5004 Event Fill

Part 1 : Understand the existing code.

Code: https://www.dropbox.com/s/7y7vxsqzsdsqyzb/StudentCode.zip?dl=0

Student Key: https://www.dropbox.com/s/hu9m4hj0cvix95l/StudentKey.zip?dl=0

Objective 1 : Examine each file. What is the purpose of each?

* Driver

This file tests and compares two different controller-view pairs, one using a non-event based controller and the other using an event-based controller. This creates two view objects, one for the command-line view (CLView) and another for the graphical view (GUIView).

* Controller

This defines a controller interface, establishing polymorphism by providing a common method signature for different types of controllers. The 'go' method is declared in the interface, and any class implementing this Controller interface must then provide an implementation for the ‘go’ method.

* EventController

This file implements the Controller interface, representing an event-based controller interacting with GUIView and FillModel. This sets the model and view, connecting the event controller to the view, and provides a method to display the GUI window.

* NonEventController

This also implements the controller interface, representing a non-event-based controller. It initializes a view and model, as well as a Scanner to receive user input. The ‘go’ method contains a loop which will continuously ask the user to enter coordinates, updating the view, and finally seeing if the user wants to quit the program or continue. This is an example of MVC design.

* CLView
* IFillAlgo
* FillAlgo
* FillModel
* IView
* GUIView
* ICLView
* Screens

What’s the difference between the event and non-event controller?

Which is synchronous and which is asynchronous?

Objective 2 : Run the command line version of the application

1. What is this application doing?
2. Could you change out the fill algorithm?
3. How does the model represent different colors?
4. How else could the command line version be implemented?

Objective 3 : Run the GUI version

1. Why are there two different controllers needed?
2. How else could the GUI be implemented?
3. What makes this event driven programming?
4. Is this MVC?
5. How is the callback to the controller enabled so that it controls the user input?

Part 2 : Make some changes

(The only files you’ll need to change are the GUIView and the Controller. I left the rest of the functions alone.)

Objective 4 : Reset the image

1. Watch as the instructor demonstrates Windows Builder.
2. Add a reset button that returns the map back to the original state.

        Tip: You can’t just clone a 2d array. You have to clone each individual array.

             for (int count = 0; count < screen.length; ++count)

                                        reset[count] = screen[count].clone();

Objective 5 : Create a random image

1. Create a randomizer button.
2. Connect that button to the controller and have it create a random screen of dots

        Tips: I left some of the functions in place to do this with.

Objective 6 : Adding a timer object (NOT IN KEY)

1. Add a timer that counts down from 10
2. At the end of 10 seconds if resets the screen again with a random set of dots

Ok I swear I didn’t mean to create a game, but it happened. If you want to take it further as a bonus challenge, have fun. Create a stop button, add a score keeper, increase points if all the dots are green at the end of 10 seconds, decrease if not, speed up the timer each round, etc…

Here are some hints, but I did not include this one in the key. I’ll leave it to you to implement it as an ICE. Remember ICEs are good faith effort attempts.

Hint: Here’s how to create a timer listener…

Timer timer = new Timer(1000, new ActionListener() {

                        public void actionPerformed(ActionEvent evt) {

                                //do stuff

                        }});

Then somewhere you have to start the timer: timer.start();

                JLabel lblCountDown = new JLabel("10");

                lblCountDown.setBounds(24, 115, 45, 13);

                getContentPane().add(lblCountDown);

If you want to remember something, you have to put it as an instance variable. Like a countdown timer.

JLabels would make an easy score display. Look up the function that lets you change the text.

ICE Submission:

Complete objective 6

Make one additional improvement

Describe one design improvement that could be made (There’s plenty)

Describe one error you found (I’m sure there’s plenty)