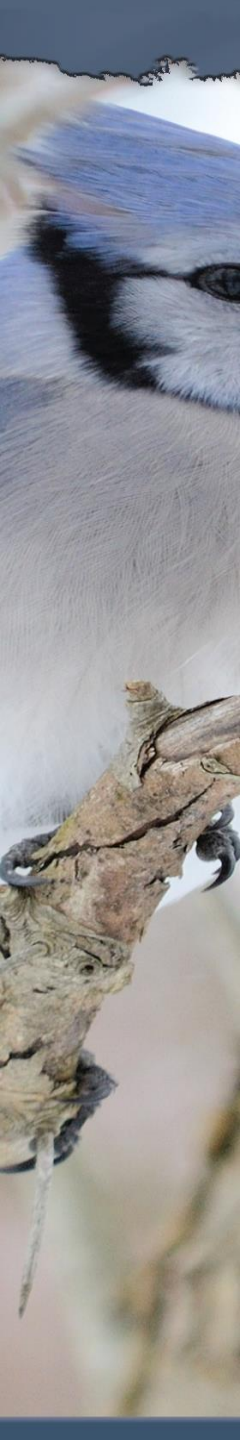
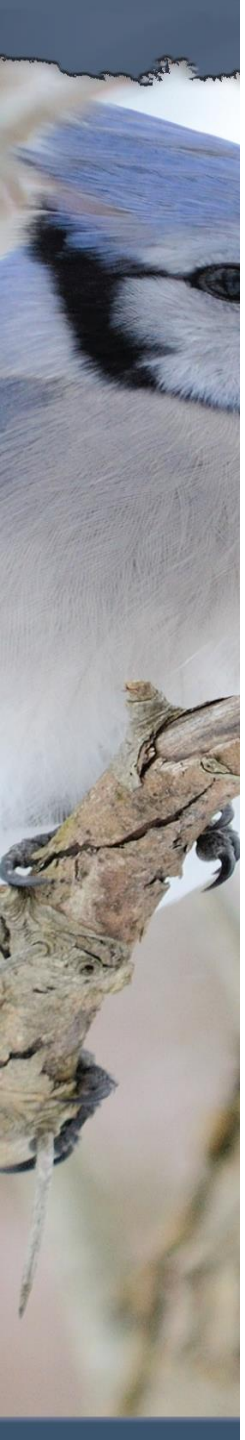
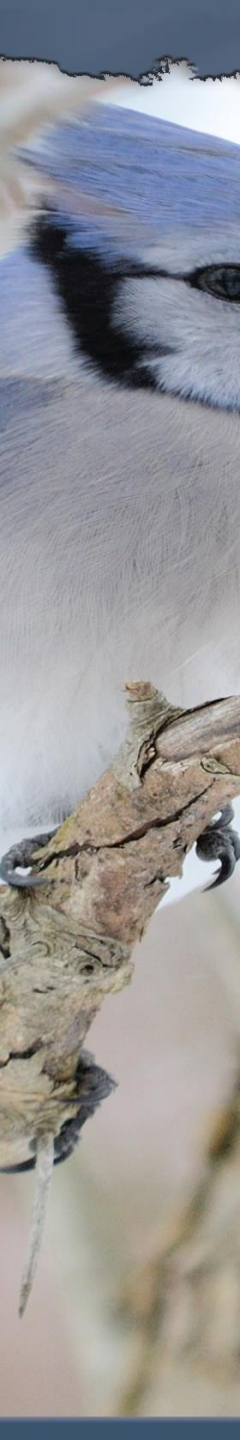


Building Graphical User Interfaces



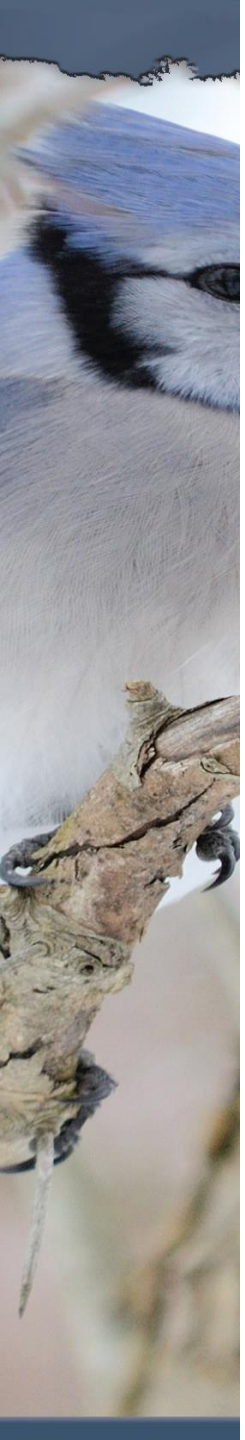
Overview

- Constructing GUIs
- GUI layout
- Event handling



History of GUI in Java

- First there was AWT – Abstract Windowing Toolkit
  - The smallest common elements accross platforms (button, textfield, checkbok, frame etc.)
  - Very limiting
  - Used the GUI-elements in the operating system
- Then there was Swing
  - Uses AWT core functionallity
  - All visible components are drawn in Java – hence support for all sorts of GUI components, even custom-made.
  - Supports multiple ”look-and-feels” (Windows, Linux, Mac, Swing etc)
- Now there is JavaFX
  - Oracle has stopped development of Swing
  - JavaFX will replace Swing



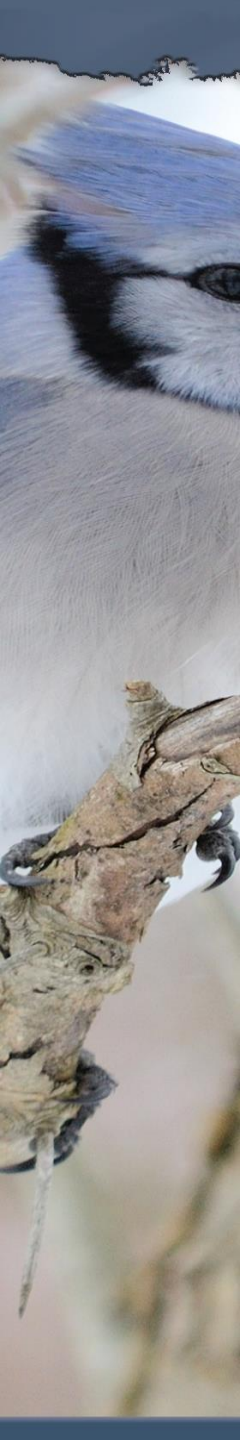
Web vs Desktop

- Desktop GUI
  - GUI in an application running on your device.
  - Usually is a part of an application that has to be installed on your device
  - Often called ”Rich Client Interface” – although you will get rich client interfaces on web today with HTML 5++
- Web GUI
  - Runs in a web-browser
  - Some requires installation of plugin (Microsoft Silverlight, Adobe Flash)
  - With the new HTML 5 + CSS + Javascripts, no need for plugins
  - Is taking over for desktop GUI more and more
- So what is ”best”?
  - Depends on the domain, on the us, on the hardware etc.

GUI Frameworks

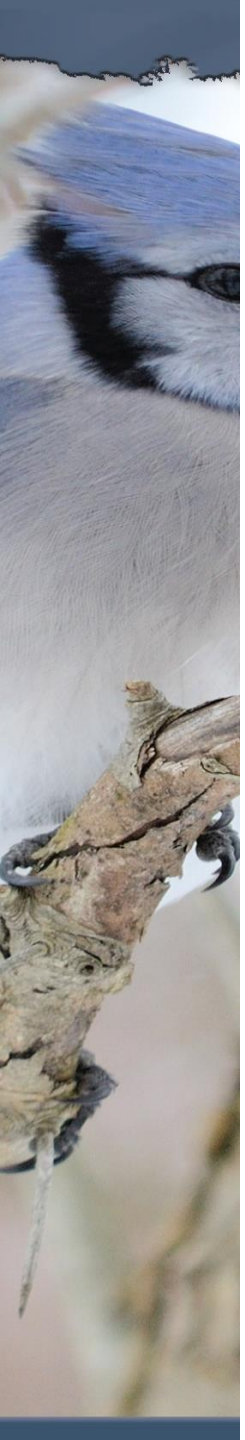
High-level	Cross-platform, by language	C		GTK+ · IUP · Tk · wxC · XForms · XVT
		C++		CEGUI · CLX · FLTK · FOX toolkit · GLUI · Gtkmm · JUCE · Nana · Qt · Rogue Wave Views · TnFOX · Ultimate++ · VCF · Wt · wxWidgets · YAAF
		Objective-C		GNUstep
		CLI	Desktop	Gtk# · Tao (OpenTK, TaoClassic) · wx.NET · UIML.NET · MonoGame
			Web	Moonlight
			Mobile	MonoGame · Xamarin.Forms
		D		DFL · DlangUI · DWT · GtkD · QtD · wxD
		Flash		Apache Flex (MXML)
		Haskell		Gtk2Hs · wxHaskell
		Java	Desktop	AWT · FXML (JavaFX) · Qt Jambi · Swing · SWT · wx4j · jUIML
			Web	GWT · FXML (JavaFX)
			Mobile	LWUIT
		JavaScript		Dojo Toolkit · Echo · Ext JS · Google Closure · jQuery UI · Qooxdoo · YUI
		Common Lisp		CAPI · CLIM · Common Graphics · Ltk · McCLIM
		Lua		IUP · wxLua
		Pascal	Desktop	LCL
			Mobile	LCL
		Object Pascal	Desktop	CLX · fpGUI · IP Pascal · LCL
			Mobile	LCL
		Perl		Perl/Tk
		PHP		PHP-GTK · PHP-Qt · wxPHP
		Python		PyGObject · PyGTK · Pyjs · PyQt · PySide · Tkinter · wxPython
		Ruby		Shoes · QtRuby · wxRuby
		Tcl		Tcl/Tk
		XML		Ample SDK · GladeXML · Lively Kernel · Pyjs · Rialto Toolkit · XAML · XUI · XUL · Wt
		shell		whiptail · dialog

[wikipedia]



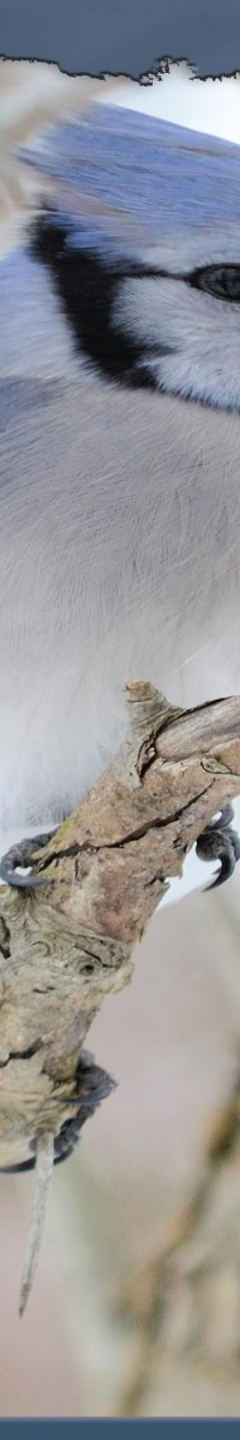
GUI Principles

- Components: GUI building blocks.
  - Buttons, menus, sliders, etc.
- Layout: arranging components to form a usable GUI.
  - Using layout *managers*.
- Events: reacting to user input.
  - Button presses, menu selections, etc.



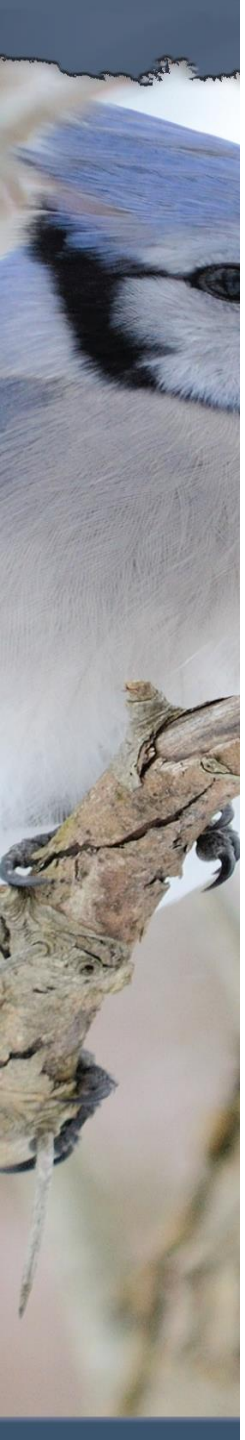
Java code or XML?

- With JavaFX we may build the user interface using java code or (F)XML.
- We will use java code in this course, but feel free to explore FXML.
- You may also find drag and drop programs (like Scenebuilder) that will create code for you. Try it, if it helps your learning☺



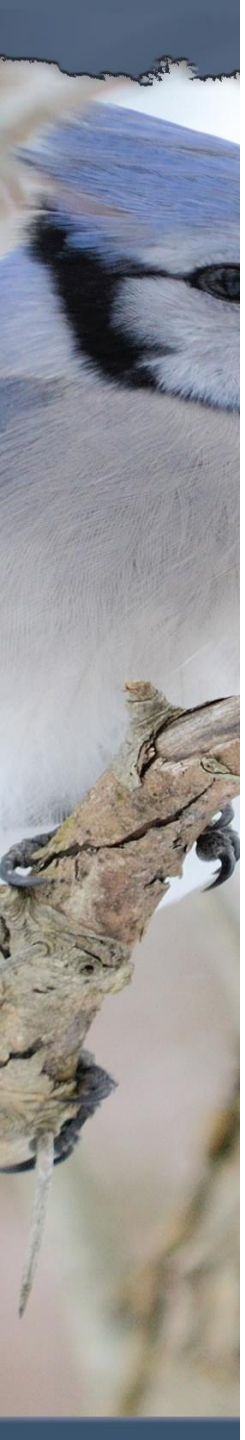
Tutorials

- Oracle
- Tutorialspoint
- Jenkov



Hello World!

- We create an application displaying window named “Hello world”.
  - 2 imports.
  - Inherit from javafx.application.Application
  - Override start(Stage primaryStage) throws Exception



javafx packages

- All packages
- We find
  - javafx.application
  - javafx.stage

```
public abstract class Application
extends Object
```

Application class from which JavaFX applications extend.

Life-cycle

The entry point for JavaFX applications is the Application class. The JavaFX runtime does the following, in order, whenever an application is launched:

1. Constructs an instance of the specified Application class
2. Calls the `init()` method
3. Calls the `start(javafx.stage.Stage)` method
4. Waits for the application to finish, which happens when either of the following occur:
    - the application calls `Platform.exit()`
    - the last window has been closed and the `implicitExit` attribute on `Platform` is true
5. Calls the `stop()` method

Note that the `start` method is abstract and must be overridden. The `init` and `stop` methods have concrete implementations that do nothing.

Calling `Platform.exit()` is the preferred way to explicitly terminate a JavaFX Application. Directly calling `System.exit(int)` is an acceptable alternative, but doesn't allow the Application `stop()` method to run.

A JavaFX Application should not attempt to use JavaFX after the FX toolkit has terminated or from a ShutdownHook, that is, after the `stop()` method returns or `System.exit(int)` is called.

```
public class Stage
extends Window
```

The JavaFX Stage class is the top level JavaFX container. The primary Stage is constructed by the platform. Additional Stage objects may be constructed by the application.

Stage objects must be constructed and modified on the JavaFX Application Thread.

Many of the Stage properties are read only because they can be changed externally by the underlying platform and therefore must not be bindable.

void	**setTitle**(**String** value) Sets the value of the property title.
void	**show**() Attempts to show this Window by setting visibility to true

Style

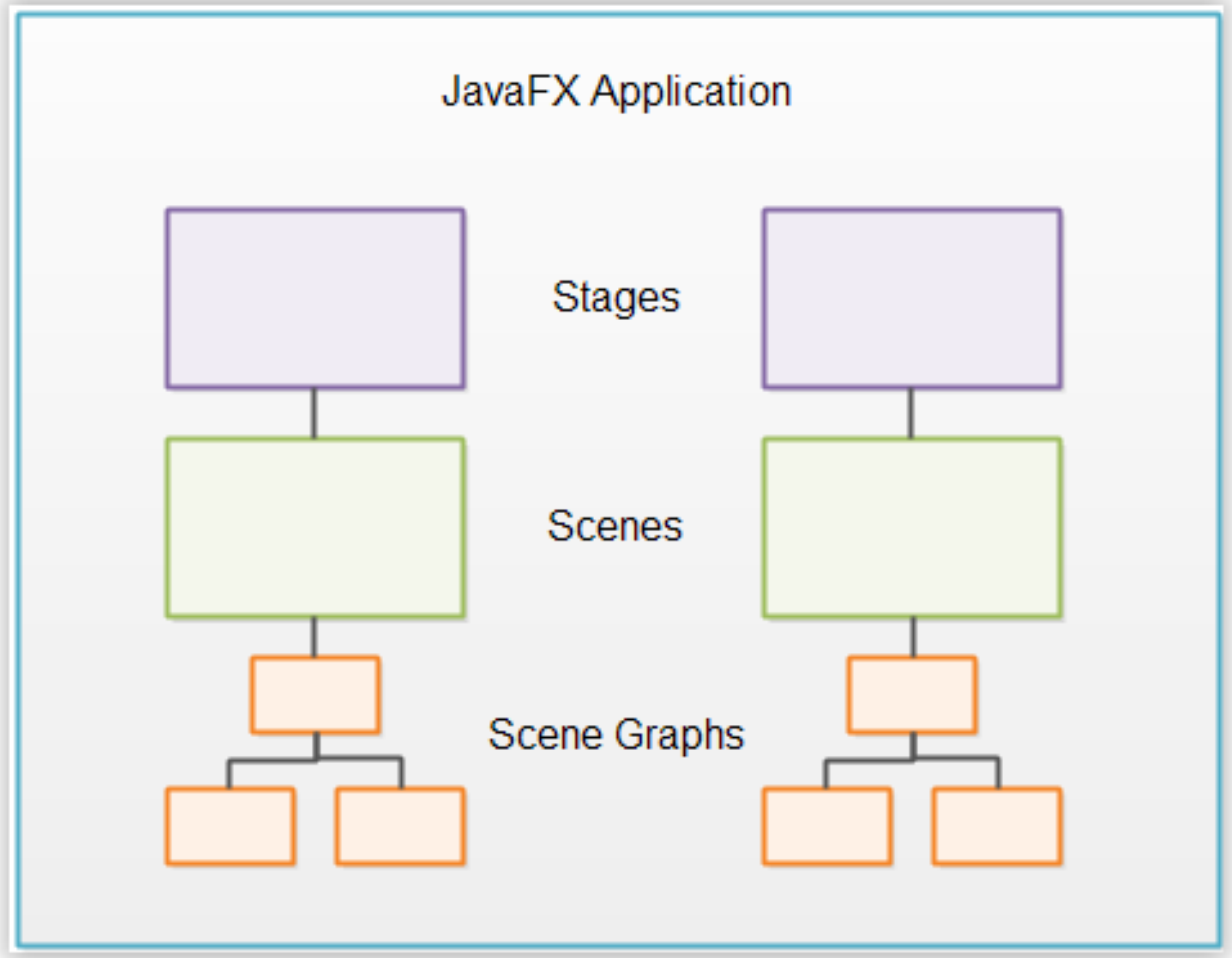
A stage has one of the following styles:

- `StageStyle.DECORATED` - a stage with a solid white background and platform decorations.
- `StageStyle.UNDECORATED` - a stage with a solid white background and no decorations.
- `StageStyle.TRANSPARENT` - a stage with a transparent background and no decorations.
- `StageStyle.UTILITY` - a stage with a solid white background and minimal platform decorations.

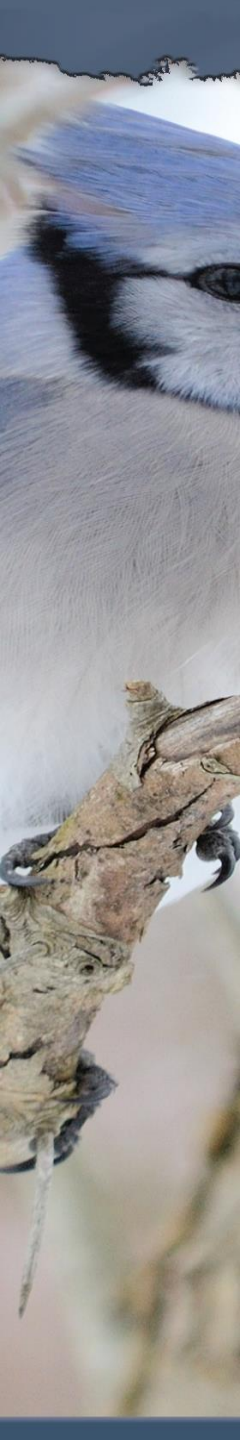
The style must be initialized before the stage is made visible.

On some platforms decorations might not be available. For example, on some mobile or embedded devices. In these cases a request for a DECORATED or UTILITY window will be accepted, but no decorations will be shown.

Stage, Scenes and Scene Graphs

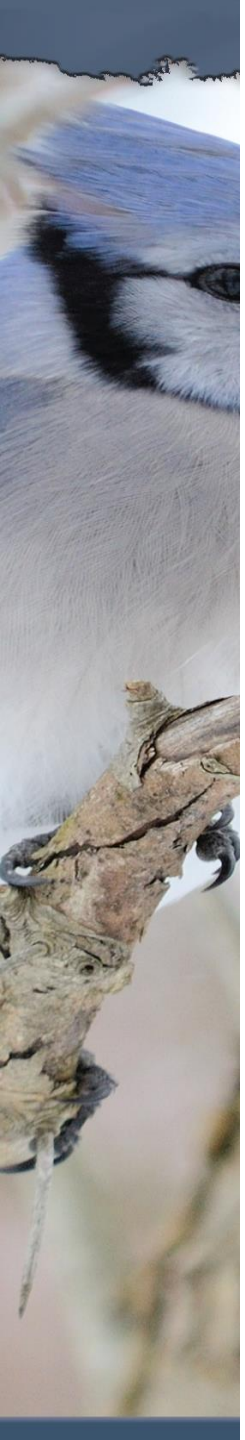


jenkov.com



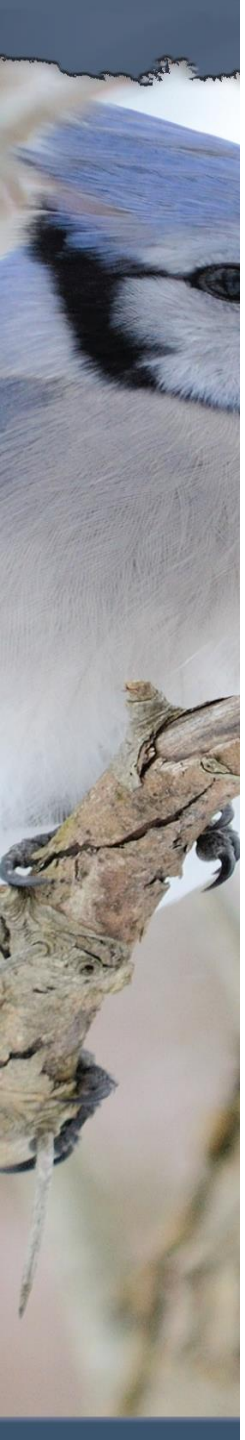
Stage

- The *stage* is the outer frame for a JavaFX application. The stage typically corresponds to a window.
- When used in a desktop environment, a JavaFX application can have multiple windows open. Each window has its own stage.



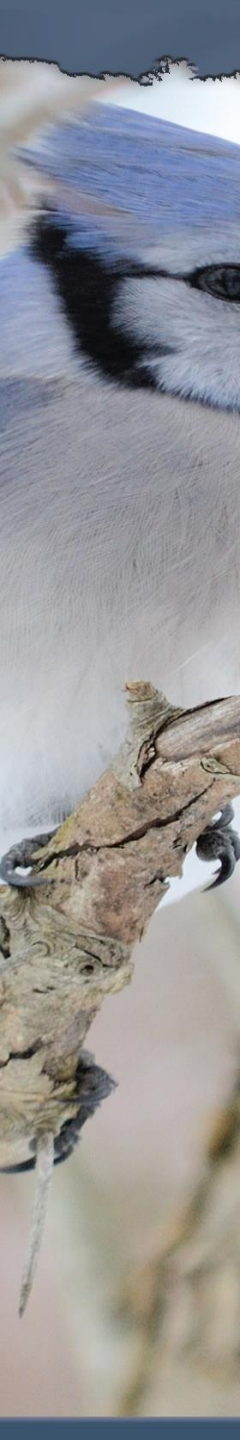
Stage, cont.

- Each stage is represented by a Stage object inside a JavaFX application.
- A JavaFX application has a primary Stage object which is created for you by the JavaFX runtime.
- A JavaFX application can create additional Stage objects if it needs additional windows open. For instance, for dialogs, wizards etc.



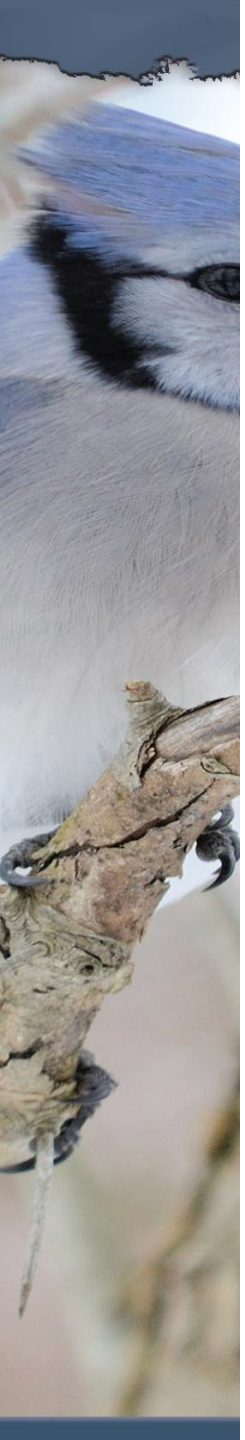
Scene

- To display anything on a stage in a JavaFX application, you need a *scene*.
- A stage can only show one scene at a time, but it is possible to exchange the scene at runtime.
- A scene is represented by a Scene object inside a JavaFX application. A JavaFX application must create all Scene objects it needs.



Scene Graph

- All visual components (controls, layouts etc.) must be attached to a scene to be displayed, and that scene must be attached to a stage for the whole scene to be visible.
- The total object graph of all the controls, layouts etc. attached to a scene is called the *scene graph*.

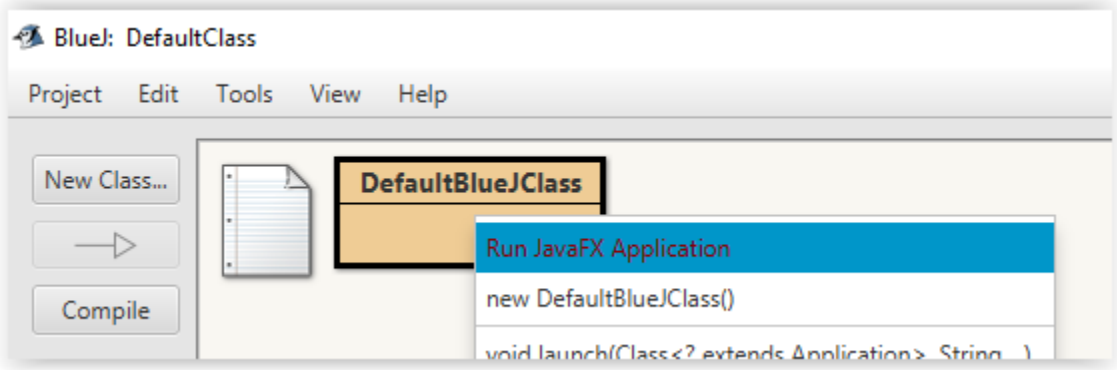


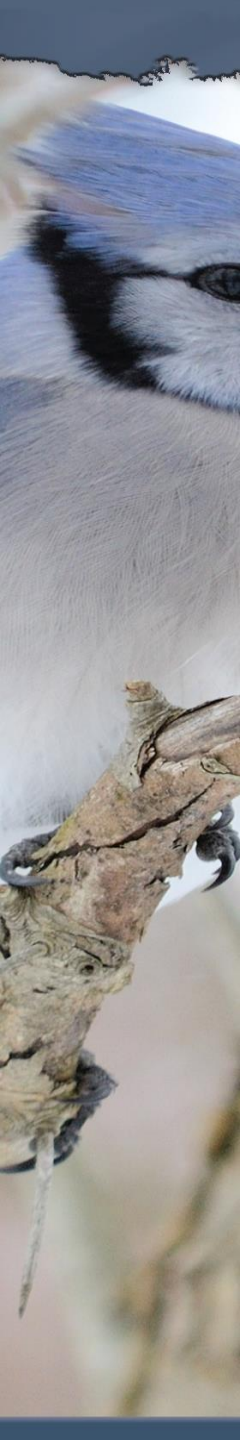
Nodes

- All components attached to the scene graph are called nodes. All nodes are subclasses of a JavaFX class called javafx.scene.Node .
- There are two types of nodes: Branch nodes and leaf nodes.
  - A branch node is a node that can contain other nodes (child nodes). Branch nodes are also referred to as parent nodes because they can contain child nodes.
  - A leaf node is a node which cannot contain other nodes.

JavaFX-class in BlueJ

- We create a default JavaFX-class in BlueJ.
- Non-BlueJ users may find the code in ITL…





New packages

- javafx.event
- javafx.geometry
- javafx.scene.control
- javafx.scene.layout

javafx.scene.control.Label

```
private Label myLabel = new Label("0");
```

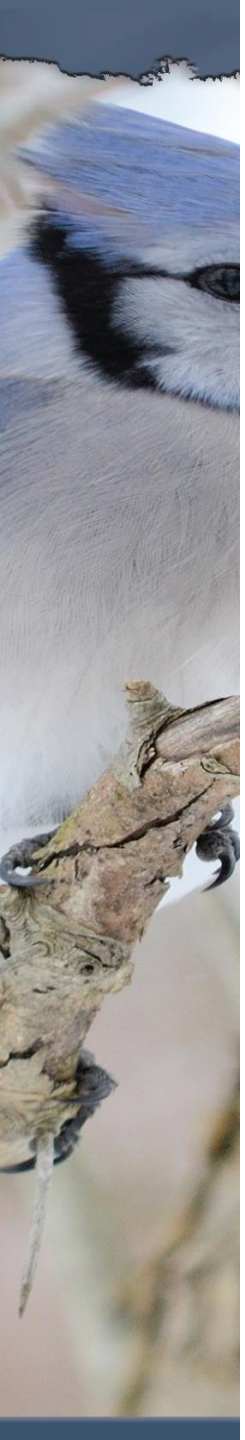
```
public class Label
extends Labeled
```

Label is a non-editable text control. A Label is useful for displaying text that is required to fit within a specific space, and thus may need to use an ellipsis or truncation to size the string to fit. Labels also are useful in that they can have mnemonics which, if used, will send focus to the Control listed as the target of the `labelFor` property.

Label sets focusTraversable to false.

Example:

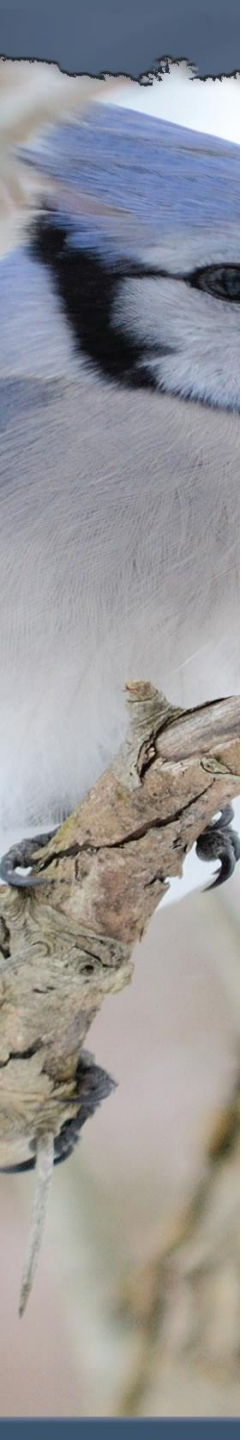
```
Label label = new Label("a label");
```



Controls

- JavaFX controls are JavaFX components which provide some kind of control functionality inside a JavaFX application. For instance, a button, radio button, table, tree etc.
- For a control to be visible it must be attached to the scene graph of some Scene object.
- Controls are usually nested inside some JavaFX **layout component** that manages the **layout** of controls relative to each other.

jenkov.com



Control examples

- Button
- CheckBox
- ColorPicker
- Label
- Menu
- ProgressBar
- RadioButton
- Spinner
- SplitPane
- TabPane
- TextArea
- TextField
- ToolBar
- TreeView

jenkov.com

javafx.scene.control.Button

```
Button myButton = new Button("Count");
```

```
public class Button
extends ButtonBase
```

A simple button control. The button control can contain text and/or a graphic. A button control has three different modes

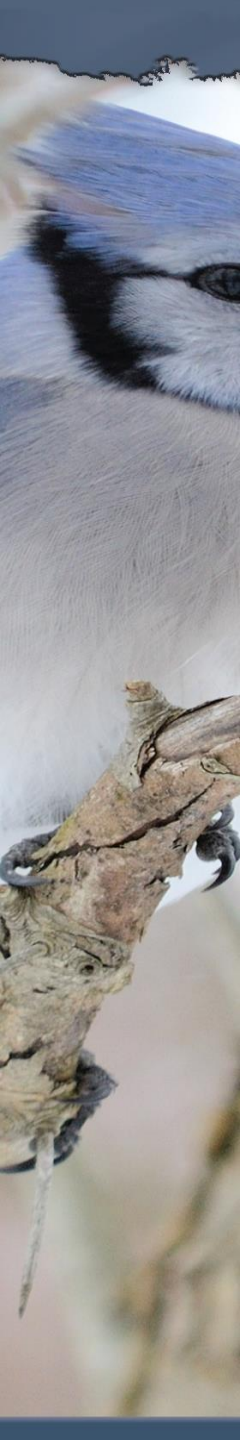
- Normal: A normal push button.
- Default: A default Button is the button that receives a keyboard VK_ENTER press, if no other node in the scene consumes it.
- Cancel: A Cancel Button is the button that receives a keyboard VK_ESC press, if no other node in the scene consumes it.

When a button is pressed and released a `ActionEvent` is sent. Your application can perform some action based on this event by implementing an `EventHandler` to process the `ActionEvent`. Buttons can also respond to mouse events by implementing an `EventHandler` to process the `MouseEvent`

MnemonicParsing is enabled by default for Button.

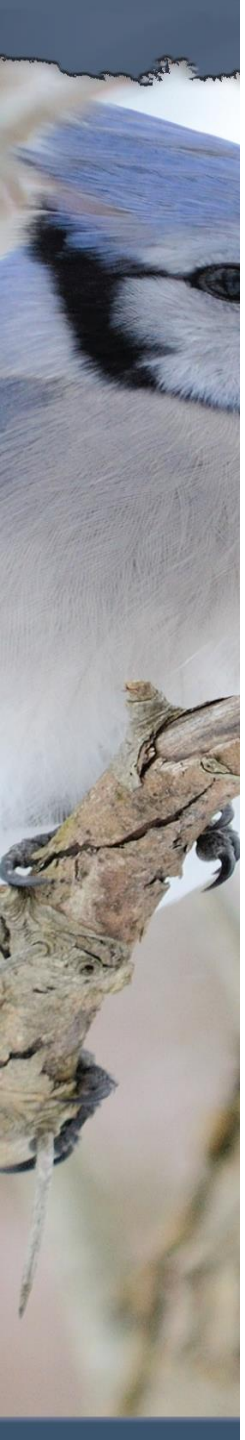
Example:

```
Button button = new Button("Click Me");
```



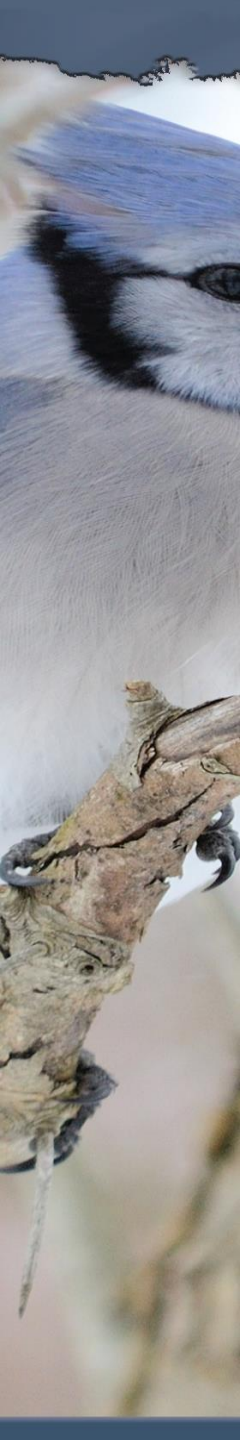
Task!

- Investigate what “MnemonicParsing” is.
- Try to change the code so that you may press the button using your keyboard with the hotkey “c”.



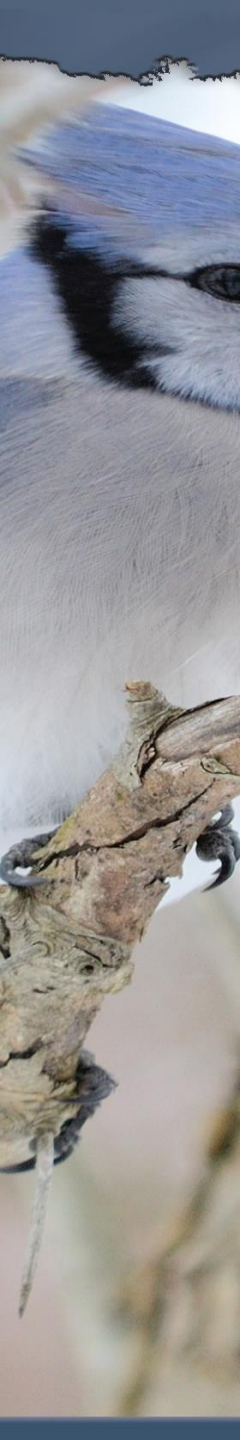
Layouts

- *JavaFX layouts* are components which contains other components inside them.
- The layout component manages the layout of the components nested inside it.
- A layout component must be attached to the scene graph of some Scene object to be visible.



Layout examples

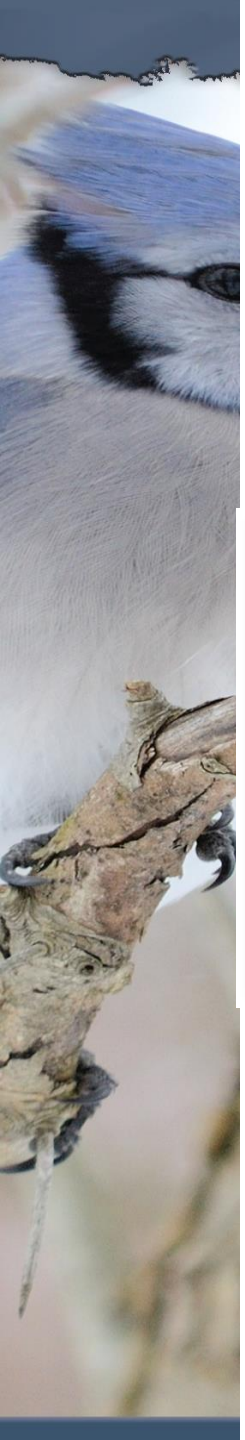
- Pane
- Hbox
- Vbox
- GridPane



Nested layouts

- It is possible to nest layout components inside other layout components.
- This can be useful to achieve a specific layout.
- For instance, to get horizontal rows of components which are not layed out in a grid, but differently for each row, you can nest multiple HBox layout components inside a VBox component.

GridPane



public class **GridPane**
extends Pane

GridPane lays out its children within a flexible grid of rows and columns. If a border and/or padding is set, then its content will be layed out within those insets.

A child may be placed anywhere within the grid and may span multiple rows/columns. Children may freely overlap within rows/columns and their stacking order will be defined by the order of the gridpane's children list (0th node in back, last node in front).

GridPane may be styled with backgrounds and borders using CSS. See Region superclass for details.

```
GridPane pane = new GridPane();
```

Insets (innfellinger)

```
public class Insets
extends Object
```

A set of inside offsets for the 4 side of a rectangular area

Constructor Detail

Insets

```
public Insets(double top,
              double right,
              double bottom,
              double left)
```

Constructs a new Insets instance with four different offsets.

```
new Insets(10, 10, 10, 10)
```

GridPane, cont.

setPadding

```
public final void setPadding(Insets value)
```

Sets the value of the property padding.

```
pane.setPadding(new Insets(10, 10, 10, 10));
pane.setMinSize(300, 300);
```

setMinSize

```
public void setMinSize(double minWidth,
                       double minHeight)
```

Convenience method for overriding the region's computed minimum width and height. This should only be called if the region's internally computed minimum size doesn't meet the application's layout needs.

setVgap

```
public final void setVgap(double value)
```

Sets the value of the property vgap.

```
pane.setVgap(10);
pane.setHgap(10);
```

setHgap

```
public final void setHgap(double value)
```

Sets the value of the property hgap.

DoubleProperty	**hgap** The width of the horizontal gaps between columns.
DoubleProperty	**vgap** The height of the vertical gaps between rows.

GridPane, cont.

add

```
public void add(Node child,
                int columnIndex,
                int rowIndex)
```

Adds a child to the gridpane at the specified column,row position. This convenience method will set the gridpane column and row constraints on the child.

```
// Add the button and label into the pane
pane.add(myLabel, 1, 0);
pane.add(myButton, 0, 0);
```

ActionEvent

- Topic next week☺

```
//set an action on the button using method reference
myButton.setOnAction(this::buttonClick);
```

Setting the scene

Scene

```
public Scene(Parent root,
             double width,
             double height)
```

Creates a Scene for a specific root Node with a specific size.

```
// JavaFX must have a Scene (window content) inside a Stage
Scene scene = new Scene(pane, 300,100);
stage.setTitle("JavaFX Example");
stage.setScene(scene);

// Show the Stage (window)
stage.show();
```

HBox

```
public class HBox
extends Pane
```

HBox lays out its children in a single horizontal row. If the hbox has a border and/or padding set, then the contents will be layed out within those insets.

HBox example:

```
HBox hbox = new HBox(8); // spacing = 8
hbox.getChildren().addAll(new Label("Name:"), new TextBox());
```

VBox

```
public class VBox
extends Pane
```

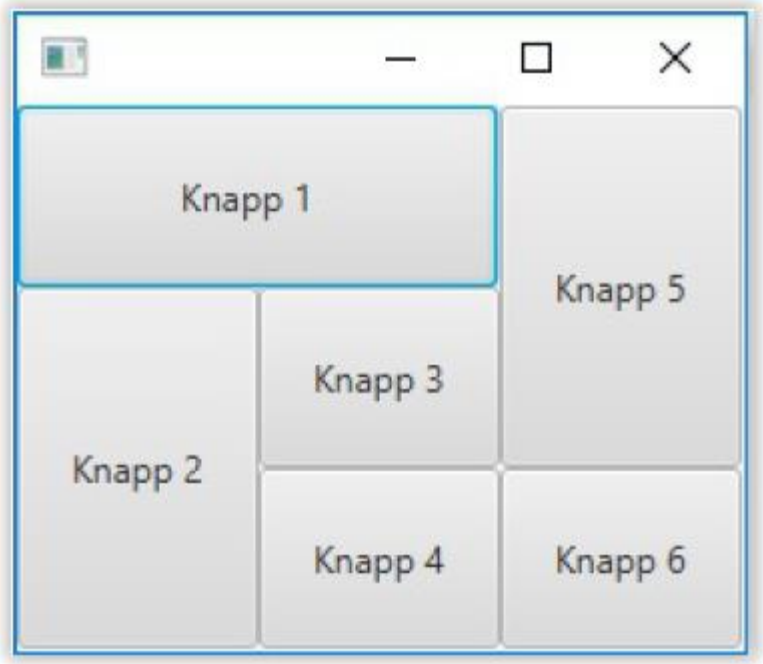
VBox lays out its children in a single vertical column. If the vbox has a border and/or padding set, then the contents will be layed out within those insets.

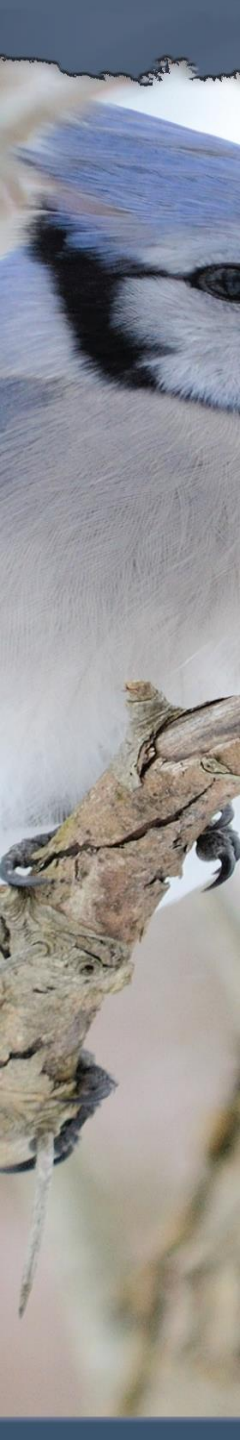
VBox example:

```
VBox vbox = new VBox(8); // spacing = 8
vbox.getChildren().addAll(new Button("Cut"), new Button("Copy"), new Button("Paste"));
```

Øvingsoppgave

- Eksamensoppgave fra Oslo Met 2017:





Nå

- Kahoot☺
- Deretter øving her på Fjerdingen (sjekk TimEdit for rom)

- Neste uke: JavaFX(2)