Linnéuniversitetet

Graphical User Interfaces

using JavaFX

Tobias Andersson Gidlund

`tobias.andersson.gidlund@lnu.se`

CC BY NC SA

Agenda

Graphical User Interfaces

- Graphical User Interfaces (GUI) has been the natural way of communicating with the computer since the 80’s.
- Many different kinds of GUIs have existed – and still exist.
  - Windows 3.x, 95, 8 (Metro/Modern UI) and now 10.
  - MacOS 1, 5, 7, 8 and now X.
  - Motif and now GTK+ as well as KDE/QT.
- Most of the design has its foundation in the work done at Xerox PARC in the 70’s.
  - Coined and used WIMP (Windows, Icons, Menu and Pointer) with different additions.
- The development has evolved steadily with new platforms like smart phones.

Programming of user interfaces

- In most cases there is no direct connection between the programming language and GUI programming.
- In many cases, like Windows 3 – Vista and Motif – most of the GUI is written in C.
- This is then called from different programming languages using *bindings*.
  - It is, for example, possible to program in Pascal and make calls to C-libraries.
- This way is not always very agile.
  - It is also very specific for a platform.
- Most programming languages today are object oriented, which C-libraries are not.

Window in Windows using Win32 in C, part 1

```c
#include <windows.h>

const char g_szClassName[] = ”myWindowClass”;

LRESULT CALLBACK WndProc(HWND hwnd, UINT msg, WPARAM wParam, LPARAM lParam)
{
    switch(msg)
    {
        case WM_CLOSE:
            DestroyWindow(hwnd);
        break;
        case WM_DESTROY:
            PostQuitMessage(0);
        break;
        default:
            return DefWindowProc(hwnd, msg, wParam, lParam);
    }
    return 0;
}

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
    LPSTR lpCmdLine, int nCmdShow)
{
    WNDCLASSEX wc;
    HWND hwnd;
    MSG Msg;
```

Window in Windows using Win32 in C, part 2

```
wc.cbSize        = sizeof(WNDCLASSEX);
wc.style         = 0;
wc.lpfnWndProc   = WndProc;
wc.cbClsExtra    = 0;
wc.cbWndExtra    = 0;
wc.hInstance     = hInstance;
wc.hIcon         = LoadIcon(NULL, IDI_APPLICATION);
wc.hCursor       = LoadCursor(NULL, IDC_ARROW);
wc.hbrBackground = (HBRUSH)(COLOR_WINDOW+1);
wc.lpszMenuName  = NULL;
wc.lpszClassName = g_szClassName;
wc.hIconSm       = LoadIcon(NULL, IDI_APPLICATION);

if(!RegisterClassEx(&wc))
{
    MessageBox(NULL, "Window Registration Failed!", "Error!",
        MB_ICONEXCLAMATION | MB_OK);
    return 0;
}
hwnd = CreateWindowEx(
    WS_EX_CLIENTEDGE,
    g_szClassName,
    "The title of my window",
    WS_OVERLAPPEDWINDOW,
    CW_USEDEFAULT, CW_USEDEFAULT, 240, 120,
    NULL, NULL, hInstance, NULL);
```

Window in Windows using Win32 in C, part 3

```
    if(hwnd == NULL)
    {
        MessageBox(NULL, "Window Creation Failed!", "Error!",
            MB_ICONEXCLAMATION | MB_OK);
        return 0;
    }

    ShowWindow(hwnd, nCmdShow);
    UpdateWindow(hwnd);

    // Step 3: The Message Loop
    while(GetMessage(&Msg, NULL, 0, 0) > 0)
    {
        TranslateMessage(&Msg);
        DispatchMessage(&Msg);
    }
    return Msg.wParam;
}
```

Programming graphical user interfaces today

- Many different programming environments for GUI building and languages exist today.
  - Visual Studio (for .Net languages such as C# and Visual Basic.Net)
  - C++ Builder
  - QT Creator (C++)
  - Gambas (Basic)
- Java has a standardised way of creating GUIs in code but not visually.
  - Today there is an almost standard way, more on that later.
- Just as earlier there are many libraries for GUI building that work for several languages.
  - But seldom a GUI builder for the specific language.

Graphical user interfaces in Java

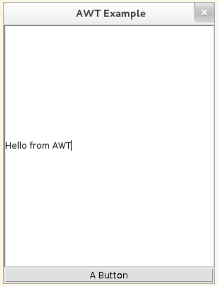
- Java was an early language to have a standardised way of creating GUIs.
  - When the language was released in 1995, really only Visual Basic could compete.
- Apart form that, Java promised ”Write Once, Run Everywhere”.
  - This also for graphical programs.
- Till today, three different standard ways of creating GUIs in Java have been released.
  - Abstract Window Toolkit
  - Swing
  - JavaFX

Abstract Window Toolkit

- The first try of was called *Abstract Window Toolkit* (AWT) and was released with the first version of Java.
- AWT is a thin wrapper around the GUI of the operating system.
  - This is called *heavyweight components*.
- With AWT the programs get the exact same looks as the other programs on the platform.
- The problem with this is that different elements, like buttons and dropdown menus look differently and take differently much space on different platforms.
- It also means that Java becomes vulnerable to changes to the platform.
  - For example when going from Windows 3.x to 95 or MacOS 9 to MacOS X.

Example

```
package lecture5;

import java.awt.Button;
import java.awt.Frame;
import java.awt.TextField;

public class GUI_AWT {
    public static void main(String[] args) {
        Frame theFrame = new Frame("The Window");
        Button theButton = new Button("A Button");
        TextField theText = new TextField("Hello from AWT");
        theFrame.setTitle("AWT Example");
        theFrame.setSize(300, 400);
        theFrame.add("Center", theText);
        theFrame.add("South",theButton);
        theFrame.setVisible(true);
    }
}
```

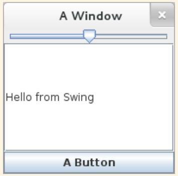


Swing

- With Java 1.2 (called Java 2) from December 1998 came Swing.
- In Swing the components are *lightweight*.
- All components are created using Java's own drawing commands instead of relaying on the operating system.
- This makes the UI look the same no matter what platform is used.
  - The standard looks is called *Metal*.
- Swing uses so called *Pluggable-look-and-feel* which makes it possible to change.
  - For example there are styles for most common platforms.

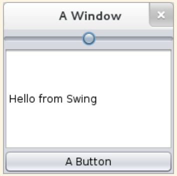
Swing with Metal (standard)

```
package lecture5;
import javax.swing.*;

public class GUI_Swing {
    public static void main(String[] args) {
        JFrame theFrame = new JFrame("A Window");
        JButton theButton = new JButton("A Button");
        JTextField theText = new JTextField("Hello from Swing");
        JSlider theSlider = new JSlider();
        theFrame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
        theFrame.setBounds(100, 100, 200, 200);
        theFrame.setVisible (true );
        theFrame.add("North", theSlider);
        theFrame.add("Center", theText);
        theFrame.add("South", theButton);
        theFrame.setVisible (true );
    }
}
```



Swing with the Nimbus theme

```
package lecture5;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JSlider;
import javax.swing.JTextField;
import javax.swing.UIManager;
import javax.swing.UnsupportedLookAndFeelException;

public class GUI_Swing_Nimbus {
    public static void main(String[] args) throws ClassNotFoundException,
      InstantiationException, IllegalAccessException, UnsupportedLookAndFeelException {
        UIManager.setLookAndFeel("com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel");
        JFrame theFrame = new JFrame("A Window");
        JButton theButton = new JButton("A Button");
        JTextField theText = new JTextField("Hello from Swing");
        JSlider theSlider = new JSlider();
        theFrame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
        theFrame.setBounds(100, 100, 200, 200);
        theFrame.setVisible (true );
        theFrame.add("North", theSlider);
        theFrame.add("Center", theText);
        theFrame.add("South", theButton);
        theFrame.setVisible (true );
    }
}
```



JavaFX

- JavaFX began its life in 2008 but was something completely different from today.
  - The first version was a script language for the web, similar to JavaScript.
  - The last version of the "old" JavaFX was 1.3 and included support for desktop and mobile devices.
- After the acquisition of Sun Microsystems by Oracle in 2010, development of JavaFX was stopped.
- The reason for this, was that Oracle wanted something different of JavaFX.
- Work began and in October 2011, JavaFX 2.0 was announced at JavaONE.

JavaFX today

- Today JavaFX is decoupled from the JDK as the project OpenJFX.
  - This happend with Java 11, previous versions has it built in.
- JavaFX is today:
  - Hardware accelerated
  - Has support for most audio and video formats
  - Can display 3D graphics
  - Has two inbuilt themes, Caspian (old) and Modena (new)
- It, of course, has most widgets one would expect like buttons, input fields, drop down list, dialogues and so on.
- As an addition, to make transition easier, it is also possible to embedd Swing in JavaFX.

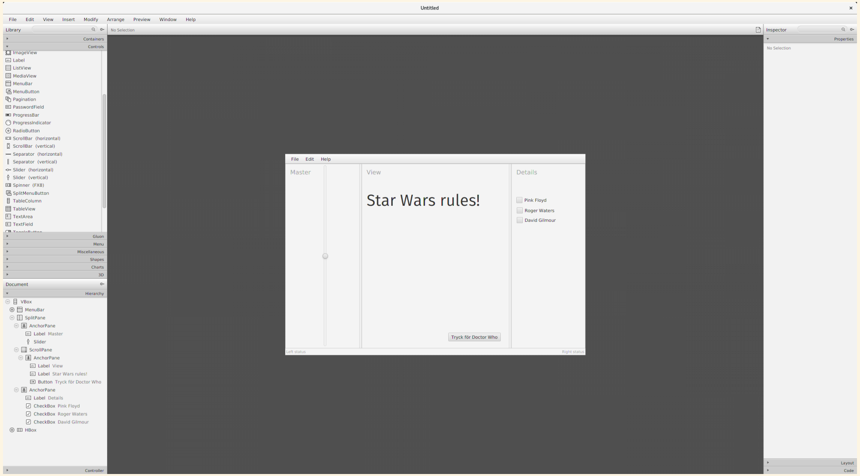
Support in IDEs

- For Java 11 and later, download OpenJFX from `https://openjfx.io/`
- Instructions for installing and using it in your IDE is available at `https://openjfx.io/openjfx-docs/`
  - Download the JavaFX SDK (*not* jmods) and follow the instructions for non-modular projects.
- For Eclipse, a plugin called e(fx)clipse is available.
  - Read more at `http://www.eclipse.org/efxclipse/index.html`
- E(fx)clipse will add support for code completion and makes certain that the program compiles.
  - Search for it in Eclipse Marketplace and install it if it isn't installed already.

Scene Builder

- To build GUIs using Java was from the beginning something you did by hand.
- As time passed, the need for GUI builders rose to make the development faster.
- The problem is that for Java, we now have *several* GUI builders.
- For JavaFX Oracle decided to create *one standard* GUI builder called *Scene Builder*.
- However, Oracle only supplies the source code for the tool but the JavaFX geared company Gluon does supply binaries at: `https://gluonhq.com/products/scene-builder/`

Scene Builder

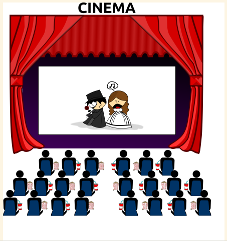


In code or FXML

- The GUI can be built using code just as with Swing.
  - JavaFX works slightly differently, as we will see.
- The other way to code JavaFX is by using *FXML*.
- This is an XML format that defines the GUI.
- Tools like Scene Builder use this and it is then later imported into the code.
  - FXML is seldom coded by hand and instead left to tools to deal with.
- To separate GUI and code in this way means that designers can work on design and programmers on code.
- In this course, we use only code – not FXML or Scene Builder.

The theatre metaphor

- JavaFX uses a *theatre metaphor* to define the interface.
- A theatre consists first and foremost of a *Stage*.
- On the stage, there are several *Scenes* on which the rest of the performing takes place.
- On the scene, several *Nodes* are used.
  - Common nodes are graphics and controls, the visible part of the GUI.

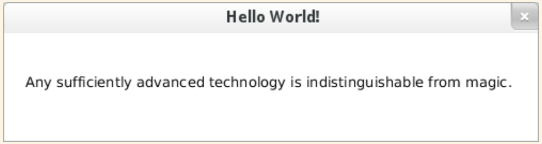


A first example!

- Now it is time for a first example!
- The example is a simple “Hello World!” application with exact positioning.
  - As in Swing, it is possible to use layout management.
  - In contrast to Swing, this is actually quite easy to do!
- The main method does just one thing in a JavaFX program – see to that the `start` method is executed.
  - In fact, it is actually not used at all other than as a fallback.
- All JavaFX applications extend the base class `Application`.

```java
package javafxlecture;

import javafx.application.Application;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.text.Text;
import javafx.stage.Stage;

public class JavaFXLecture extends Application {

    @Override
    public void start(Stage primaryStage) {
        Text text = new Text(20, 50, "Any sufficiently advanced technology is indistinguishable from magic.");

        Group root = new Group();
        root.getChildren().add(text);
        Scene scene = new Scene(root, 500, 100);

        primaryStage.setTitle("Hello World!");
        primaryStage.setScene(scene);
        primaryStage.show();
    }

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

}
```

The result in graphics



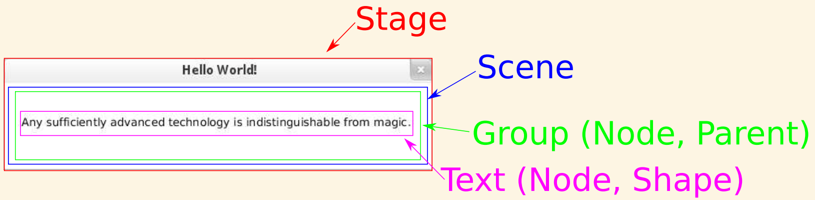
Any sufficiently advanced technology is indistinguishable from magic.

Examining the code

- All action is happening in the `start` method.
- To this parameter, the main `Stage` is sent, that is the window itself.
- A `Text` object is created to display the text.
- As stated previously, most often a layout manager is used to hold the nodes, but in this case we create a `Group`.
  - `Text` is not only a `Node` but also a `Shape` and needs to be put into something.
- To this group, the text is added.
- All programs will need a `Scene` and to this scene we attach the root node, that is all of our GUI.
- Lastly, the `primaryStage` is set up with title and size.

In order

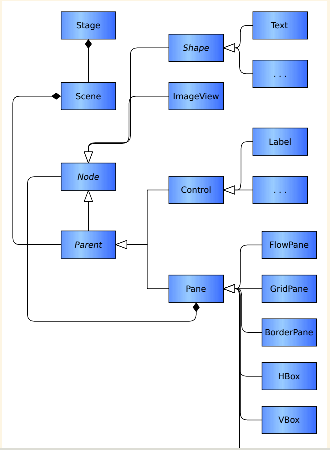
- The following image shows where the different parts are.


Stage
Scene
Group (Node, Parent)
Text (Node, Shape)
Hello World!
Any sufficiently advanced technology is indistinguishable from magic.

- Each Stage can contain several Scenes (although that is not usually the case) and each Scene will contain several Nodes.

Overview

- The following diagram is an overview of how different parts of JavaFX are related using UML.
- Notice that a Scene only can contain a subclass of `Parent` and not `Shape`.
  - This is why a button is acceptable as a root element, but not a piece of text.
- The different panes can contain any number of `Node`s, as this builds the GUI.

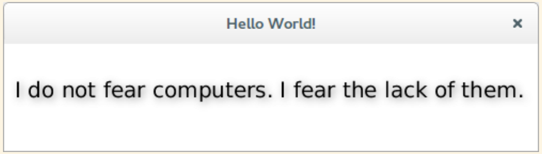


Vector nodes

- All nodes are treated as vector graphics.
- *Effects* are classes which transform or add visual properties to nodes.
- Even though effects are mostly for the fun of it, it can improve use as well.
- JavaFX contains many different effects that can be placed onto most nodes.
  - Most nodes are vector graphics, but even if bitmap images are used they can be decorated with effects.
- All effects are classes that are applied to other nodes.
  - **DropShadow**
  - **GaussianBlur**
  - **InnerShadow**
  - **Reflection**

```
public void start(Stage primaryStage) {
   Text text = new Text(10, 50, "I do not fear computers.
                                 I fear the lack of them.");

   text.setFont(Font.font("SansSerif", 20));

   DropShadow ds = new DropShadow();
   ds.setOffsetX(2.0f);
   ds.setOffsetY(2.0f);
   ds.setColor(Color.rgb(50,50,50,.588));

   text.setEffect(ds);

   Group root = new Group();
   root.getChildren().add(text);
   Scene scene = new Scene(root, 500, 100);

   primaryStage.setTitle("Hello World!");
   primaryStage.setScene(scene);
   primaryStage.show();
}
```

Example two running



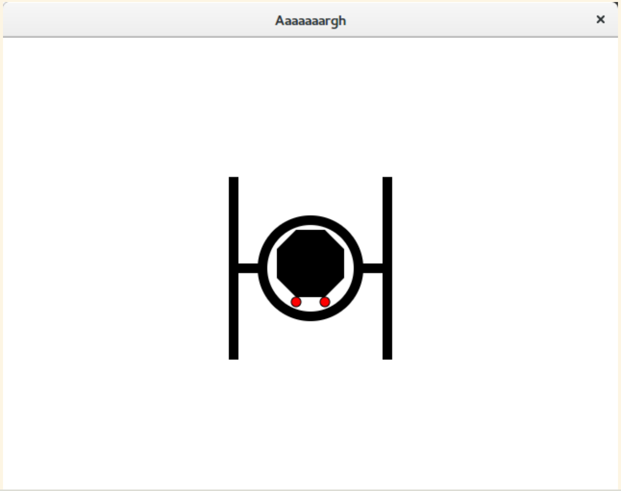
I do not fear computers. I fear the lack of them.

Drawing shapes

- It is also possible to draw simple (and complex) 2D shapes as a node in JavaFX.
  - It is also possible to draw 3D shapes but that is outside the scope of the course.
- As with the positioning of text, the coordinate system in Java sets the (0,0) in the upper left corner.
- Each shape is a class that is added to the scene, some of the classes are:
  - Line
  - Rectangle
  - Circle
  - Arc
- Many of them work more or less the same (set position, size and stroke or filling).

```java
public void start(Stage primaryStage) {

    Group root = new Group();

    Circle cockpit = new Circle(320, 240, 50);
    cockpit.setStroke(Color.BLACK);
    cockpit.setStrokeWidth(10.0);
    cockpit.setFill(null);

    Line connector1 = new Line(250, 240, 270, 240);
    connector1.setStrokeWidth(10.0);

    Line connector2 = new Line(370, 240, 390, 240);
    connector2.setStrokeWidth(10.0);

    Line leftWing = new Line(240, 150, 240, 330);
    leftWing.setStrokeWidth(10.0);

    Line rightWing = new Line(400, 150, 400, 330);
    rightWing.setStrokeWidth(10.0);

    Polygon window = new Polygon();
    window.getPoints().addAll(new Double[]{
       305.0, 200.0,
       335.0, 200.0,
       355.0, 220.0,
       355.0, 250.0,
       335.0, 270.0,
       305.0, 270.0,
       285.0, 250.0,
       285.0, 220.0
    });

    Circle leftCannon = new Circle(305, 275, 5);
    leftCannon.setStrokeWidth(1.0);
    leftCannon.setStroke(Color.BLACK);
    leftCannon.setFill(Color.RED);

    Circle rightCannon = new Circle(335, 275, 5);
    rightCannon.setStrokeWidth(1.0);
    rightCannon.setStroke(Color.BLACK);
    rightCannon.setFill(Color.RED);

    root.getChildren().addAll(cockpit, connector1,
      connector2, leftWing, rightWing, window,
      leftCannon, rightCannon);

    Scene scene = new Scene(root, 640, 480);

    primaryStage.setTitle("Aaaaaaargh");
    primaryStage.setScene(scene);
    primaryStage.show();
}
```

In graphics



Actions

- For most of the time, a GUI should respond to actions from the user.
  - Button clicks, menu selections and many more.
- Since JavaFX is Java, the model is to use event handlers.
- This is done rather easily by using the `setOnAction()` method.
- In it we need to implement the interface method `handle` – preferably using lambda.
- Also notice that anything that needs to be reached inside of this method, needs to be declared as `final`.
  - This as no objects are to be changed in the method, only the contents of them.

Example

- The following code (as done before Java 8):

```
btn.setOnAction(new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent event) {
        System.out.println("Hello World!");
    }
});
```

- Can be replaced with the following in Java 8 and later:

```
btn.setOnAction(event -> {
    System.out.println("Hello World!");
});
```

- The slim notation is possible since the only thing that can be instantiated in this method is an `EventHandler`.

Using an action

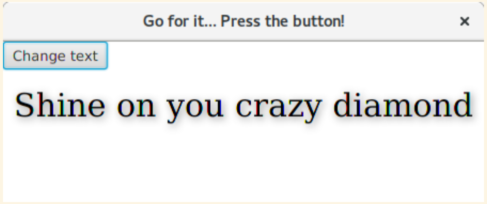
- The button can be constructed and given an action as below:

```
Button theButton = new Button("Change text");

theButton.setOnAction(e -> {
    if(!theText.getText().equals("Shine on you crazy diamond"))
        theText.setText("Shine on you crazy diamond");
    else
        theText.setText("May the Force be with You!");
});
```

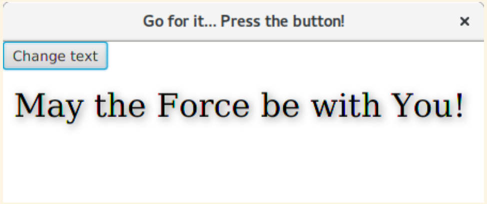
- You may use either this “Java 8” format, or the older with an explicit inner class.
  - It is, however, good to know of the older syntax as there is still a lot of examples on the Internet using it.

```
public void start(Stage primaryStage) {
    primaryStage.setTitle("Go for it... Press the button!");
    final Text theText = new Text(10, 70, "May the Force be with You!");
    theText.setFontSmoothingType(FontSmoothingType.LCD);
    Font font = Font.font("Serif", 30);
    theText.setFont(font);

    DropShadow ds = new DropShadow();
    ds.setOffsetX(2.0f);
    ds.setOffsetY(2.0f);
    ds.setColor(Color.rgb(50, 50, 50, .588));
    theText.setEffect(ds);

    Button theButton = new Button("Change text");

    theButton.setOnAction((ActionEvent e) -> {
        if(!theText.getText().equals("Shine on you crazy diamond"))
            theText.setText("Shine on you crazy diamond");
        else
            theText.setText("May the Force be with You!");
    });

    Group group = new Group();
    group.getChildren().addAll(theButton, theText);

    primaryStage.setScene(new Scene(group, 450, 150));
    primaryStage.show();
}
```

Output



Shine on you crazy diamond



May the Force be with You!

Using layouts

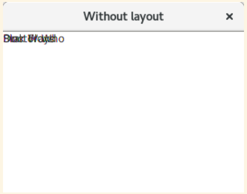
- Since JavaFX still is Java, the use of *layout managers* is a good idea.
- Layouts are used as they make it possible to predict the behaviour of programs on different platforms.
    - The visual relationship with the other components is preserved.
- In JavaFX the layout managers are called *panes*.
- They work by defining an area with a specific behaviour for placing visual components.
    - Stretches according to the layout algorithms of the specific pane.
- It is also possible to manually position the visual components, but make a habit of using panes instead.

VBox and HBox

- Two new and very easy to use layouts are `VBox` and `HBox`.
  - For vertical and horizontal box.
- With these it is very simple to create user interfaces that look as we like.
- To adjust the space in and around the layout, there are two methods:
  - `setPadding()` – for the space around the layout box.
  - `setSpacing()` – for the space around the controls of the box.
- To adjust the size of the visible components themselves, use the `setPrefSize()`.
  - This sets the *preferred* size of the component, but it will adjust itself to the surrounding.

Without layout manager

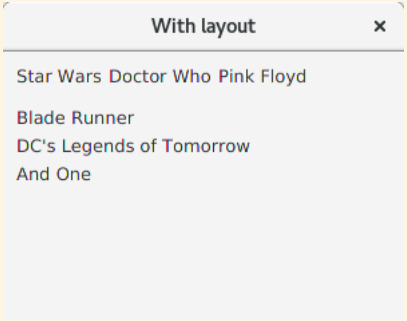
```
public class WithoutLayout extends Application {

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

    @Override
    public void start(Stage primaryStage) {
        Label lblSW = new Label("Star Wars");
        Label lblDW = new Label("Doctor Who");
        Label lblPF = new Label("Pink Floyd");

        Group root = new Group();
        root.getChildren().addAll(lblSW, lblDW, lblPF);

        Scene scene = new Scene(root, 300, 200);
        primaryStage.setTitle("Without layout");
        primaryStage.setScene(scene);
        primaryStage.show();
    }
}
```



With layout manager

```java
public void start(Stage primaryStage) {
    Label lblSW = new Label("Star Wars"); Label lblBR = new Label("Blade Runner");
    Label lblDW = new Label("Doctor Who"); Label lblDC = new Label("DC's Legends of Tomorrow");
    Label lblPF = new Label("Pink Floyd"); Label lblAO = new Label("And One");

    VBox root = new VBox();
    root.setPadding(new Insets(5));
    root.setSpacing(5);

    HBox horizontally = new HBox();
    horizontally.setPadding(new Insets(5));
    horizontally.setSpacing(5);
    horizontally.getChildren().addAll(lblSW, lblDW, lblPF);

    VBox vertically = new VBox();
    vertically.setPadding(new Insets(5));
    vertically.setSpacing(5);
    vertically.getChildren().addAll(lblBR, lblDC, lblAO);

    root.getChildren().addAll(horizontally, vertically);

    Scene scene = new Scene(root, 300, 200);
    primaryStage.setTitle("With layout");
    primaryStage.setScene(scene);
    primaryStage.show();
}
```

Running program



```
Star Wars Doctor Who Pink Floyd

Blade Runner
DC's Legends of Tomorrow
And One
```

More layouts

- The simplest of all panes is called `Pane` and it simple allows the programmer to place nodes on the window.
- As shown, most layouts can be created using a number of `VBoxes` and `HBoxes`.
- `GridPane` is useful for dividing the window into different areas (top, left, centre and so on).
- `AnchorPane` allows for controls to be attached to each others.
- Much more on JavaFX layouts can be studied on
  `https://docs.oracle.com/javase/8/javafx/layout-tutorial/index.html`

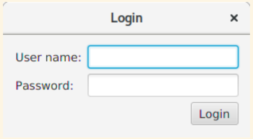
Gridpane

- A useful layout manager for placing controls in a grid is... `GridPane`.
- Each node is placed in columns and rows which are indexed from 0 and up.
- Alignment can be set both for the pane itself but also for the individual nodes.
- Nodes are added using `add()` with two indexes, the first for the column the second for row.
    - The number of columns and rows do not need to be decided in advance, it is calulated from the added nodes.
- In the example, the size of the scene is not set as it too can be calculated from the layout it contains.

Code

```
primaryStage.setTitle("Login");

GridPane pane = new GridPane();
pane.setAlignment(Pos.CENTER);
pane.setPadding(new Insets(11.5, 12.5, 13.5, 14.4));
pane.setHgap(5.5);
pane.setVgap(5.5);

pane.add(new Label("User name:"), 0, 0);
final TextField username = new TextField();
pane.add(username, 1, 0);
pane.add(new Label("Password:"), 0, 1);
final PasswordField password = new PasswordField();
pane.add(password, 1, 1);
final Label result = new Label();
pane.add(result, 0, 2);
Button testLogin = new Button("Login");
pane.add(testLogin, 1, 2);
GridPane.setHalignment(testLogin, HPos.RIGHT);

testLogin.setOnAction(e -> {
    if(username.getText().equals("CharlesClemens") && password.getText().equals("Pink Floyd"))
        result.setText("OK");
    else
        result.setText("No way!");
});
```

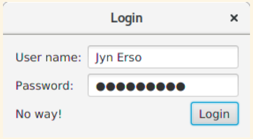
In graphics


Login ×

User name:

Password:

Login

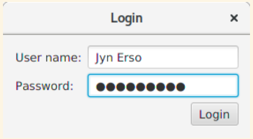

Login ×

User name: Jyn Erso

Password: ●●●●●●●●●

No way!

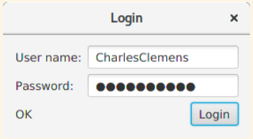
Login


Login ×

User name: Jyn Erso

Password: ●●●●●●●●●

Login


Login ×

User name: CharlesClemens

Password: ●●●●●●●●●●

OK

Login

A few controls

- In this lecture only a few controls will be shown.
  - Enough to make you through the assignments...
- JavaFX contains a large amount of controls for various tasks and it is also possible to create your own.
  - The 3rd party library *ControlsFX* is a great example of that.
- It is not possible in this short time to look at all controls, but this lecture and the next will show a number of the most common.
- A lot more information can be found at:

  `https://docs.oracle.com/javase/8/javase-clienttechnologies.htm`

Inputting text

- An example of text input has already been shown with grid layout.
- The simplest way of entering text is to use `TextField`.
  - All controls inheriting from `TextInput` can be use.
- The most important methods of `TextField` are `setText()` and `getText()`.
- In addition, there are methods for managing a global clipboard.
  - The methods are called `copy()`, `cut()`, `paste()` and `selectAll()`.

Example (In Swedish!!!)

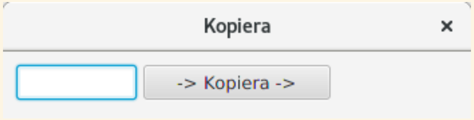


Figure: Program start

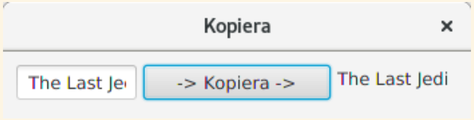


Figure: Text copied from the left to the right

Source code

```java
public void start(Stage primaryStage) {
    TextField left = new TextField();
    left.setPrefSize(100, 20);

    Label right = new Label();
    right.setPrefSize(100, 20);

    Button btn = new Button();
    btn.setText(" -> Kopiera -> "); // -> Copy ->
    btn.setPrefSize(150, 20);
    btn.setOnAction(e ->{
        right.setText(left.getText());
    } );

    HBox root = new HBox();
    root.setPadding(new Insets(10));
    root.setSpacing(5);
    root.getChildren().addAll(left, btn, right);

    Scene scene = new Scene(root, 350, 50);

    primaryStage.setTitle("Kopiera"); // Copy
    primaryStage.setScene(scene);
    primaryStage.show();
}
```

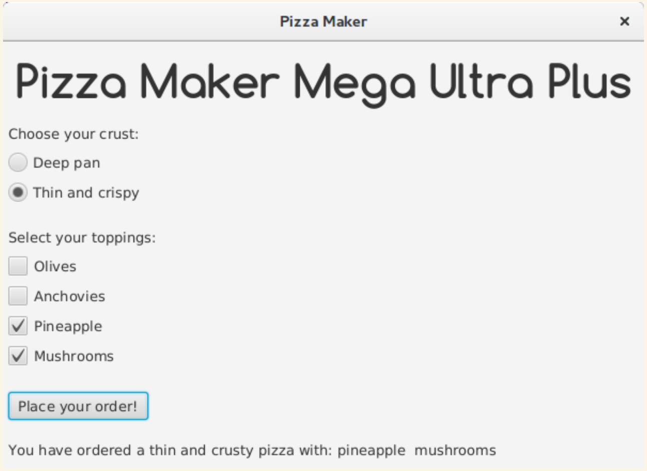
Radio and Check buttons with toggles

- The `RadioButton` and `CheckBox` controls are quite similar.
  - The first allows for one active choice and the other several active choices.
- In most cases both of them are put inside of a `ToggleGroup` to group the choices together.
  - Most important for radio buttons since only one option can be selected at any time.
- There are several ways of identifying the selected choice, but the easiest is to read `isSelected();`
- It is possible to add a listener to the buttons, but in most cases that is considered bad behaviour, it is better to read the values in a button press (or similar).

```
primaryStage.setTitle("Pizza Maker");
VBox layout = new VBox();
layout.setAlignment(Pos.CENTER);
layout.setPadding(new Insets(5, 5, 5, 5));
layout.setSpacing(5);

Label heading = new Label("Pizza Maker Mega Ultra Plus");
heading.setFont(new Font("Comfortaa", 42));

VBox boxCrust = new VBox();
boxCrust.setPadding(new Insets(10, 0, 0, 0));
boxCrust.setSpacing(10);
Label lblCrust = new Label("Choose your crust:");
RadioButton deep = new RadioButton("Deep pan");
RadioButton thin = new RadioButton("Thin and crispy");
ToggleGroup tglCrust = new ToggleGroup();
deep.setToggleGroup(tglCrust);
thin.setToggleGroup(tglCrust);
boxCrust.getChildren().addAll(lblCrust, deep, thin);

VBox boxTopping = new VBox();
boxTopping.setPadding(new Insets(20, 0, 0, 0));
boxTopping.setSpacing(10);
Label lblTopping = new Label("Select your toppings:");
CheckBox olives = new CheckBox("Olives");
CheckBox anchovies = new CheckBox("Anchovies");
CheckBox pineapple = new CheckBox("Pineapple");
CheckBox mushrooms = new CheckBox("Mushrooms");
boxTopping.getChildren().addAll(lblTopping, olives, anchovies, pineapple, mushrooms);
```

```
VBox boxOrder = new VBox();
boxOrder.setPadding(new Insets(20, 0, 0, 0));
boxOrder.setSpacing(20);
Button order = new Button("Place your order!");
final Label answer = new Label();
final StringBuilder theOrder = new StringBuilder("You have ordered ");

order.setOnAction(e -> {
    if(deep.isSelected())
        theOrder.append(" a deep pan pizza with: ");
    else if (thin.isSelected())
        theOrder.append("a thin and crusty pizza with: ");

    if(olives.isSelected())
        theOrder.append("olives ");
    if(anchovies.isSelected())
        theOrder.append("anchovies ");
    if(pineapple.isSelected())
        theOrder.append("pineapple ");
    if(mushrooms.isSelected())
        theOrder.append(" mushrooms");

    answer.setText(theOrder.toString());
});

boxOrder.getChildren().addAll(order, answer);

layout.getChildren().addAll(heading, boxCrust, boxTopping, boxOrder);

Scene scene = new Scene(layout, 600, 400);
```

In graphics



Pizza Maker

Pizza Maker Mega Ultra Plus

Choose your crust:

- ( ) Deep pan
- (●) Thin and crispy

Select your toppings:

- [ ] Olives
- [ ] Anchovies
- [✓] Pineapple
- [✓] Mushrooms

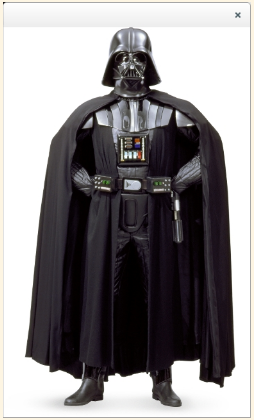
```
You have ordered a thin and crusty pizza with: pineapple  mushrooms
```

Images

- Images are often used in applications of all sorts.
- JavaFX supports a number of common file formats, including JPG, PNG, GIF and BMP.
- To display an image is done it two steps.
  1. Place an `ImageView` control where the images is going to be displayed.
  2. Use an `Image` object to load and hold the image itself.
- The `ImageView` is used as an *view port* of the image.
  - It is possible to show only part of an image or to scroll it in different directions.
- Images are loaded from the default classpath for the project which depends on your IDE.

Example

```
package graphics;
import javafx.application.Application;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.stage.Stage;

public class Test extends Application {
    @Override
    public void start(Stage primaryStage) {
        Image theImage = new Image("vader.png");
        ImageView theIV = new ImageView();
        theIV.setImage(theImage);

        Group group = new Group();
        group.getChildren().add(theIV);
        Scene theScene = new Scene(group);
        primaryStage.setScene(theScene);
        primaryStage.show();
    }

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



More on images

- When looking around on the internet, another way of handling image resources might appear.
- It is possible to import the image to display into the package.
- However, to be able to reach the file then, another argument to `Image` is needed:

```
Image theImage = new Image(getClass().getResourceAsStream("tux.png"));
```

- The method `getClass()` returns the `Class` object that represents the runtime class of this object.
- Sometimes, depending on IDE, another solution is to add `file:` before the image as in:

```
Image theImage = new Image("file:images/vader.png");
```

More on working with graphics

- The size and other properties of the graphics is controlled via the `ImageView` object.
- To size it to a specific size use either `setFitWidth()` or `setFitHeight()`.
- This will adjust on one axis, but to preserve the aspect ratio, add `setPreserveRatio(true)`.
- The view port of the image can be decided using a `Rectangle2D` object that sets what part of the image to show.
- Many other properties exist, for rotating, smoothing and similar.

Example

- The original image is about 400×400 pixels large.
- In the program a part of the image is selected and rotated.

Figure: Original image.

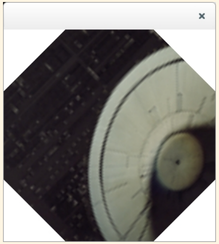


Figure: Viewport and rotation in JavaFX.

Code for example

```
package graphics;
import javafx.application.Application;
import javafx.geometry.Rectangle2D;
import javafx.scene.Group;import javafx.scene.Scene;
import javafx.scene.image.Image;import javafx.scene.image.ImageView;
import javafx.stage.Stage;

public class PartOfImage extends Application {
    @Override
    public void start(Stage primaryStage) {
        Image theImage = new Image("deathstar.png");
        ImageView theView = new ImageView(theImage);
        theView.setFitWidth(300);
        theView.setPreserveRatio(true);
        Rectangle2D rect = new Rectangle2D(200, 100, 100, 100);
        theView.setViewport(rect);
        theView.setRotate(45.0);

        Group group = new Group();
        group.getChildren().add(theView);
        Scene theScene = new Scene(group, 300, 300);
        primaryStage.setScene(theScene);
        primaryStage.show();
    }

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

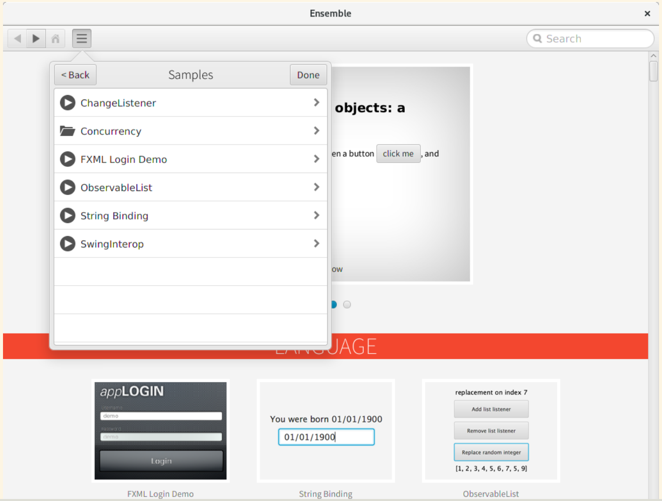
Wrap Up

Ensemble and other demo programs

- Oracle has released a number of demo programs with source code to use to understand JavaFX.
- It can be found on:

  `http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html`
- Two really interesting programs are “Ensemble” and “Modena” as they showcase most of the controls available in JavaFX.
- As the source code is provided, it is possible – and highly recommended – that you have a look at it.
- We end todays lecture with showing two screenshots of those programs.

Ensemble



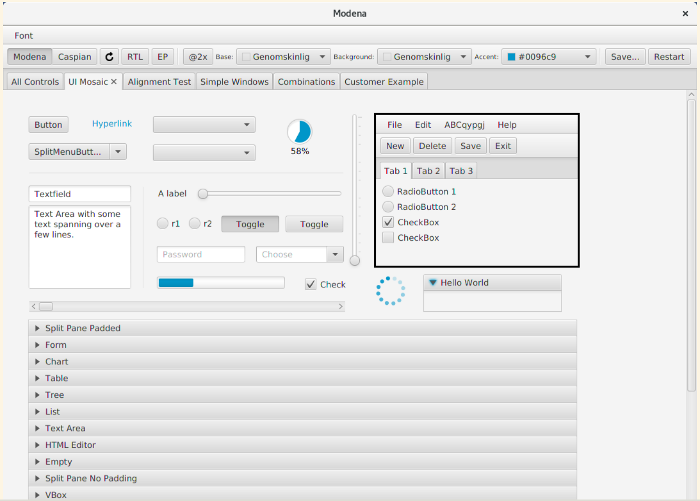
You were born 01/01/1900

01/01/1900

replacement on index 7

[1, 2, 3, 4, 5, 6, 7, 5, 9]

Modena


Modena
Font
Modena
Caspian
RTL
EP
@2x
Base: Genomskinlig
Background: Genomskinlig
Accent: #0096c9
Save...
Restart
All Controls
UI Mosaic
Alignment Test
Simple Windows
Combinations
Customer Example
Button
Hyperlink
SplitMenuButt...
58%
Textfield
Text Area with some text spanning over a few lines.
A label
r1
r2
Toggle
Toggle
Password
Choose
Check
File
Edit
ABCqypgj
Help
New
Delete
Save
Exit
Tab 1
Tab 2
Tab 3
RadioButton 1
RadioButton 2
CheckBox
CheckBox
Hello World
Split Pane Padded
Form
Chart
Table
Tree
List
Text Area
HTML Editor
Empty
Split Pane No Padding
VBox