Swing/GUI Cheat Sheet

General reminders

To display a Swing component, you must:

- Construct and initialize the component.
  Example: `button = new JButton ("ButtonLabel");`
- Add it to the content pane of the window or to a JPanel that is added to the display.
  Example: `getContentPane().add (button);`
- Import `javax.swing.*` and sometimes also `java.awt.*` at the beginning of the class creating the components.

To get events from a GUI component, you must do the following:

- Declare that the class handling the event implements the appropriate listener interface.
  Example: `implements ActionListener`
- Define the method that the listener interface requires.
  Example: `public void actionPerformed (ActionEvent event)`
- Add a listener appropriate for the component to the component.
  Example: `button.addActionListener (this);`
- Import `java.awt.event.*` (and occasionally `javax.swing.event.*`) at the beginning of the class that is the listener.

When the listener method is called, you can find out which component sent the event by calling `getSource()` on the event.

```
public void actionPerformed (ActionEvent event) {
    Object theButton = event.getSource();
    if (theButton == framedCircleButton) {
        // Create a framed circle
    }
}
```

If a method returns a `String`, remember to compare the result using the `equals` method, not `==`:

```
aMenu.getSelectedItem ().equals ("A value")
```

GUI Components

The following methods can be applied to any Component:

```
void setFont (Font f)
void setForeground (Color c)
void setBackground (Color c)
```

To construct a font use:

```
new Font (String name, int style, int size)
```

Find out the font names on the computer in Eclipse as follows:

- Open the Window menu
- Select Preferences
- click on the triangle next to Workbench
- Select Fonts
- Select Text Font to the right and then click on the Change... button.

Style can be one of `Font.BOLD`, `Font.ITALIC`, `Font.PLAIN`, or `Font.BOLD + Font.ITALIC`.

The specific components we have considered:

1. **JButton**
   Constructor:

   ```
   new JButton (String s)
   ```

   General Methods:

   ```
   String getText ( )
   void setText (String s)
   ```

   Listener Interface:

   ```
   ActionListener
   ```

   Adding the listener:

   ```
   void addActionListener (ActionListener al)
   ```

   Listening Method:

   ```
   void actionPerformed (ActionEvent e)
   ```

2. **JComboBox**
   Constructor and Initialization:

   ```
   new JComboBox ( )
   void addItem (Object item)
   ```

   General Methods:

   To find out which item was selected, use:

   ```
   Object getSelectedItem ( )
   ```

   If you wish to treat the value returned from this method as a String, you may use a *type cast*:

   ```
   String text = (String) menu.getSelectedItem ( );
   ```

   To find out the index of the item that was selected, use:

   ```
   int getSelectedIndex ( )
   ```

   Listener Interface:

   The JComboBox component is unusual in that it can hear about the user making a menu selection by either implementing ItemListener or ActionListener. Be sure to be consistent in your

choice of listener interface, method to add the listener, and listening method.

`ItemListener`

or

`ActionListener`

Adding the listener:

```
void addItemListener (ItemListener il)
```

or

```
void addActionListener (ActionListener al)
```

Listening Method:

```
void itemStateChanged (ItemEvent e)
```

or

```
void actionPerformed (ActionEvent e)
```

3. **JLabel**
   Constructors:

```
new JLabel (String s)

new JLabel (String s, int align)
```

align is one of `JLabel.RIGHT, JLabel.LEFT, JLabel.CENTER`

General Methods:

```
void setText (String s)

String getText ( )
```

Listener Interface:

no listeners available for JLabels

4. **JSlider**
   Constructor:

```
new JSlider (int orientation,
             int minimum, int maximum, int initialValue)
```

orientation is one of `JSlider.HORIZONTAL` *or* `JSlider.VERTICAL`

General Methods:

```
void setValue (int newVal)
```

To find out the current value, use:

```
int getValue ( )
```

Listener Interface:

`ChangeListener`

Adding the Listener:

```
addChangeListener (ChangeListener al)
```

Listening Method:

```
void stateChanged (ChangeEvent e)
```

5. **JTextField**
   Constructors:

```
new JTextField (String s)
```

General Methods:

```
void setText (String s)
```

To find out the value typed, use:

```
String getText ( )
```

Listener Interface:

```
ActionListener
```

Adding the Listener:

```
addActionListener (ActionListener al)
```

Listening Method:

```
void actionPerformed (ActionEvent e)
```

Containers

Both `JPanel` and the object obtained by sending `getContentPane()` to a `WindowController` object are containers (and have type `Container`). The following methods are available for all containers. To define the type of layout, use:

```
void setLayout (LayoutManager lm)
```

`LayoutManager` may be any of the layout managers listed below.

To add something to a container:

```
void add (Component c)
```

`Component` may be any `Component` (such as `JButton`, `JTextField`, `JSlider`, ...) or `Container` (such as `JPanel`). Use the method above if the container has a `FlowLayout` or `GridLayout`. Use the one below if it has a `BorderLayout`.

```
void add (Component c, int position)
```

The position may be any of `BorderLayout.NORTH`, `BorderLayout.SOUTH`, `BorderLayout.EAST`, `BorderLayout.WEST`, or `BorderLayout.CENTER`.

Constructing a `JPanel` is very straightforward:

```
new JPanel ()
```

Layout Managers

1. **BorderLayout** (*Default for WindowController*)

   Constructor:

   ```
   new BorderLayout ()
   ```

2. **FlowLayout** (*Default for JPanel*)

   Constructor:

   ```
   new FlowLayout ()
   ```

3. **GridLayout**

   Constructors:

   ```
   new GridLayout (int rows, int cols)
   ```

   ```
   new GridLayout (int rows, int cols, int colSpacing, int rowSpacing)
   ```
