Dialog");
alert.setHeaderText("Look, a Confirmation Dialog");
alert.setContentText("Are you ok with this?");

Optional<ButtonType> result = alert.showAndWait();
if (result.get() == ButtonType.OK){
   // ... user chose OK
} else {
   // ... user chose CANCEL or closed the dialog
}
```



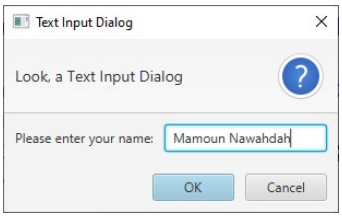
Confirmation Dialog with Custom Actions

```
Alert alert = new Alert(AlertType.CONFIRMATION);
alert.setTitle("Confirmation Dialog with Custom Actions");
alert.setHeaderText("Look, a Confirmation Dialog with Custom Actions");
alert.setContentText("Choose your option.");
ButtonType oneBT = new ButtonType("One");
ButtonType twoBT = new ButtonType("Two");
ButtonType threeBT = new ButtonType("Three");
ButtonType cancelBT = new ButtonType("Cancel", ButtonData.CANCEL_CLOSE);
alert.getButtonTypes().setAll(oneBT, twoBT, threeBT, cancelBT);
Optional<ButtonType> result = alert.showAndWait();
if (result.get() == oneBT){   // ... user chose "One"
} else if (result.get() == twoBT) {   // ... user chose "Two"
} else if (result.get() == threeBT) {   // ... user chose "Three"
} else {
   // ... user chose CANCEL or closed the dialog
}
```



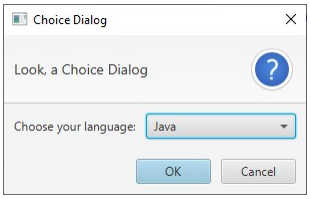
Text Input Dialog

```
TextInputDialog dialog = new TextInputDialog("walter");
dialog.setTitle("Text Input Dialog");
dialog.setHeaderText("Look, a Text Input Dialog");
dialog.setContentText("Please enter your name:");

Optional<String> result = dialog.showAndWait();
if (result.isPresent()){
   System.out.println("Your name: " + result.get());
}
```



Choice Dialog

```
List<String> choices = new ArrayList<>();
choices.add("Java"); choices.add("C"); choices.add("HTML");

ChoiceDialog<String> dialog = new ChoiceDialog<>("Java", choices);
dialog.setTitle("Choice Dialog");
dialog.setHeaderText("Look, a Choice Dialog");
dialog.setContentText("Choose your language:");

Optional<String> result = dialog.showAndWait();
if (result.isPresent()){
   System.out.println("Your choice: " + result.get());
}
```



FileChooser

- **FileChooser** is used to invoke
  - *file open dialogs* for selecting a single file (**showOpenDialog**)
  - file *open dialogs for selecting multiple files* (**showOpenMultipleDialog**)
  - *file save dialogs* (**showSaveDialog**)

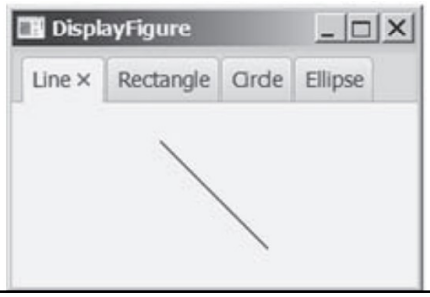
Creates a new FileChooser dialog

```
FileChooser fileChooser = new FileChooser();
File file = fileChooser.showOpenDialog(stage);
```

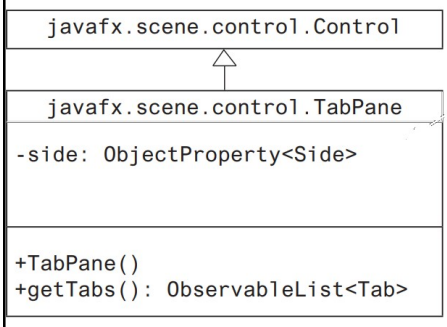
Method	Explanation
getInitialDirectory()	Returns the initial directory of the file chooser.
getTitle()	Returns the title of file chooser.
setInitialDirectory(File f)	Sets the initial directory of the filechooser.
setTitle(String t)	Sets the title of the file chooser.
showOpenDialog(Window w)	Shows a new open file selection dialog.
setInitialFileName(String n)	Sets the initial file name of file chooser.
showSaveDialog(Window w)	Shows a new Save file selection dialog.
getInitialFileName()	Returns the initial filename of the file chooser.

TabPane

- **TabPane** is a useful control that provides a set of mutually exclusive tabs
- You can switch between a group of tabs.
- Only one tab is visible at a time.
- A **Tab** can be added to a **TabPane**.
- Tabs in a TabPane can be placed in the position **top**, **left**, **bottom**, or **right**.



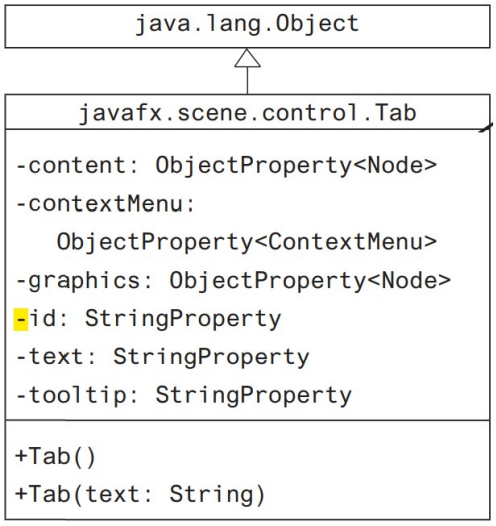
TabPane displays and manages the tabs



The position of the tab in the tab pane. Possible values are `Side.TOP`, `Side.BOTTOM`, `Side.LEFT`, and `Side.RIGHT` (default: `Side.TOP`).

Creates a default `Tab Pane`.
Returns a list of tabs in this `Tab Pane`.

Tab contains a node

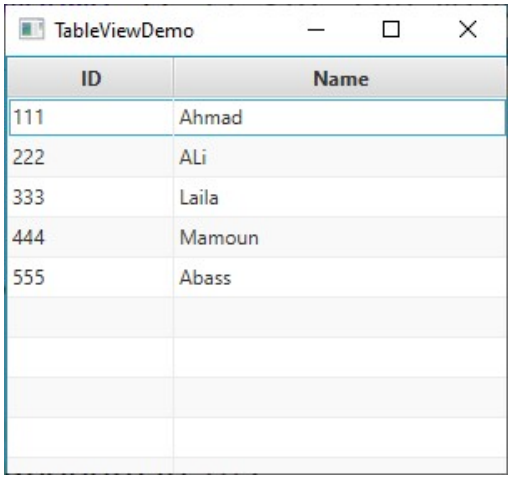


The content associated with the tab.
The context menu associated with the tab.
The graphics in the tab.
The id for the tab.
The text shown in the tab.
The tooltip associated with the tab.
Constructs a default tab.
Constructs a tab with the specified string.

```
public void start(Stage primaryStage) {
    TabPane tabPane = new TabPane();
     Tab tab1 = new Tab("Line");
     StackPane pane1 = new StackPane();
     pane1.getChildren().add(new Line(10, 10, 80, 80));
     tab1.setContent(pane1);
     Tab tab2 = new Tab("Rectangle");
     tab2.setContent(new Rectangle(10, 10, 200, 200));
     Tab tab3 = new Tab("Circle");
     tab3.setContent(new Circle(50, 50, 20));
     Tab tab4 = new Tab("Ellipse");
     tab4.setContent(new Ellipse(10, 10, 100, 80));
     tabPane.getTabs().addAll(tab1, tab2, tab3, tab4);

     Scene scene = new Scene(tabPane, 300, 250);
     primaryStage.setTitle("DisplayFigure");
     primaryStage.setScene(scene);
     primaryStage.show();
}
```

TableView

- **TableView** is a control that displays data in rows and columns in a two-dimensional grid



- **TableView**, **TableColumn**, and **TableCell** are used to display and manipulate a table.
  - **TableView** displays a table.
  - **TableColumn** defines the columns in a table.
  - **TableCell** represents a cell in the table.

- Creating a **TableView** is a multistep process.
  - First, you need to create an instance of **TableView** and associate data with the TableView.
  - Second, you need to create columns using the **TableColumn** class and set a column cell value factory to specify how to populate all cells within a single TableColumn.