CS 2300 Lab Week 13

Basic MVC GUI – Part 5

Color Panel

1. Do some preliminary reading on topics for today's lab on the `Color` class and the `JColorChooser` component.

2. Continue on with the project from the last lab. This should closely parallel the classroom development.

COLOR PANEL PART

Next we will put in a more interesting control panel

7. Go to the `ColorPanel` class. Create a field `JLabel jlColor` with the text `Set Color:` on it with the following statement:

```
private final JLabel jlColor = new JLabel("Set Color");
```

Fix imports, and do this whenever needed below.

8. Also create a field `JButton jbColor` with the text `Change` on it, like this:

```
private final JButton jbColor = new JButton("Change");
```

9. Next create a c-tor for the class. The first line should be: `super(model)`.

10. Add an `ActionListener` to `jbColor` to the c-tor, like this:

```
jbColor.addActionListener(ae -> changeColor()).
```

This will cause an error, because we have not yet created the `changeColor` method. We will get to that shortly. NetBeans will offer to translate this "new style" statement with an "old style" version. Don't do it.

10.1. At the bottom of the c-tor, add the lines:

```
add(jlColor);

add(jbColor);
```

These lines "add" the label and the button to the Color Panel, so that they show up. Run the program to see that this is the case.

10.2. Action Listeners are the key to making GUI components work. Here is what is happening. We are telling the button `jbColor` what to do if it gets clicked on. Namely, call the method `changeColor`. Next we will write this method to deal with changing color.

11. Below the c-tor, write a method with signature `private void changeColor(){}`.

12. This method has only two lines. Before we write the first one (which is long and involved), we will use a temporary simple version (do one little thing at a time!) . Write the two lines:

```
model.setColor(Color.GREEN);

model.getView().repaint();
```

This will cause the color green to be used in a repainting. Run the program. When you click the button, it will change the circle to green. This is just a temporary step to be sure things are coded right.

Here is what `model.getView().repaint();` is doing. How can we tell the view panel to repaint itself? By telling the model to get the address of the view, and call its repaint method. That is why we need the `getView` method.

13. New replace `Color.GREEN` with `JColorChooser.showDialog(a,b,c).` Here, `a,b,c`, are three arguments that will be replaced with the correct values in a minute.

14. The first argument, `a`, can be `this`, or a reference to some component on the screen.

15. The second argument, `b`, is the text that will be displayed to the user. Something like "`Choose new color`" will be good.

16. The third argument, `c` , is the color to be used in the demo parts of the `JColorChooser`. A good choice is the current color of the ball. To get this color, replace `c` with: `model.getColor();`.

17. In situations like this, where several involved arguments are sent to a method, it is sometimes a good idea to write it out like this:

```
model.setColor(JColorChooser
            .showDialog(
                  this,
                  "Choose new color",
                  model.getColor()));
```

This "one parameter per line" style makes it easy to read if you have to do some serious thinking. On the other hand, it takes a lot of space. If you are sure what you are doing, or have a lot of similar lines, you can write it something like this:

```
model.setColor(JColorChooser.showDialog(this, "Choose new color",
      model.getColor()));
```

There is too much code for one line, so it will extend too far to the right. Thus, pick a good place to start a new line (usually after a comma). Always indent at least one tab if you are continuing a line. Try to break the line at a good place, such as after a comma, or before a dot. Note you CANNOT break a line inside a string, such as "`Choose new color`". If you have a really long string, and need to break it, do it like this: "`Choose new " + "color`", and go to a new line after the `+`.

The style of using a lot of lines (as above) is rather new (I have been seeing it now for a couple of years) and I think it is a good idea, and here to stay. It is all about making it easy for the programmer to get right. And, the programmer is YOU.

18. Here is what is happening. When the button gets clicked, up pops the built in `JColorChooser` component. This is as highly sophisticated component that will return whatever color the user selects. We will discuss this component in class.

19. Run the program. Fix if needed. Try the various ways of choosing a color. Say “Wow! This is cool!”.

20. Show the lab instructor that you can change the color.