Lesson 11: GUIs with Swing

* A listener object performs some action in response to an event
* A simple window in known as a JFrame, it has buttons to minimize and maximize and close

**JFrames**

* The JFrame constructor has an optional sting argument that will be used as the title

Creating a JFrame object

JFrame firstWindow = new JFrame();

* A JFrame can have components added to it, such as buttons, menus and text labels.

firstWindow.add(closeButton);

* It can be made visible using the setVisible method

firstWindow.setVisible(true);

* The JFrame can do four things on close (default is hide, doesn’t end the program)
  + DO\_NOTHING\_ON\_CLOSE don’t do anything
  + HIDE\_ON\_CLOSE hides the frame
  + DISPOSE\_ON\_CLOSE hides and disposes the frame
  + EXIT\_ON\_CLOSE exit the application using system.exit

firstWindow.setDefaultCloseOperation(

JFrame.DO\_NOTHING\_ON\_CLOSE);

Things you can do with JFrames

* setSize
* setTitle
* add – appends the specified component to the JFrame
* dispose – release all of the native screen resource used by this window
* setLayout
* setMenuBar
* set a default close operation

**Buttons**

Creating a button

JButton closeButton = new JButton("Press Me");

firstWindow.add(closeButton);

* clicking a button fires an event which is sent to a listener
* A GUI must therefore specify listener objects

ButtonCloseListener button = new ButtonCloseListener();

closeButton.addActionListener(button);

**Ending a Swing program**

* GUIs will run in an infinite loop
* To end a GUI program use System.exit(0)

**Labels**

* A label is an object of the class JLabel which can be used to add text to a JFrame.

**Color**

* Notice the American spelling
* A JFrame cannot be colored directly, instead a program must color something called the *content pane* of the JFrame.

Setting the background colour of the JFrame

getContentPane().setBackground(Color);

* Color constants like Color.BLACK can be used

**Layout managers**

* The add method adds objects to the JFrame but does not specify how they will be displayed, that is what layout managers are used for.
* Three main types:
  + BorderLayout
  + FlowLayout (default)
  + GridLayout

setLayout(new BorderLayout());

add(label1, BorderLayout.NORTH);

BorderLayout:

* Places the objects into five main regions
  + North
  + East
  + West
  + South
  + Center

FlowLayout:

* Arranges objects one after the other left to right, with no location specified

GridLayout:

* Makes a layout with a specified number of rows and columns

setLayout(new GridLayout(rows, columns));

* Add method only has one parameter, the location is not specified, and objects are places left to right. The top row is filled first, grid positions cannot be skipped.

**Panels**

* A panel is a container for objects
* It can have its own layout manager
* Stored inside a JFrame under JPanel

**Menus**

* A menu is an object of the class JMenu. A choice on a menu is called a menu item and is an object of the class JMenuItem
* A menu can contain any number of items

**Text fields**

* **­**the JTextField object displays a field that user the user to enter a single line of text.

JtextField name = new JTextField("Enter name here.", 30);

* Allows the user to enter 30 characters
* A JTextArea is the same thing but allows for multiple lines to be entered. It has an additional second parameter for the number of lines a user can enter and then is still followed by the number of allowed characters on each line.

theText.setLineWrap(true);

* This allows for text to wrap lines
* The number of characters per line is the number of **em spaces** which is the space used to hold the uppercase letter M, which is the widest letter in the alphabet
* Any text inputted or outputted on a GUI will need to be of type String