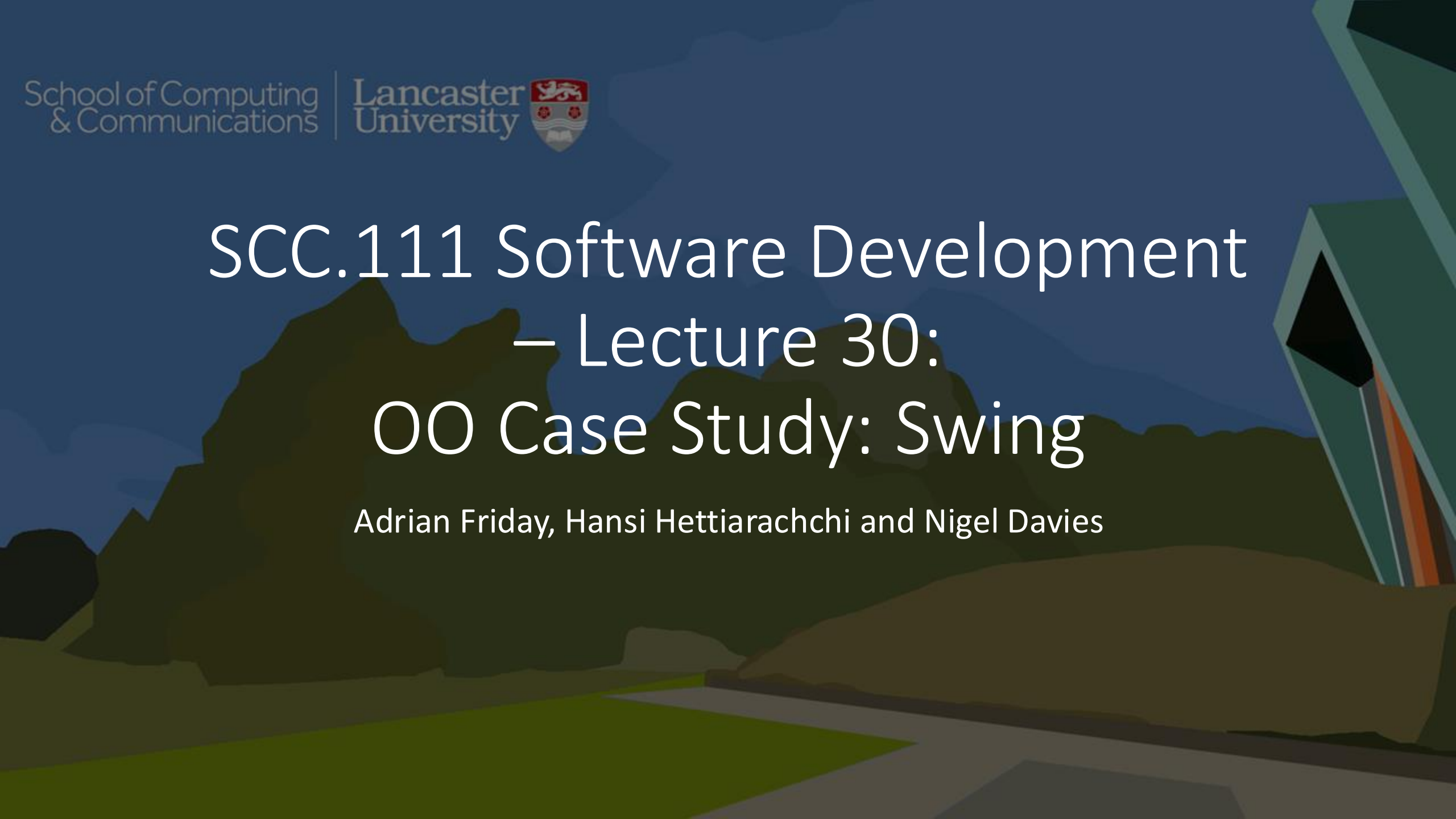


SCC.111 Software Development – Lecture 30: OO Case Study: Swing

Adrian Friday, Hansi Hettiarachchi and Nigel Davies

Introduction

Lancaster University

- Last lecture, we looked at:
  - How to apply the core Oriented Object concepts in Java
  - The Java Class Library
  - How to automatically document our code using JavaDoc

- Today we're going to see a case study of these concepts in action.
  - Develop some working knowledge of core Swing classes
  - **By the end of this lecture, you should be able to write your own simple Graphical User Interface (GUI) in Java**

GUIs in Java: Swing

Lancaster University

- Swing is the standard java package for Graphical User Interfaces (GUIs)
  - Still platform independent, like the rest of Java

- The Swing package is very large and extremely flexible
  - Includes textboxes, sliders, buttons, images
  - We'll look at some of the most basic, common features today
    - Check Java API documentation for more info

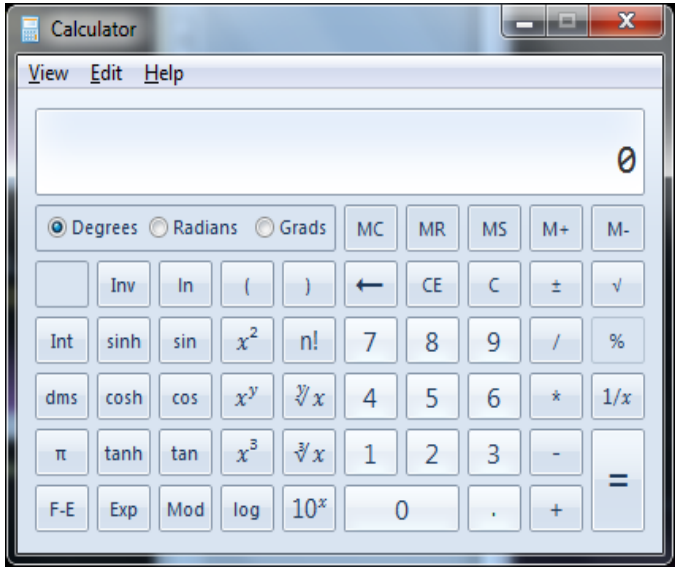
- **Swing is heavily object oriented**
  - So it makes a nice case study in clean OO design
  - And you can apply what you have learned about OO programming!

GUI components

Lancaster University

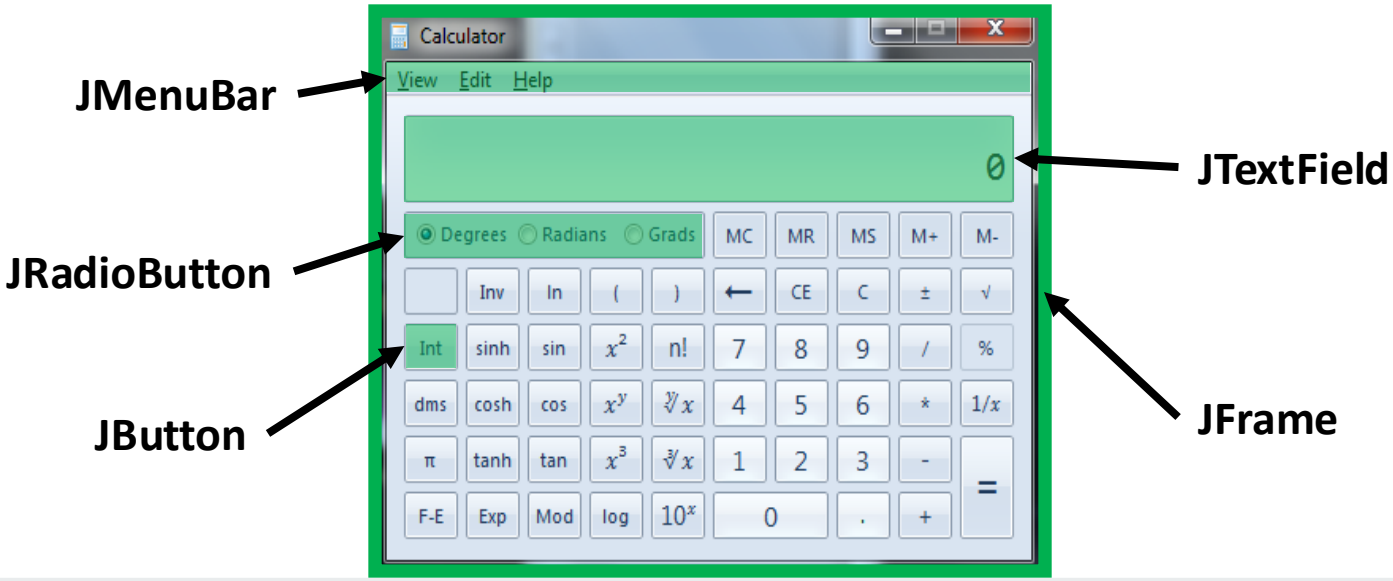
- **GUIs are built from a standard set of components**
  - Traditional GUIs can be broken down into clear parts
  - Windows, Buttons, text input areas, menus, etc.
  - These are often referred to as **components** (a fairly general term in Computer Science)

- For example:



GUI components in Java

- **Graphical components are implemented as Java classes**
  - Each type of component is a different class
  - Implementation is therefore encapsulated
  - The complexity of the task is hidden
  - You don't need to know how it works - just how to use it



Packages

Lancaster University

Swing classes reside in a package called javax.swing

- A package is just a named collection of classes grouped together
- Packages also act much like C++ namespaces…
- Classes from inside packages can be used via full name (e.g. `javax.swing.JFrame`)

The import keyword tells the java compiler of any additional classes packages you want to import into your program's namespace…

- Import statements are always on the first lines of any class file that uses classes from a package, even before a class definition
- Classes from that package then become visible to the java compiler.

```
import javax.swing.*;

public class ...
```

JFrame

Lancaster University

JFrame represents a window in the host Operating System

- Style (Window Decoration) matches local OS.
- Includes title bar, icons to minimize, resize, close, etc.

JFrame constructor has two commonly used forms:

```
JFrame();
JFrame(String title);
```

JFrame contains useful (private) attributes and associated accessor / mutator methods:

```
boolean getVisible();
void setVisible(boolean b);
String getTitle();
void setTitle(String s);
void setSize(int x, int y);
```

JFrame: instantiation

Lancaster University

Create an instance of the JFrame class to create a window

- One instance for every window you need
- Make an instance in the usual way: use **new** to call its constructor!
- Windows are created in an invisible state…why?
- So that all the components can be built before it is displayed to the user…

Invoke methods on the instance to control its behaviour

```
JFrame a = new JFrame();     // Create a blank window
a.setVisible(true);          // Make it visible
a.setTitle("Hello world!"); // Change window title
a.setSize(300, 300);         // Change window size
```

JFrame: closing

Lancaster University

Defining what happens when the window is closed

- **setDefaultCloseOperation(int operation);**
- **JFrame.EXIT_ON_CLOSE** (terminate application)
- **JFrame.DISPOSE_ON_CLOSE** (close window, keep app running)
- **JFrame.DO_NOTHING_ON_CLOSE** (ignore)

```
JFrame a = new JFrame();     // Create a blank window
a.setVisible(true);          // Make it visible
a.setTitle("Hello world!"); // Change window title
a.setSize(300, 300);         // Change window size
a.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

JFrame: an aside…

Lancaster University

- **What is JFrame.EXIT_ON_CLOSE ?**

  Class **Integer variable**

- This is an example of a static final variable:

```
public static final int EXIT_ON_CLOSE = 1;
```

- A **static** variable or method is called on a class, not an instance
  - Use sparingly: Useful only when a more procedural programming style is preferred to OO
  - The variable is also shared between all instances of that class.
  - Don’t use static when defining methods in a typical class
- A **final** variable cannot have it’s value changed after it has been initialized
  - Commonly used technique for defining constants
  - More humanly readable than trying to remember values
  - Make code more readable

Adding components 1

Lancaster University

- **JFrames are responsible for managing the window, not its content**
  - We need a container to hold all our components
  - The JPanel class provides this functionality
- **JPanel** holds a list of components, and provides **add()** and **remove()** methods for Swing components, so:
  - Create an instance of JPanel
  - Set it as the default panel for your new frame

```
import javax.swing.*;
public class HelloWorld
{
    JFrame a = new JFrame();          // Create a blank window
    JPanel panel = new JPanel();      // Create a panel
    a.setContentPane(panel);          // Use panel on Window
}
```

Adding components 2

Lancaster University

- To add a component to a window:
  - Instantiate the component
  - See Java API for details on a component's constructor
    - https://docs.oracle.com/en/java/javase/23/docs/api/
  - Add it to the relevant JPanel

- Common components include:
  - **JButton**
  - **JLabel**
  - **JTextField**

Components: JLabel

Lancaster University

JLabel is the simplest component: a read only text field

- Text cannot be changed by the user on the GUI
- Very useful for prompts, status messages, instructions to the user, etc.

- Several commonly used constructors:

```
JLabel(String s);    //Plain Text
JLabel(Icon i);      //Image
```

- Accessor and mutator methods provided to inspect and change the text displayed:

```
String getText();
void setText(String s);
```

Components: JButton

Lancaster University

JButton is a clickable component

- Useful for performing user-initiated actions

- Several simple constructors:

```
JButton(String s);      // Clickable Text Button
JButton(ImageIcon i);   // Clickable Image Button
```

Components: JTextField

Lancaster University

JTextField is user editable block of text useful for collecting user input

- Simple constructors:

```
JTextField();            //Initially empty
JTextField(String s);    //Created with given text
```

- Accessor and mutator methods provided to inspect and change the text displayed:

```
String getText();            // Retrieve the text that has been typed in
void setText(String s);      // Set the text box to the given value
void setEditable(boolean);   // Enables / disables user input
```

Swing example

Lancaster University

```
import javax.swing.*;

public class HelloWorld
{
    public static void main(String[] args)
    {
        JFrame a = new JFrame();                          // Create a blank window
        JPanel panel = new JPanel();                      // Create a panel
        JButton b = new JButton("Press!");                // Create a button
        panel.add(b);                                     // Add button to the panel
        a.setContentPane(panel);                          // Use panel on Window
        a.setTitle("Hello world!");                       // Change window title
        a.setSize(300, 300);                              // Change window size
        a.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        a.setVisible(true);                               // Make it visible
    }
}
```

Lancaster University

Is our program object oriented?

Becoming an awesome builder…

- What we have actually done is:
  - Created software that's made up of other objects
  - Imagination is vital. You need to learn to see the world as a set of components. Use **composition** to create new things!

A better example…

Lancaster University

```
import javax.swing.*;

public class HelloWorld
{
    private JFrame a = new JFrame();                    // Create a blank window
    private JPanel panel = new JPanel();                // Create a panel
    private JButton b = new JButton("Press!");          // Create a button

    public HelloWorld()
    {
        panel.add(b);                                   // Add button to the panel
        a.setContentPane(panel);                        // Use panel on Window
        a.setTitle("Hello world!");                     // Change window title
        a.setSize(300, 300);                            // Change window size
        a.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        a.setVisible(true);                             // Make it visible
    }
}
```

A better example….

Lancaster University

```
public class Driver
{
    public static void main(String[] args)
    {
        HelloWorld h = new HelloWorld();
    }
}
```

A better example…..

Lancaster University

```
public class Driver
{
    public static void main(String[] args)
    {
        HelloWorld h = new HelloWorld();
        HelloWorld w = new HelloWorld();
    }
}
```

Summary

Lancaster University

- Today we learned…
  - About a real-world example of OO software
  - How to build very simple GUIs in Java
  - How the Java API documentation contains all the details you need to know for other GUI components

    https://docs.oracle.com/en/java/javase/23/docs/api/