**University of Stirling**
**Computing Science and Mathematics**

## CSCU9P6 Software Engineering II
## Practical: week starting 21 January 2019

# Developing Object-Oriented Programs in Java using Together Architect

### Programming in Java with Together

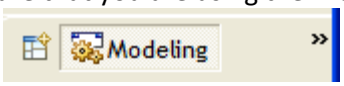
In this practical you will see how Together can be used to support Java application development, using an old, familiar (?), interactive Swing application: the `WindowBlind` program. It is also an opportunity for you to recall how to build GUI based Java applications using Swing.

1. Open your T: drive (which is file storage on our server wsv). Somewhere convenient create a new folder for a Java modelling project, call it `WindowBlind`. Inside it create a new folder called `src`. [Standard Eclipse projects have a `src` folder for source code and a `bin` folder for compiled code, and Together is based on Eclipse.]

2. Open the folder `CSCU9P6` in `Courses` on the Divisional file server (`Courses` is normally already mapped to drive K: in My Computer).

3. Open `Practicals\WindowBlind` and copy the ***file*** `WindowBlind.java` to the src folder that you created at step 1. It is a simple Java application – just a single Java file. The Java file in `WindowBlind` will probably have had its "read only" attribute set during the copy (because it came from a read-only source), so right-mouse-click on its icon and select **Properties** from the pop-up menu (or select it and choose **File** menu, **Properties**), un-set the read-only attribute and close the dialogue.

4. Launch Together Architect via the **Start** icon and either type **Together** into the search box, and click on the **Together** that is found, or navigate via **All Programs** then **Borland Together** and then click on **Together**, and *follow the instructions below* to create a new modelling project ***in your*** **`WindowBlind` folder** – the project can be called whatever you like, *but it must be in that folder*:

   - When you launch Together, make sure that you are using the **Modelling perspective**: Near the top right of the window you should see this: 

If you don't then click the icon  or the small arrows >> and choose Modelling from the offered list of perspectives.

   - If Together has opened any previous projects automatically (they look like this  or  in the **Model Navigator** pane) then close them like this: Select an opened project name in the **Model Navigator** pane to the left and then **Project** menu, **Close Project** – the icon should change to  . Open projects consume resources as Together watches them for changes: properly closing them improves performance.

   - Select **File** menu, **New** followed by **Project**. This gives the **New Project** dialogue in which you **Select a wizard.** Expand the **Modelling** section and in the expanded list choose **Java Modelling Project** and then click the **Next** button. You see the **New Java Modelling Project** dialogue.

   - Click **Create project from existing source.** Then click the **Browse** button to the right of the **Directory** field – a folder browse dialogue appears.

   - Navigate to your `WindowBlind` **folder**, click once on the **folder** name and then on **OK** – the path to the folder should now appear in the **Directory** field.

   - Type an appropriate new project name into the **Project name** field.

   - Click on **Finish**.

   Together should analyse the Java file present in the folder and show you the class diagram that it has synthesized – a pretty simple one in this case! (If you forgot to remove the read-only setting above, then Together may show a small padlock symbol in various places, and will not allow you to make changes. You can just switch to the desktop and change the setting, and Together will note the change automatically when you switch back – **you do not need to close and re-launch Together!**)

5. The Together window has probably opened up with its **Model Navigator** pane to the left (which displays a project explorer **–** click **+**s or small triangles to expand sections, double-click items to select them for display in the other panes), with the class diagram to the right, and with a **Properties** pane below that. A program editor may appear later in a lower right pane. The boundaries between the panes can be dragged to resize them, the panes can be dragged to different positions within the window, and the **Window** menu, **View** options can used to hide and show the various panes. If the various panes get in a mess, then you can easily reset to a standard arrangement using **Window** menu, **Reset Perspective**.

6. Now you should be able to compile and run the program. The easiest way is to: First double-click on the `WindowBlind` class in the class diagram (or right-click on the `WindowBlind` class in the Model Navigator and choose **Open**) – this opens a Java editor pane for the class. Then choose the **Run** menu, **Run** option or click the toolbar button: .

   [Sometimes Together might prompt you for information to set up a "Run configuration": In the Run Configurations dialogue, click once on **Java Application** in the explorer-style list on the left, then on the **New** button at the top . In the configuration pane that appears: enter a name for the configuration (e.g WindowBlind), next to the Project field click **Browse**, select the WindowBlind project, then **OK**, next to the Main class field click **Search,** select the main class (again WindowBlind) then **OK**, and finally click **Run**. Together remembers the configuration for subsequent runs: either choose **Run** menu, **Run Last Launched**, or click the toolbar button: .]

7. In the Together class diagram, double-click on the `WindowBlind` class and inspect the code in the Editor pane. The following notes describe the Java event handling for the slider in this application:

   - `JSliders` generate "`ChangeEvent`"s when they are adjusted, and the `WindowBlind` class (an extension of `JFrame`) promises that it "`implements ChangeListener`", where "`ChangeListener`" is a Java *interface*
   - `WindowBlind` satisfies the requirements of the `ChangeListener` *interface* by supplying the one required event handling method:
     ```
     public void stateChanged(ChangeEvent e)
     ```
   - Objects which implement an event listener interface are *not* automatically informed of the relevant events unless they "register" themselves with the object generating the event as a "listener" for that event, so where the slider is set up in the `WindowBlind` constructor there is the statement
     ```
     slider.addChangeListener(this);
     ```
   - Once everything is set up, when the user adjusts the slider, the Java VM informs the `JSlider` object itself, which automatically invokes the `stateChanged` method of each registered listener.
   - Not represented in this code is that the `ChangeEvent` object (and similar parameters of all other event handler methods) contains the identity of the widget generating the event: this is extracted by the expression `e.getSource()`
     and so in an event handler, if you need to find out which widget activated the event so that appropriate action can betaken, code like the following is often required:
     ```
     if (e.getSource() == slider)
         do something appropriate …
     ```

8. In the Editor pane, Together shows small **–** symbols in the left margin to allow you to collapse sections of code (hiding them, not deleting them). Expand again using the **+** symbols. This can be useful when dealing with large classes. Other symbols may appear to the left of the Editor pane, indicating warnings or errors.

9. Now an exercise (see next page): To carry out the exercise **you should try to exploit Together's *"round trip engineering" features*** – that means that many alterations can be made *either* in the class diagram, *or* in the program editor, *or* in the **Properties** pane, and those changes will *automatically* be reflected in the other places. **Watch carefully what happens.** The **Properties** pane may be rather small: it can be enlarged by dragging the boundaries, or it can be undocked and resized separately by right-clicking on its title bar and choosing **Detached**.

**Read all the steps on this page before starting:**

- You are given the information that, to react to **button** click events:

  1. You must use Swing `JButton` widgets, which are created like this:
     `myButton = new JButton("text for the label");`
  2. The program must `implement` the `ActionListener` interface,
  3. The `ActionListener` interface requires the following event handling method to be present
     `public void actionPerformed(ActionEvent e)`
     and
  4. An object registers itself with a button b to be notified of click events using
     `b.addActionListener(this);`

- **Following the instructions given below**, you are required to **add three JButtons** to the `WindowBlind` application: one which opens the blind completely, one which closes it completely, and one which quits the application.

  1. *In the **Class diagram*** add three attributes to the class to hold the `JButton`s (they should appear as global variable declarations in the code in the Editor pane automatically). *Sensible identifiers, please!*
  2. *In the **Editor pane*** add statements to the `WindowBlind` constructor to create and set up the three buttons (in a similar way to the slider set up that is already there).
  3. *Use the **Properties** pane* to add `ActionListener` to the `implements` list for the class: select the class on the class diagram, find the **implements** property in the Properties pane list and add `ActionListener` – just type it after `ChangeListener` that is already there, preceded by a comma

     (there is a library browser if you click on the small button  that appears to the right when you select the **implements** property, but you need to know the library structure in order to use it). `ActionListener` should appear in the class diagram and the code automatically.
  4. *In the **Class diagram*** add the `actionPerformed` method to the class. **Or you could take advantage of Together's underlying Eclipse's advanced Java development features**: If you look at the ***Editor pane***, an error is now flagged up on the `public class WindowBlind` line: There is a small red cross to the left because the "`implements ActionListener`" **requires** the presence of an `actionPerformed` method. *There is also a small yellow lightbulb* – this indicates that Eclipse can offer help fixing the problem: Click once on the lightbulb and some suggestions will pop up. Double-click on the **Add unimplemented methods** suggestion to add an `actionPerformed` method, and after a short while the job is done for you! Eclipse understands Java **quite well**!
  5. Fill some suitable action into the body of the `actionPerformed` method *in the **program editor pane***, and any other details that need adding. Remember: use `if` statements to check which `JButton` has been clicked, use `repaint();` to force screen update, and exit the program using the statement `System.exit(0);`
  6. Of course, make sure that it compiles and runs properly!

**Checkpoint:** Now show your modified WindowBlind application to a demonstrator: Show them it working and explain how clicking the application's buttons causes the changes that are visible on the screen.

- *In the **Properties** pane Description/comment* property add some *comments* to one or more of the methods, and to the class, and see how they get added into the actual code in the program editor pane.

That's all

SBJ January 2019