In this video, we are going to begin the implementation of the view class and the main application class for our simple calculator example.

0:00-0:08

In our calculator example, the view provides a graphical representation of our model of the calculator. The model of a simple calculator might include features such as the value entered by the user, the operation selected by the user, and the calculated value.

0:08-0:29

The view is what we call a top-level window. A top-level window is simply a window that does not appear inside another window.

It has some controls to minimize, maximize, and close the window.

It has some text that displays the title of the window.

It also has a menu bar from which the user can select various menu items. Implementing all of these features ourselves would be quite difficult.

Fortunately, the class JFrame provides all of these features. By extending JFrame, our view class can inherit these features.

Here we see the inheritance hierarchy describing our view class. We can see that the hierarchy consists of many classes.

Lets focus on the view, JFrame, and Frame classes. We see that the JFrame class has a JMenuBar and it provides public methods to get and set the menu bar.

The Frame class has a string that describes the title of the frame, and it provides public methods to get and set the title.

We do not need attributes for the menubar or title in our view class because our view class inherits these features from JFrame and Frame.

1:13-2:04

ECLIPSE:
We are now ready to begin implementation of our View class. We'll create a new View class that extends JFrame.

We start by creating a default constructor.

Recall that the first line of every constructor is always a call to the super class constructor, or another constructor of View. In this case, we'll invoke the superclass constructor passing in the title of the frame.

2:04-2:51

ECLIPSE:

We can now create or Main application class. Our Main class has a main method that creates a new view.

When we run our Main application, the view does not seem to appear, but the program did run because we can see in the console that the program is terminated.

2:51-3:25

ECLIPSE

To make the view appear, we need to set its visibility to true using the method setVisible

Now when we run our Main application, the view appears, but it is very small. This is because its default size is 0 by 0 pixels.

3:25-3:43

ECLIPSE

So we need to set the size of the view using the method setSize, specifying the width and height of the frame in pixels. We'll set the size to 250 pixels and the height to 250 pixels.

When we run our Main application, the view now appears with the size that we have specified.

3:25-4:18

ECLIPSE

The final thing that we should do is to specify what happens when we click on the close button. The default behaviour from JFrame is to hide the window, but not release the system resources associated with the window. To actually close the window and exit the program, we use the method setDefaultCloseOperation and specify that we want to exit on close.

This concludes part 1 of the View and Main application video. In Part 2, we'll look at how to create the buttons, text field for the View.