P4. MENUS, TOOLBARS, AND DIALOGS IN JAVAFX

Interfaces Persona Computador

Depto. Sistemas Informáticos y Computación

UPV

Summary

- Menus
- Toolbars
- JavaFX 8 dialogs
- Modalities
- Internationalization
- Exercise
- Bibliography

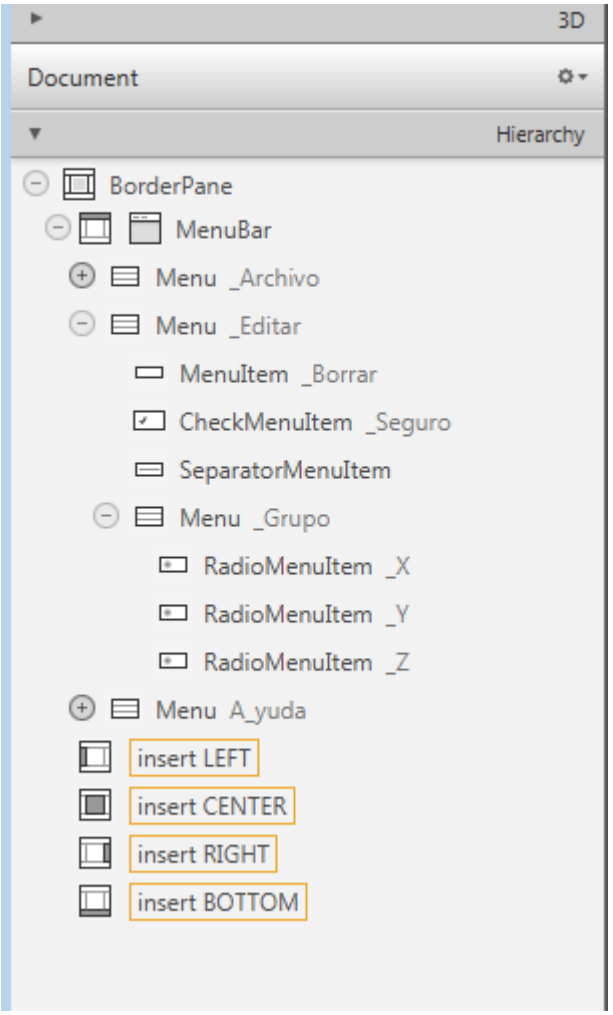
Menus

- An application's menu presents the user with a set of available actions
- Menus are hierarchically organized in high level menus (File, Edit, Help, etc.) and low level menus
  - A menu can have submenus, which in turn can have submenus…
- Menus support elements that can behave like *checkboxes* and *radiobuttons*.
- Menu items can be assigned shortcuts. Then, the user will be able to select them using the keyboard

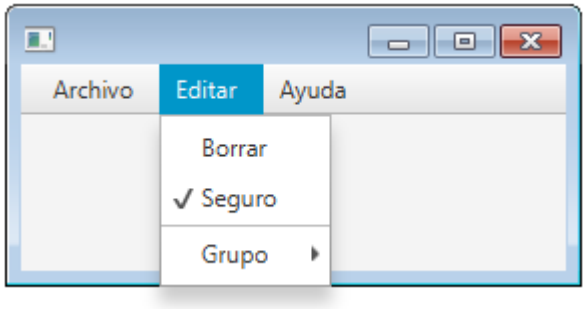
Menus

- Menu related classes:
  - `MenuBar`: a menu bar that contains an entry for each high-level menu
  - `Menu`: a high-level menu that can contain menu items and other menus (submenus that contain more menu items, etc.). They are organized in a tree-like structure
  - `MenuItem`: a leaf node in the menu hierarchy, it performs a given action
  - `SeparatorMenuItem`: a separator
  - `CheckMenuItem`: similar to a `CheckBox`, it can be selected or unselected
  - `RadioMenuItem`: like a `CheckMenuItem`, but only one of the `RadioMenuItems` that share a `ToggleGroup` can be selected at any time

Building the menu with SceneBuilder

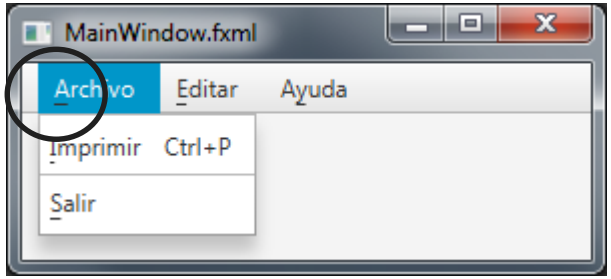


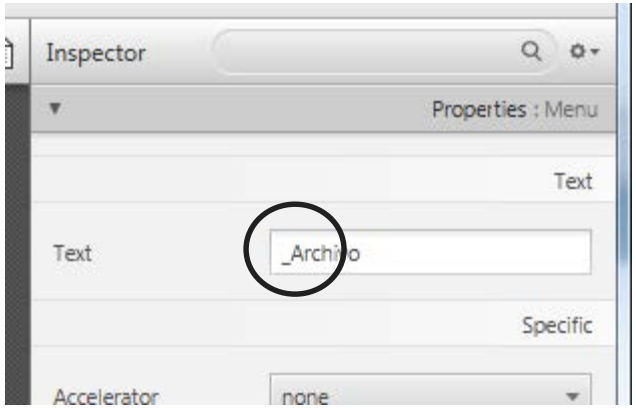
- We usually put the menu bar in the top side of a `BorderPane`
- Then, we drag to the `Hierarchy` panel the menu items we want to use
- We can change the name of an item by double clicking on its name in the panel
- We can add menus inside other menus

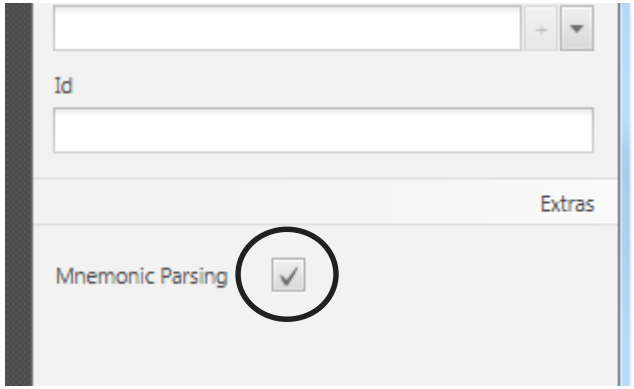


Keyboard Shortcuts

- A menu item can be activated by selecting it with the mouse, but also by using:
  - An access key

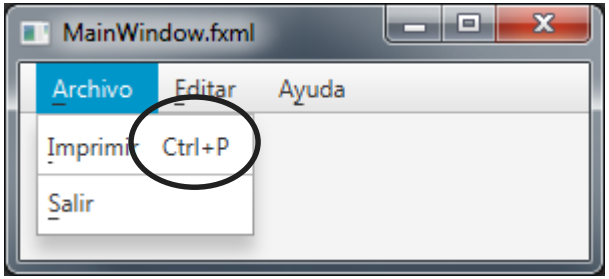


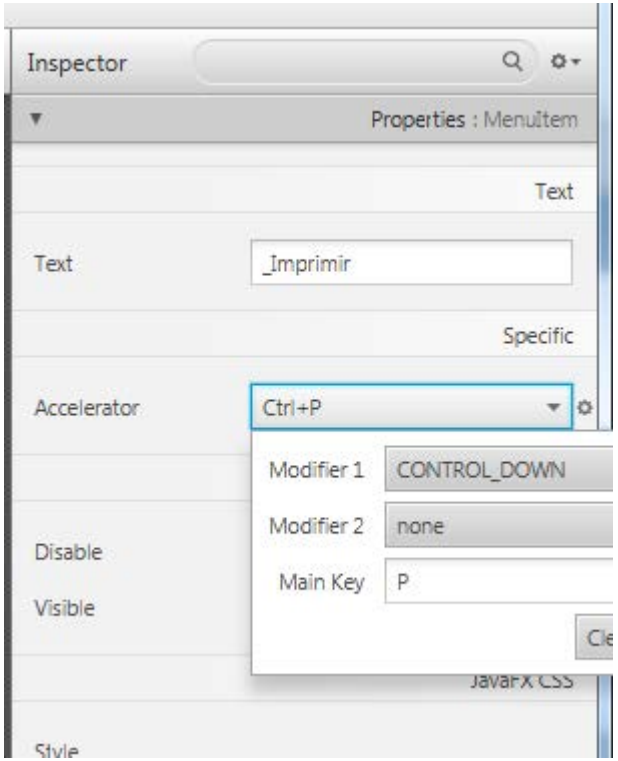




Keyboard Shortcuts

- A menu item can be activated by selecting it with the mouse, but also by using:
  - A shortcut or accelerator





The SHORTCUT modifier represents the Ctrl key in Windows and the Meta key in Mac.

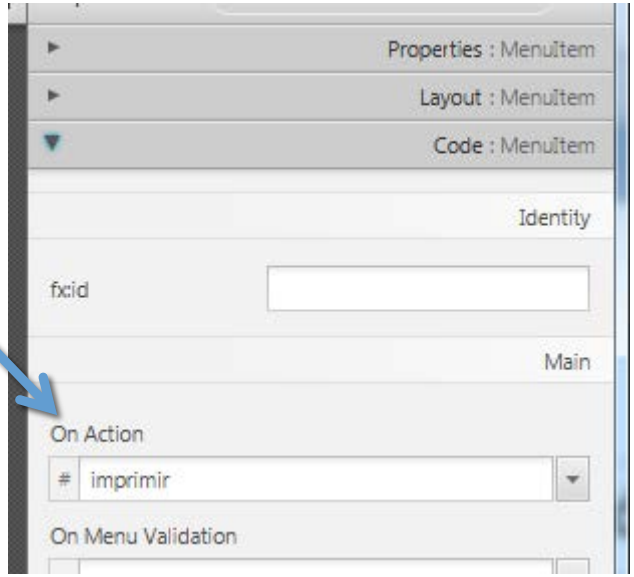
Handling Menu Events

- Menu items behave like regular buttons, so you only need to assign them a method that receives an `ActionEvent`:

```
@FXML
private void imprimir(ActionEvent e) {
  System.out.println("Print");
}
```

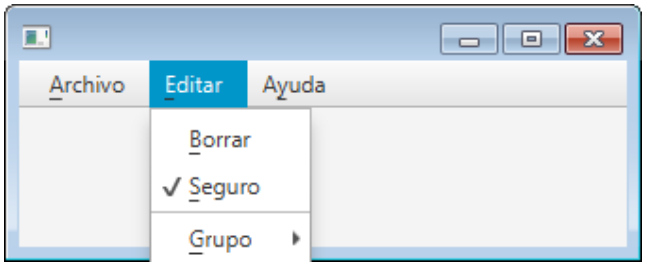
or

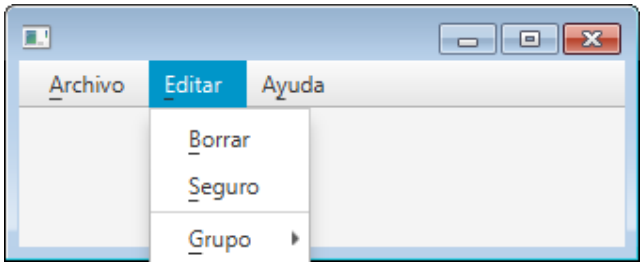
```
@FXML private MenuItem menuDelete;
@FXML  void initialize() {
   menuDelete.setOnAction(this::delete);
}
private void delete(ActionEvent e) {
   System.out.println("Delete");
}
```



CheckMenuItem

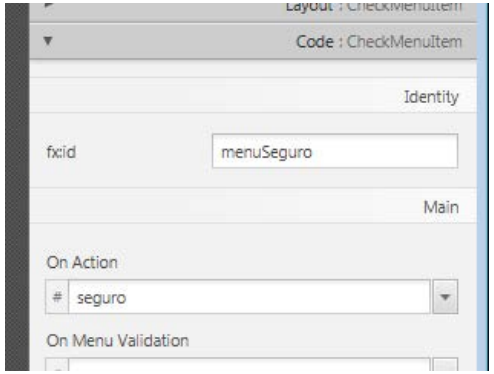
- It is a combination of a `MenuItem` and a `CheckBox`





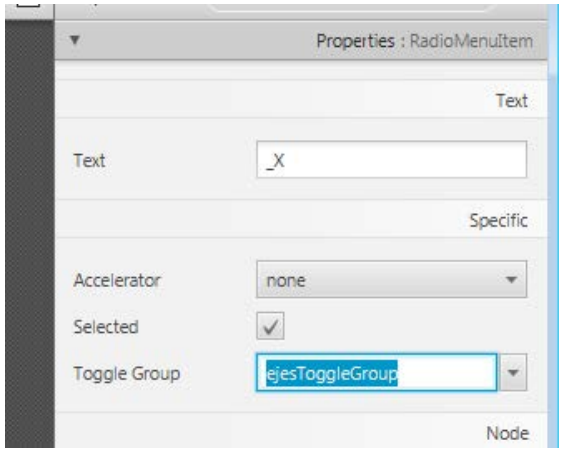
- When a menu item is selected, its associated handler is executed and its state changes between selected and unselected

```
@FXML private CheckMenuItem menuSeguro;
@FXML private void seguro(ActionEvent e) {
   System.out.println( "Are you sure: " +
        (menuSeguro.isSelected() ? "YES" : "NO"));
}
```

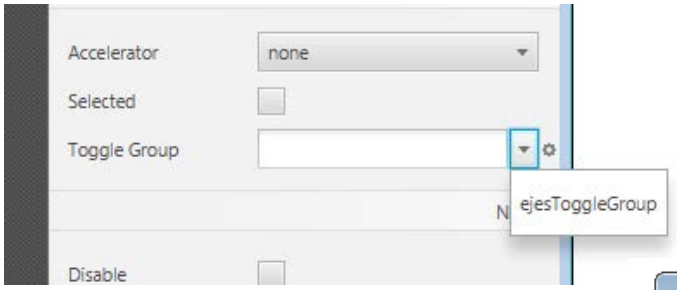


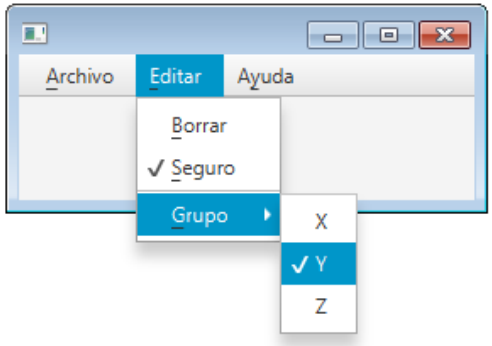
RadioMenuItem

- Behaves like a `CheckMenuItem`; however, only one `RadioMenuItem` of the same `ToggleGroup` can be selected at any time



1. Assign a name to the *toggle group* of the first *radio button*

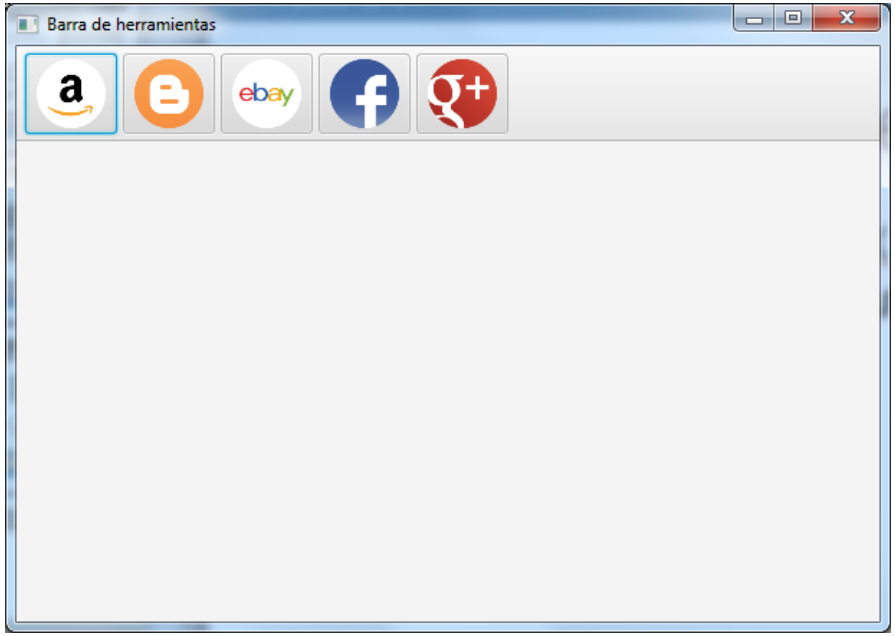




2. Select the same name from the list for all the other *radio buttons* of the group

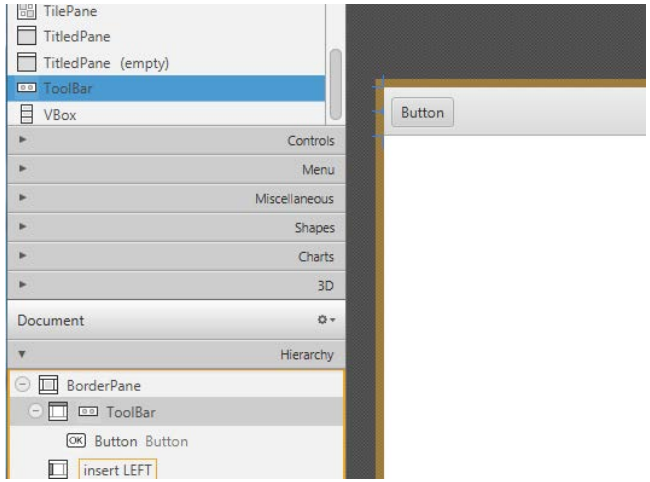
Toolbars

- The `ToolBar` class of JavaFX implements a container of buttons that can be used to implement a tool bar



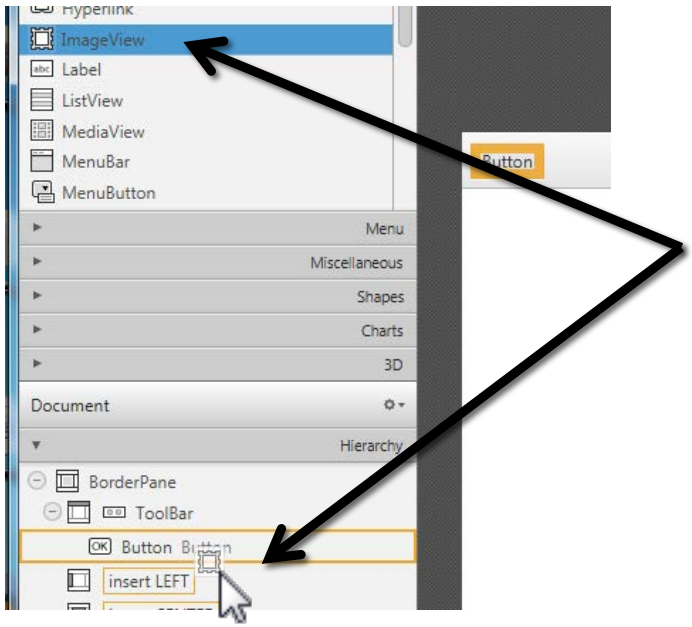
Toolbars

- A tool bar can contain any type of node, but it is normally used to contain buttons

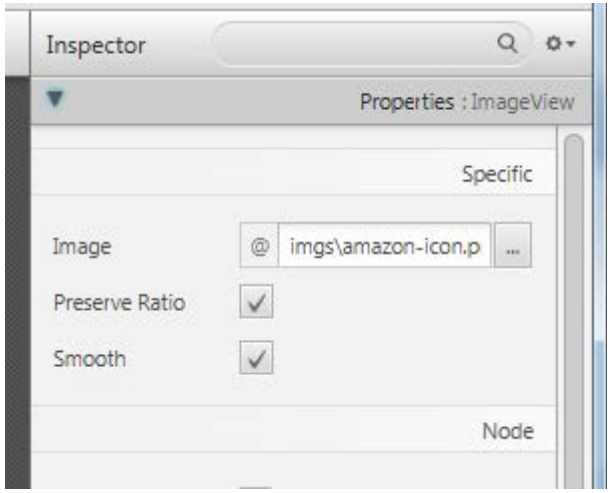


Toolbars

- A button in a toolbar normally shows a text label, but it can also show an image



1. Drag an `ImageView` to the button



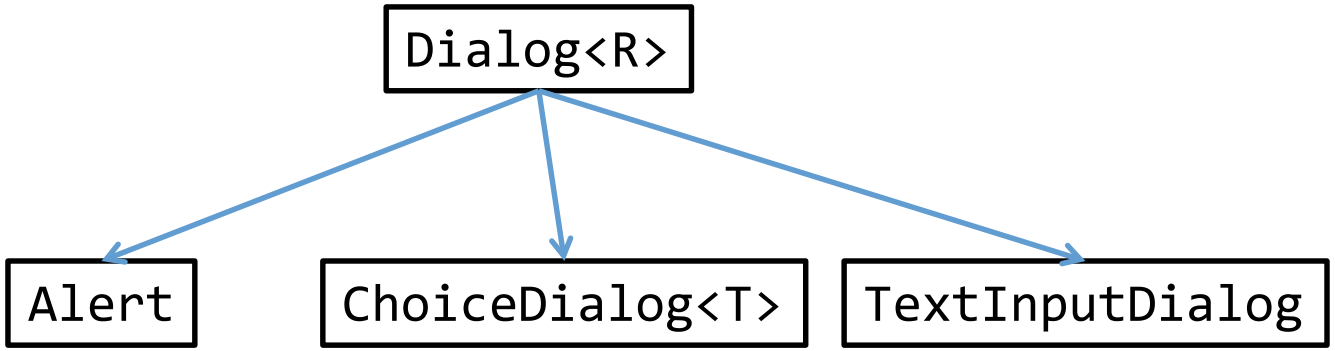
2. Assign the image to the `ImageView`. ¡Be careful! The image has to be in the `src` directory

Dialogs

- A dialog is a window that opens up during execution to ask the user for information
  - Modal dialogs: the user cannot interact with the application while the dialog is open (e.g. the Print dialog)
  - Non modal dialogs: the user can interact with the dialog or the application, while the dialog is open (e.g. the Find dialog)

Related Classes

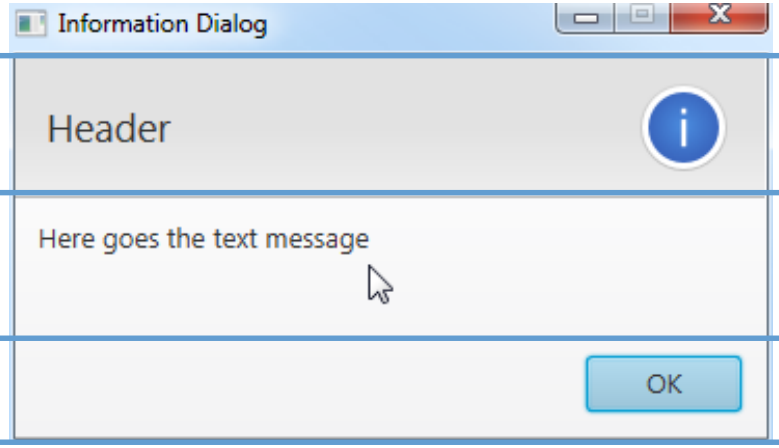
- Package: `javafx.scene.control`



Notification dialog boxes

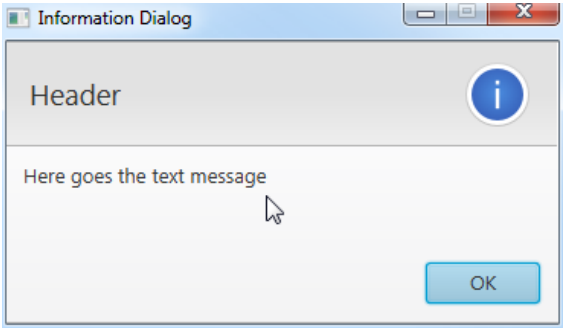
Dialog boxes for data input (an element of a list, and a text string, respectively)

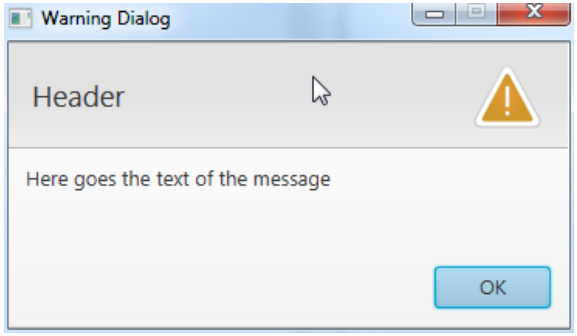
Structure of a Dialog

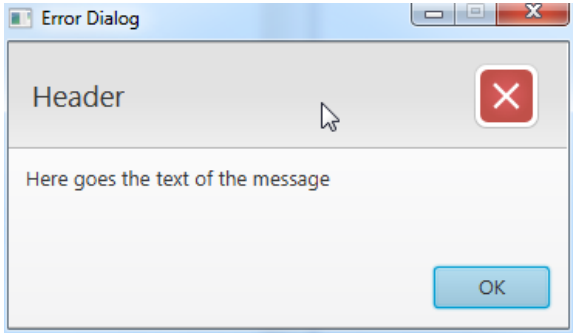
	Title
	Header (*masthead*) in boldface with an icon
	Content
	Buttons

Standard Dialogs

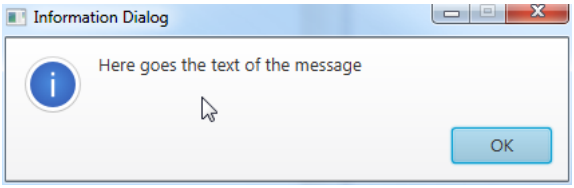
- With header

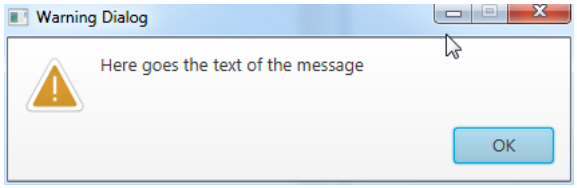


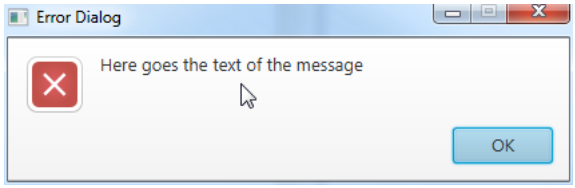




- Without header







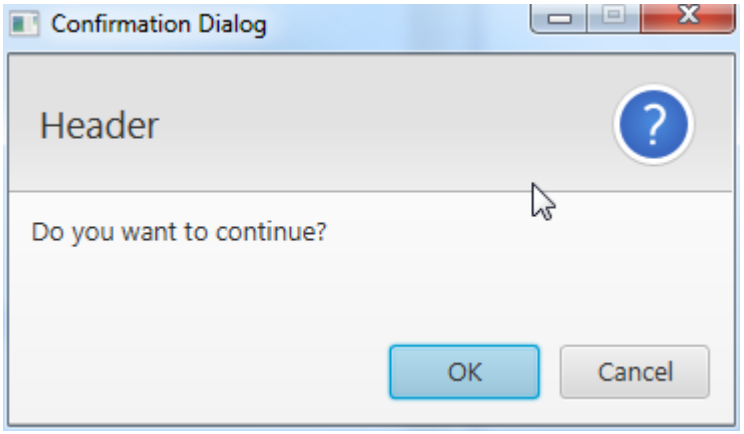
Standard Dialogs

- To create and show the dialog

```
Alert alert = new Alert(AlertType.INFORMATION);
      // or AlertType.WARNING or AlertType.ERROR or AlertType.CONFIRMATION
alert.setTitle("Information Dialog");
alert.setHeaderText("Header");
      // or null if we do not want a header
alert.setContentText("Here goes the text of the message");
alert.showAndWait();

// Also
Alert alert = new Alert(AlertType.INFORMATION, "Content");
```

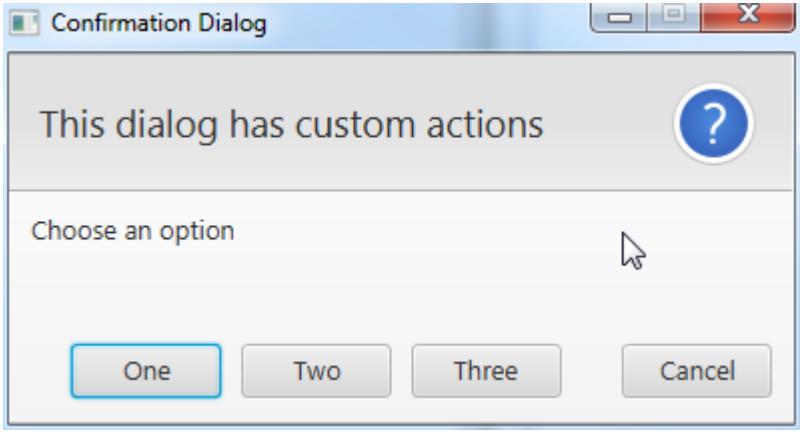
Confirmation Dialog

```
Alert alert = new Alert(AlertType.CONFIRMATION);
alert.setTitle("Confirmation Dialog");
alert.setHeaderText("Header");
alert.setContentText("Do you want to continue?");

Optional<ButtonType> result = alert.showAndWait();
if (result.isPresent() && result.get() == ButtonType.OK){
    System.out.println("OK");
} else {
    System.out.println("CANCEL");
}
```



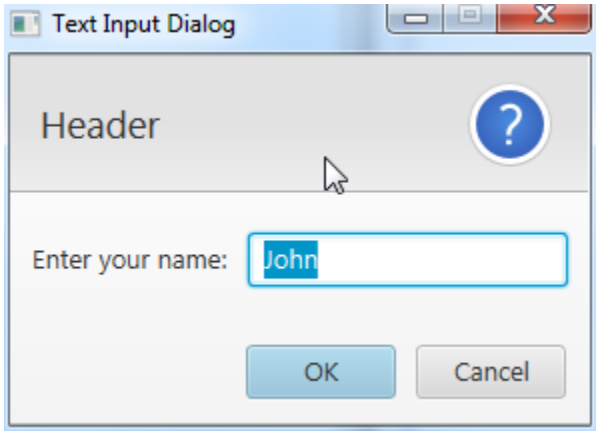
Confirmation Dialog with Custom Actions

```
Alert alert = new Alert(AlertType.CONFIRMATION);
alert.setTitle("Confirmation Dialog");
alert.setHeaderText("This dialog has custom actions");
alert.setContentText("Choose an option");

ButtonType buttonTypeOne = new ButtonType("One");
ButtonType buttonTypeTwo = new ButtonType("Two");
ButtonType buttonTypeThree = new ButtonType("Three");
ButtonType buttonTypeCancel = new ButtonType("Cancel", ButtonData.CANCEL_CLOSE);

alert.getButtonTypes().setAll(buttonTypeOne, buttonTypeTwo, buttonTypeThree, buttonTypeCancel);

Optional<ButtonType> result = alert.showAndWait();
if (result.isPresent()) {
  if (result.get() == buttonTypeOne)
    System.out.println("One");
  else if (result.get() == buttonTypeTwo)
    System.out.println("Two");
  else if (result.get() == buttonTypeThree)
    System.out.println("Three");
  else
    System.out.println("Cancel");
}
```

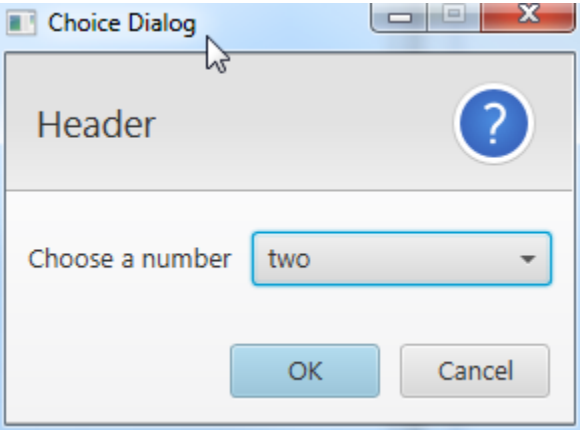


Text Input Dialog

```
TextInputDialog dialog = new TextInputDialog("John"); // Default value
dialog.setTitle("Text Input Dialog");
dialog.setHeaderText("Header");
dialog.setContentText("Enter your name:");

Optional<String> result = dialog.showAndWait();

// Obtain the result (before Java 8)
if (result.isPresent()){
        System.out.println("Hello " + result.get());
}

// Obtain the result with a lambda expression (Java 8 and later)
result.ifPresent(name -> System.out.println("Hello " + name));
```

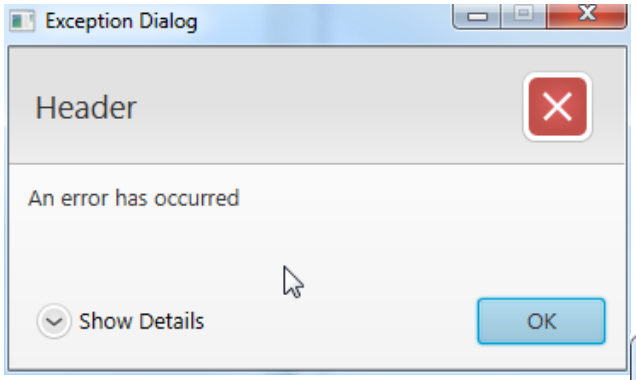


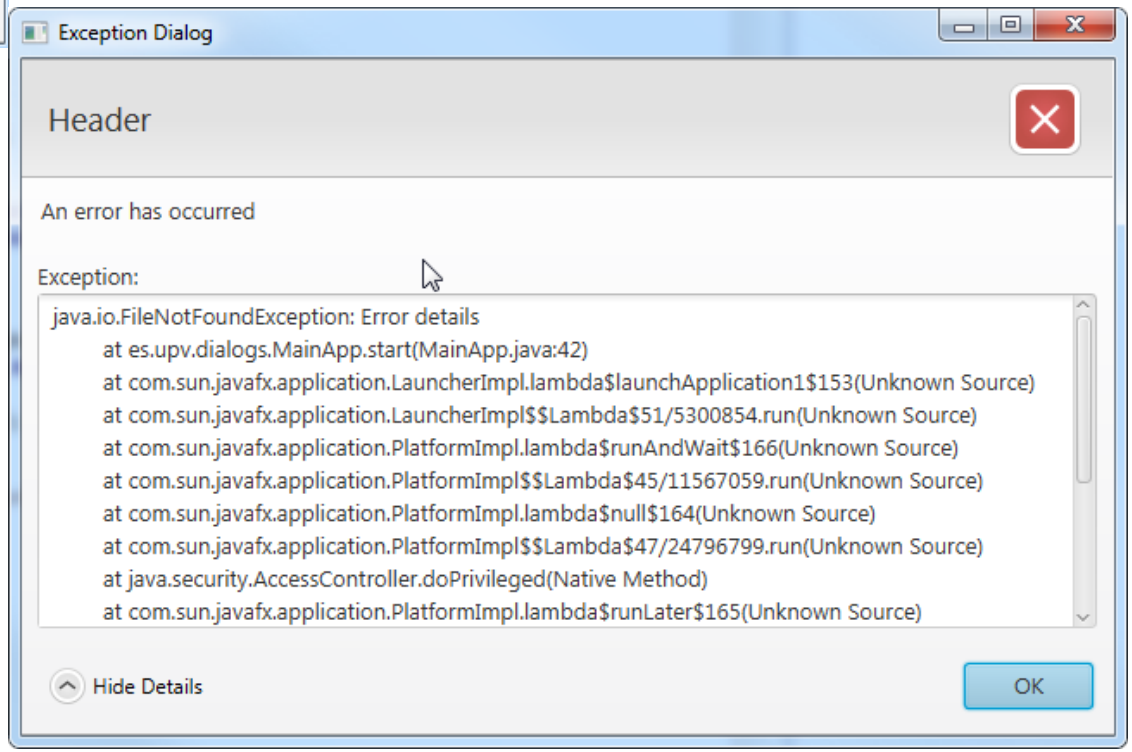
Choice Dialog




```
List<String> choices = new ArrayList<>();
choices.add("one");
choices.add("two");
choices.add("three");

ChoiceDialog<String> dialog = new ChoiceDialog<>("two", choices);
dialog.setTitle("Choice Dialog");
dialog.setHeaderText("Header");
dialog.setContentText("Choose a number");

Optional<String> result = dialog.showAndWait();
// Before Java 8
if (result.isPresent()) {
  System.out.println("Your choice: " + result.get());
}
// Getting the result with a lambda
result.ifPresent(number-> System.out.println("Your choice: " + number));
```

Error Dialog



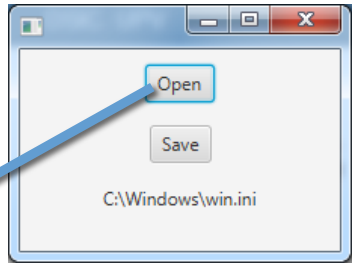


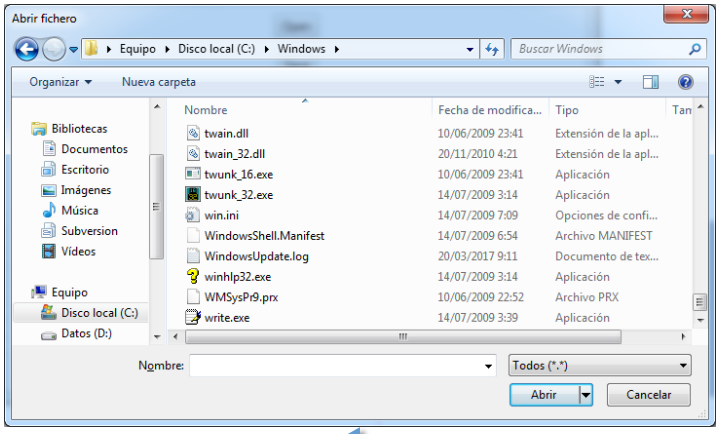
Error Dialog

```
Alert alert = new Alert(AlertType.ERROR);
alert.setTitle("Exception Dialog");
alert.setHeaderText("Header");
alert.setContentText("An error has occurred");

Exception ex = new FileNotFoundException("Error details");

StringWriter sw = new StringWriter();
PrintWriter pw = new PrintWriter(sw);
ex.printStackTrace(pw);
String exceptionText = sw.toString();

Label label =
  new Label("Exception:");

TextArea textArea =
  new TextArea(exceptionText);
textArea.setEditable(false);
textArea.setWrapText(true);
```

```
textArea.setMaxWidth(Double.MAX_VALUE);
textArea.setMaxHeight(Double.MAX_VALUE);
GridPane.setVgrow(textArea,
     Priority.ALWAYS);
GridPane.setHgrow(textArea,
     Priority.ALWAYS);

GridPane expContent = new GridPane();
expContent.setMaxWidth(Double.MAX_VALUE);
expContent.add(label, 0, 0);
expContent.add(textArea, 0, 1);

alert.getDialogPane().
     setExpandableContent(expContent);

alert.showAndWait();
```

Open/Save File Dialog






```
@FXML
private void onOpen(ActionEvent event) {
  FileChooser fileChooser = new FileChooser();
  fileChooser.setTitle("Open resource");
  fileChooser.getExtensionFilters().addAll(
    new ExtensionFilter("Text files", "*.txt"),
    new ExtensionFilter("Images", "*.png", "*.jpg", "*.gif"),
    new ExtensionFilter("Sounds", "*.wav", "*.mp3", "*.aac"),
    new ExtensionFilter("All", "*.*"));
  File selectedFile = fileChooser.showOpenDialog(
        ((Node)event.getSource()).getScene().getWindow());
  if (selectedFile != null) {
    label.setText(selectedFile.getAbsolutePath());
  }
}
```

showOpenDialog has a parameter: the dialog's parent window (*stage*). If it is not null, the dialog will be modal with respect to the window. This code shows how to get the *stage* in which a node is located.

Open/Save File Dialog

- Other methods of `FileChooser`:
  - `File showSaveDialog(Window ownerWindow)`
    - Opens a dialog for saving a file
  - `List<File> showOpenMultipleDialog(Window ownerWindow)`
    - Opens a dialog for opening multiple files
  - `final void setInitialFileName(String value)`
    - For saving a file, the default name of the new file
  - `final void setInitialDirectory(File value)`
    - Directory shown when the dialog opens

Modality

- By default JavaFX dialogs are modal, that is, they don't allow interaction with the rest of the application
  - You can change the modality of a dialog with the method `dialog.initModality(modality)`

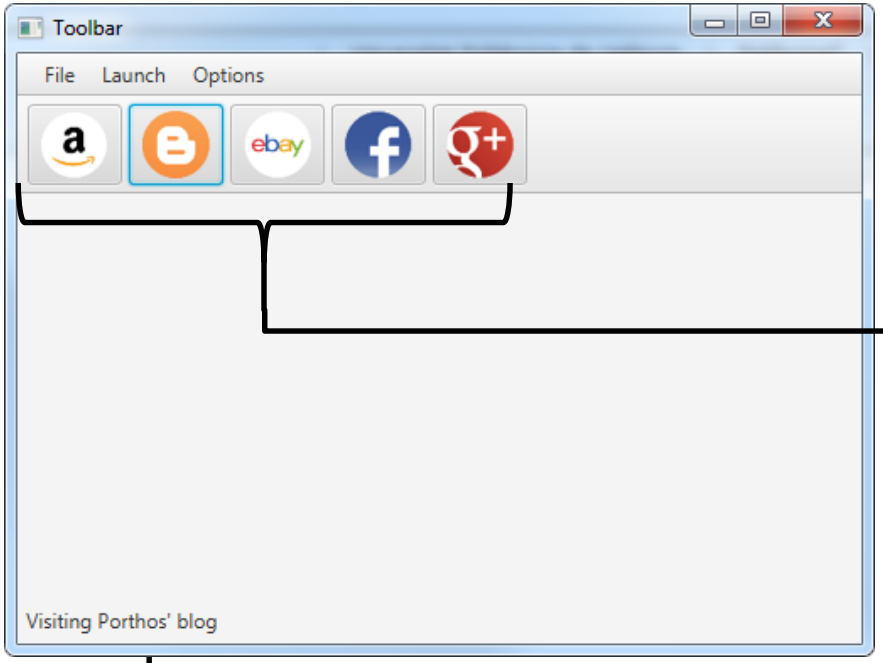
    where:
    - `modality: Modality.NONE, Modality.WINDOW_MODAL, ó Modality.APPLICATION_MODAL`
- Modality aside, you can also specify whether opening a dialog blocks the execution or not
  - Blocking: `showAndWait()`
  - Non-blocking: `show()`

Other Options

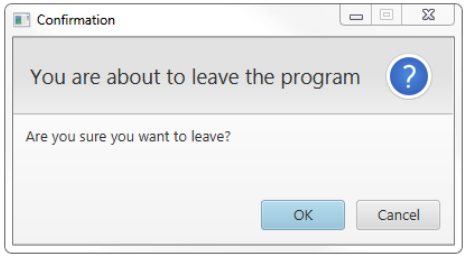
- You can also establish a dialog's parent
  - `dialog.initOwner(parentWindow);`
  - If you don't establish a parent or set it to `null`, the dialog's window does not depend on another window, it's a top-level, unowned dialog

Exercise

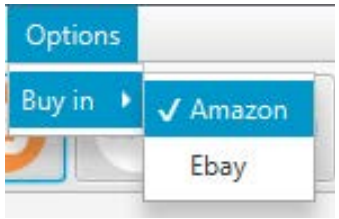
- Implement the following application




```
File
    Exit
Launch
    Amazon
    Blogger
    Ebay
    Facebook
    Google+
Options
    Buy in
        Amazon
        Ebay
```



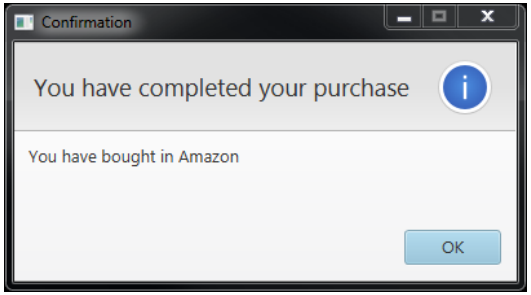
Status bar (a Label)

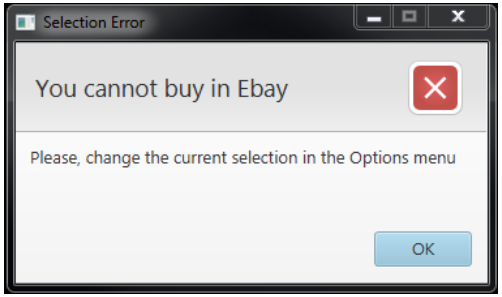


RadioMenuItem

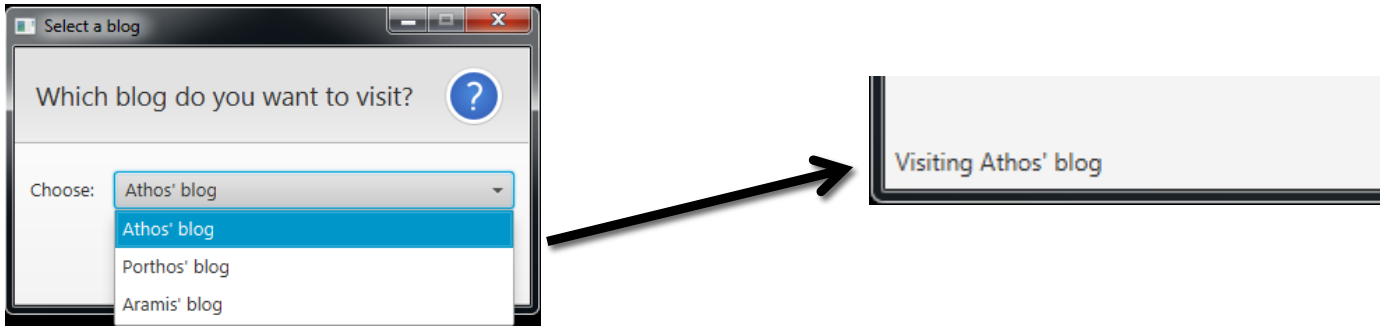
Exercise

- If the user clicks Amazon or Ebay, the application will check whether the same option is marked in the Options menu and it will display a confirmation message or an error message



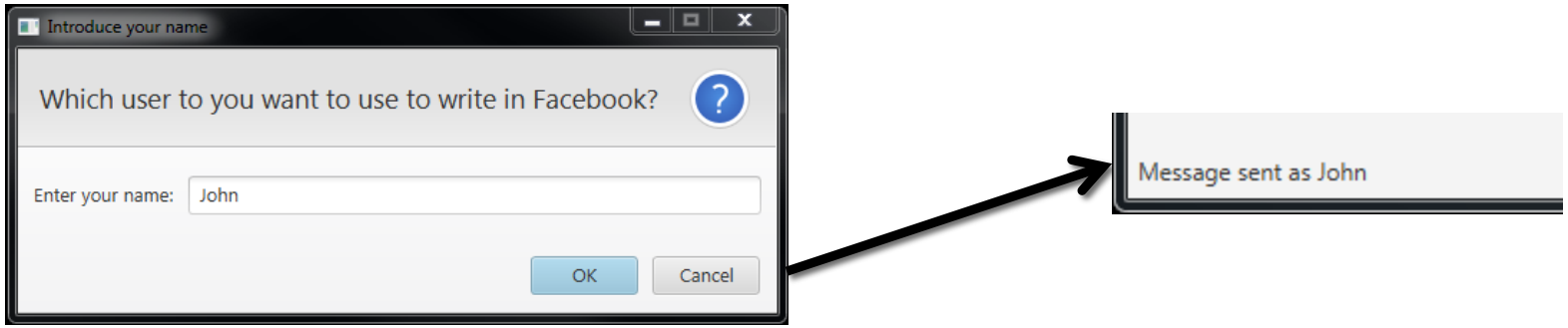


- If the user selects Blogger, the application will ask which blog he wants to visit and will show the owner of the blog in the status bar:



Exercise

- If the user selects Facebook or Google+, the application will ask for a user name to be used to send a message and it will display it in the status bar



- Extension: translate the application to a different language to your system's and load its Locale by hand, for showing the strings in your application in that other language

Bibliography

- Dialogs: http://code.makery.ch/blog/javafx-dialogs-official
  - https://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/Dialog.html
  - https://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/Alert.html
  - https://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/TextInputDialog.html
  - https://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/ChoiceDialog.html
  - https://docs.oracle.com/javase/8/javafx/api/javafx/stage/FileChooser.html
- Internationalization
  - http://docs.oracle.com/javafx/scenebuilder/1/user_guide/i18n-support.htm
  - https://docs.oracle.com/javase/8/docs/api/java/util/Locale.html
- C. Dea et al. JavaFX 8. Introduction by Example. Apress, 2014