CSCU9A2 Practical 3B (Week 4) Spring 2017
Graphical User Interfaces 2 (9 February)

Remember to register your practical attendance at the start of each session.

If you get stuck or need help at any time during the practical, ask a demonstrator.

Remember: This week's checkpoints must be completed by the end of week 6.

THIS WORKSHEET:

This worksheet is based on the WindowBlind example discussed in lectures. For the checkpoint you will experiment with the event handling associated with a JSlider and then extend the program with an extra window blind. Then you will extend the application with a second slider to control the second blind separately.

JSLIDER DOCUMENTATION

Remember that information about Swing **JSlider** widgets can be found on the Oracle Java web site: In BlueJ select **Help** menu/**Java class libraries**, wait for the browser window to appear, scroll down the lower left frame to find **JSlider** and click on it — the documentation will appear in the right hand frame.

SLIDERS: THE WINDOWBLIND PROGRAM

At this step you will build a new project with a main program class that displays a simple empty frame, and then experiment with adjusting its settings. Finally you will extend it by adding three text fields to display two numbers and their sum.

➢ First you need to create a new basic application from an existing file as you have done before:

- Create a new **WindowBlind** folder in your CSCU9A2 working folder. Copy the file **WindowBlind.java** from **Groups on Wide (V:)\CSCU9A2\Java** to your **WindowBlind** folder.
- Launch BlueJ. Using the **Project** menu, **Open Non BlueJ**... create a BlueJ project in your **WindowBlind** folder – remember, when you navigate to it click once on the folder name then **Open in BlueJ** – do not double click to open the folder.
- Compile and run the program.

➢ This an example program from the early editions of Java for Students. It draws a picture of a window (just a rectangular frame), and a blind is moved up and down the window as a slider is adjusted. The blind is drawn by a `fillRect`, with the height of the filled rectangle being determined by the slider's value. All the drawing is carried out in the `paintScreen` method, which is called automatically when a screen refresh occurs.

Examine the program, and note its structure. It has a `stateChanged` method, where interactive events are handled, and `paintScreen` method that actually draws on the graphics panel. It also has the usual two other methods, `main` and `createGUI`, which launch the program and set up the graphical user interface.

Here are some experiments for you to try out. Compile and run after each change:

- Change the slider so that it is vertical (guess how!). Choose which orientation you prefer.

- Draw the window *outline* in black and the *adjustable blind* in a *different colour*. You may need to exchange the order of the `drawRect` and the `fillRect` for the best picture (with the black frame always visible) — try both orders.
- Try altering the *initial setting* of the slider when it is constructed in `createGUI` (the fourth parameter of the `new JSlider` constructor): for example, 0 or 100. Look at the effect in each case. [Change back to 50 afterwards.]
- Try altering the *range* of values for the slider when it is constructed in `createGUI` (the second and third parameters of the `new JSlider` constructor): for example, 20–80 or 0–150 — in each case you will also need to choose an appropriate *initial* setting for the slider. Look at the effect in each case. [Change back to 0–100 afterwards.]

➢ Now you are going to create a new method `drawWindow` to draw a window frame and its blind:

- The method header line should be:

  ```
  private void drawWindow(Graphics g, Color c,
                          int x, int y, int level) {
  ```

  The parameters are going to be used as follows: `g` is the usual graphics area information, `c` gives the colour to be used for the blind (`Color` is a *type* name — you can also have variables of this type), `x` and `y` are the position where the top left corner should be drawn, and `level` gives the value indicating how far down the blind is to be filled.
- You can obtain the program statements for the *body* of `drawWindow` by *moving* the code from the body of `paintScreen` into the body of `drawWindow`.
- To make sure that the method's parameters are used correctly by the method body, you must make minor edits to introduce the parameter names where appropriate: `x` should be used in place of `120`; `y` should be used in place of `80`; `level` should be used instead of `blindHeight` for drawing the blind rectangle; and the parameter `c` should be used in the `g.setColor` instruction just before the filled blind rectangle is drawn instead of an explicit colour.
- Also add a statement the body of `paintScreen` to call `drawWindow` instead of drawing the window explicitly. You will need a call like this:

  ```
  drawWindow(g, Color.red, 120, 80, blindHeight);
  ```

  where the appropriate information for the window and blind to be drawn are given as *actual parameters*.

  The output should look just the same as before.

➢ Now, by adding a second call of your `drawWindow` method, draw a second window on the screen, controlled by the same slider, ***but with the second blind drawn in a different colour, and at a different position on the screen***. The two blinds should move up and down together as you adjust the slider — compile and make sure that the program runs properly.

Remember: Your code must be well formatted and clearly commented.

Checkpoint [GUIs 2]: Now show a demonstrator your program with the new method for drawing your window blind, and the two calls of the method in paintScreen, and demonstrate it running. Answer any questions they ask you.

ADDING A SECOND SLIDER

In this exercise, you will enable the two blinds to be controlled separately:

- Add a *new slider* to control the *second* window blind, and alter `paintScreen` to draw the two windows using one slider to control each. Here is how to do it:

  You will have to declare a new global variable at this step for the new slider itself, say `slider2`.
  You will have to set up the slider in `createGUI`: creating it with appropriate constructor parameters, adding it to the display, and registering the program as responding to its adjustment events.

  Alter `paintScreen` so that the second window blind is controlled by the *second slider*.

  You can make these changes by carefully copying, pasting and editing the code already present — be careful to make all the necessary edits to the pasted code!

- Introduce some `JLabel` widgets to identify the different sliders. You might need to widen the program window by changing the panel and frame sizes.

- Add diagnostic statements (using `System.out.println`) *just at the start* and *just at the end* of **each** method body, displaying messages like "Entering stateChanged" and "Returning from stateChanged".

  When you run the program you will now see a complete summary of the "dynamic" path of execution through all of the methods, displayed in the BlueJ terminal window.

  Remember this trick. It's very helpful when you are developing your own programs, to give you some idea of what is happening at runtime (especially if your program isn't working properly!)

Remember: Your code must be well formatted and clearly commented.

That's all for this worksheet.

SBJ 3 February 2017