EXERCISES

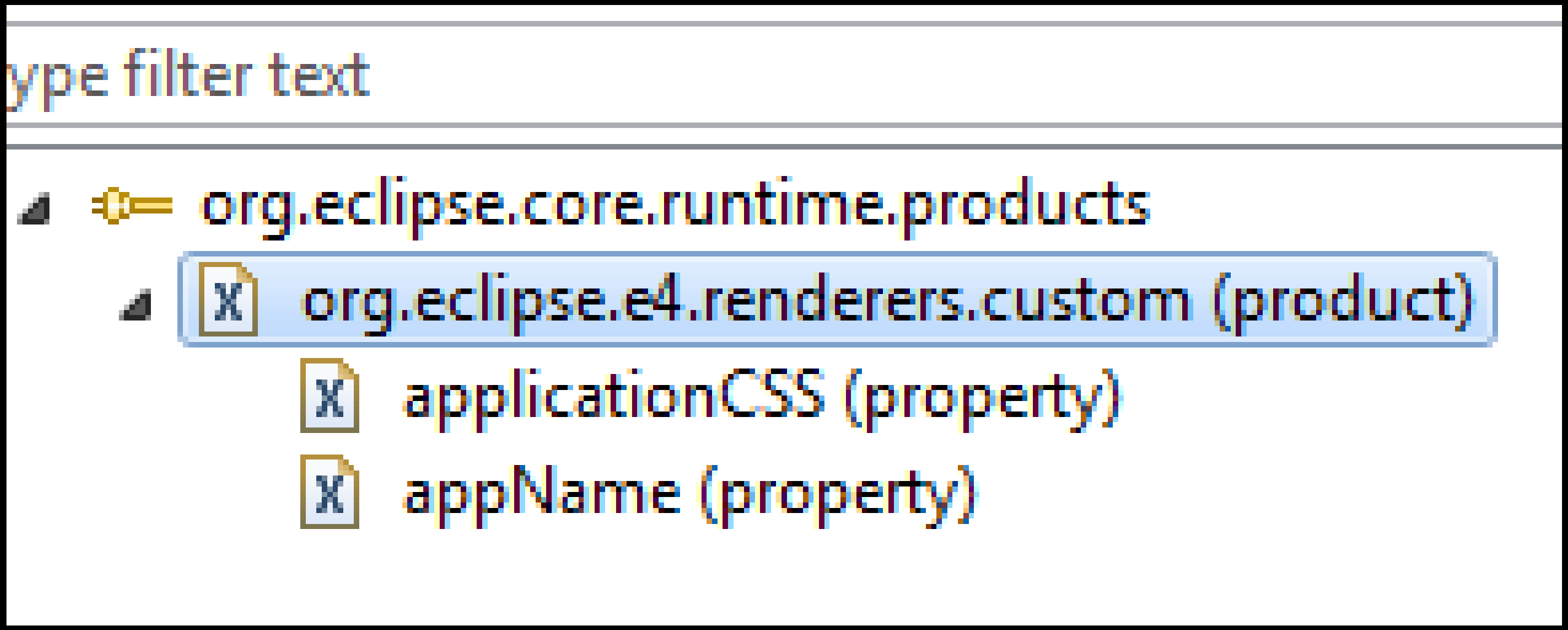
3 exercises for custom SWT renderers

1

Define custom renderer factory

- We will reuse most of the rendering infrastructure and just add our custom behavior

- Add a ‘rendererFactoryUri’ property in plugin.xml pointing to the CustomRendererFactory class in the pattern ‘bundleclass:// projectId / fqn_of_class’



```
ype filter text
org.eclipse.core.runtime.products
  org.eclipse.e4.renderers.custom (product)
    applicationCSS (property)
    appName (property)
```

Produce renderers from your factory

- Override the 'getRenderer' method of the CustomRendererFactory class to produce a renderer for a specific model element, in this case MWindow

- Hint: If a renderer is requested for instances of MWindow, create, initialize and return CustomWindowRenderer , else return whatever default you would return (super.getRenderer)

A custom Window renderer

- Goal: have a custom window renderer which creates half-transparent windows (alpha = 128)
- Edit class CustomWindowRenderer
- Override it's createWidget method (which returns a Shell)

2

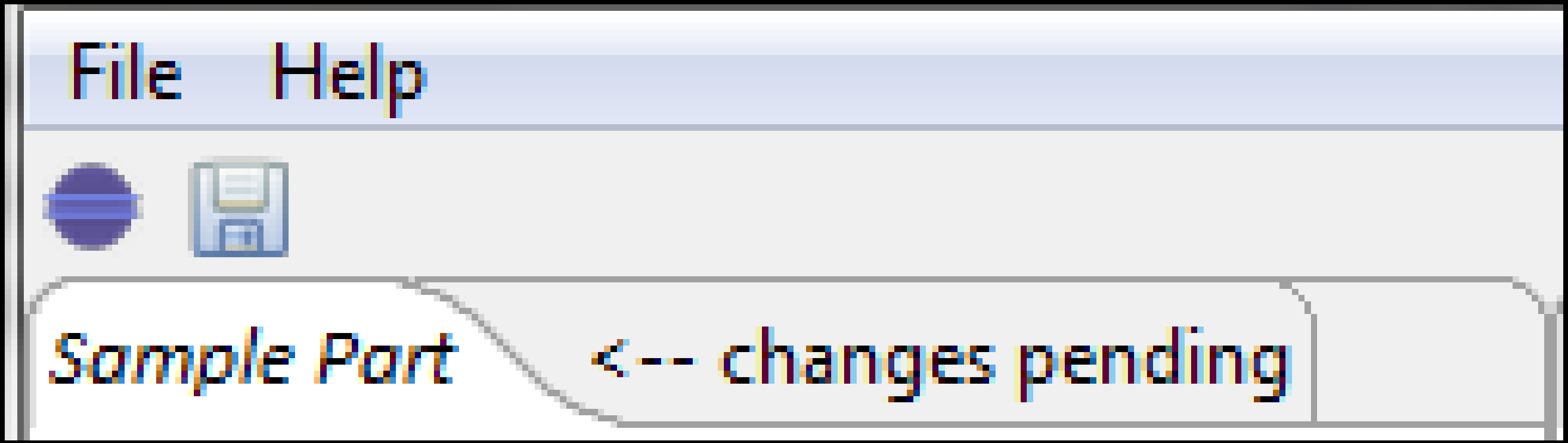
A custom Sash renderer

- Goal: create a custom renderer out of the existing SashRenderer to have locked layout (non-resizable)
- Hint 1: override createWidget (which returns a Composite)
- Hint 2: use FillLayout as layout
- Hint 3: use SWT style bits for layout orientation (FillLayout(type) constructor)

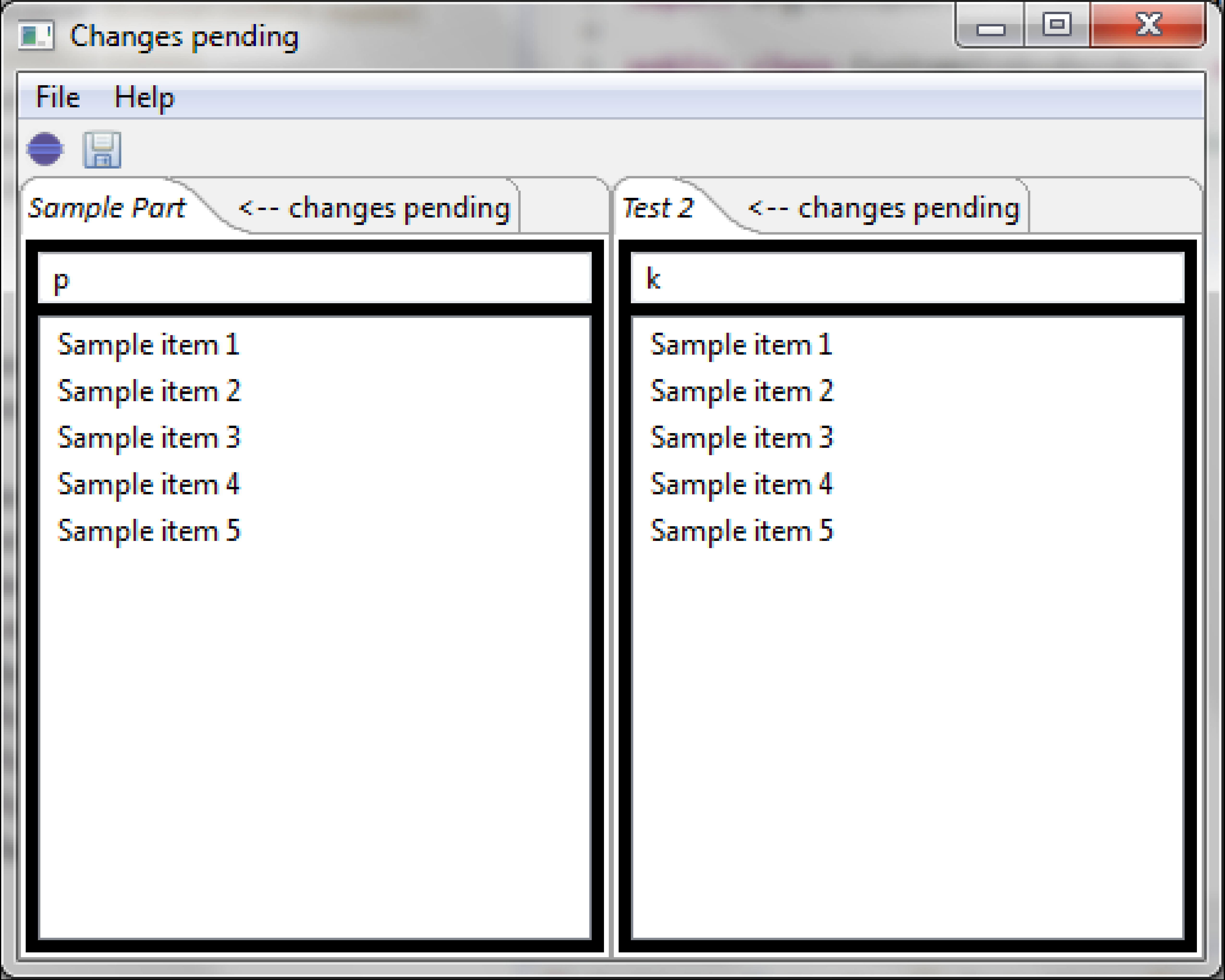
3

A custom Part Stack renderer

- Goal: have a custom 'dirty' indicator



- Hint: override updateTab method of StackRenderer
- Hint 2: Use Mpart#isDirty condition



p

Sample item 1
Sample item 2
Sample item 3
Sample item 4
Sample item 5

k

Sample item 1
Sample item 2
Sample item 3
Sample item 4
Sample item 5