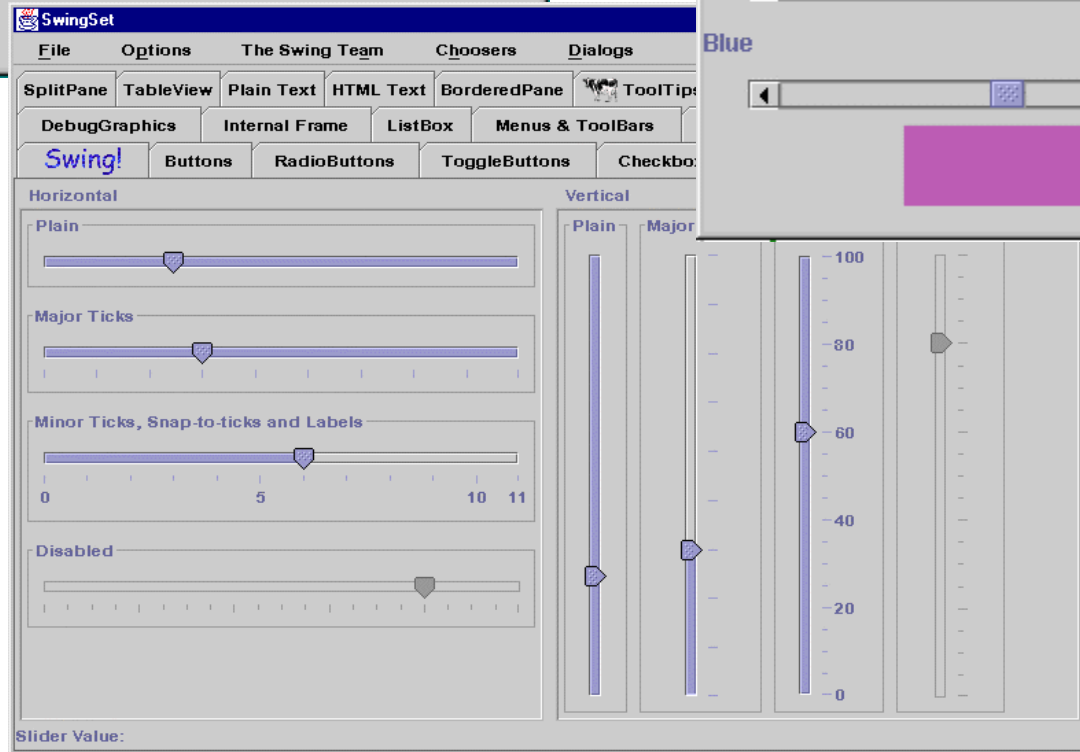
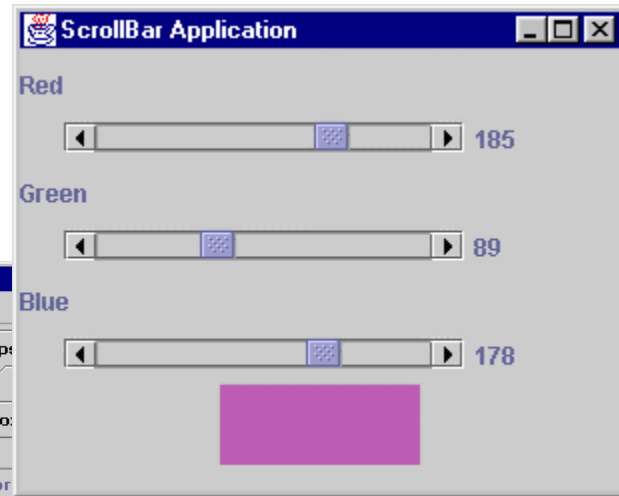
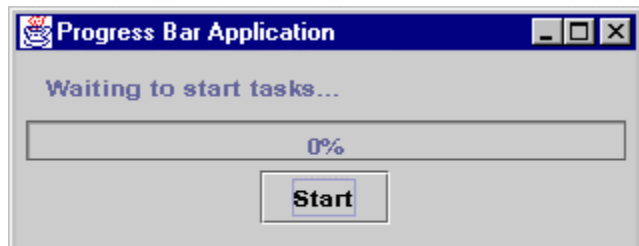


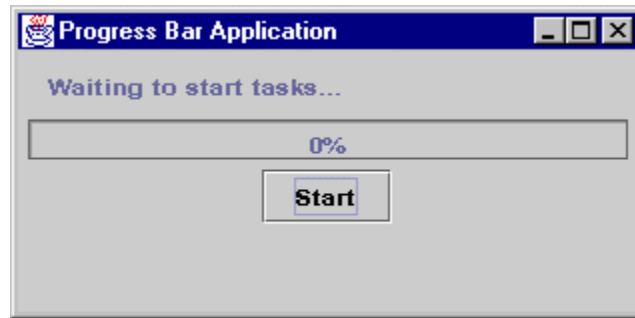
# Progress Bars, Scroll Bars and Sliders



# Progress Bars

- JProgressBar
- Feedback is essential to the user for any action which is not immediate.
- If duration of event is unknown, consider a “mock” progress bar which flows back and forth (similar to Netscape’s), or use JDK 1.4's JProgressBar which supports indeterimnate bars.
- JProgressBar presents the user with a bar graph
- Lets user know that program has not hung
  - `JProgressBar()`
  - `JProgressBar(int min, int max)`
  - `JProgressBar(int orient)`
  - `JProgressBar(int orient, int min, int max)`
- Methods to select painting of Border
- Ability to display a percent string in bar
- Supports change Listeners

# Code Example: MyProgressBar.java



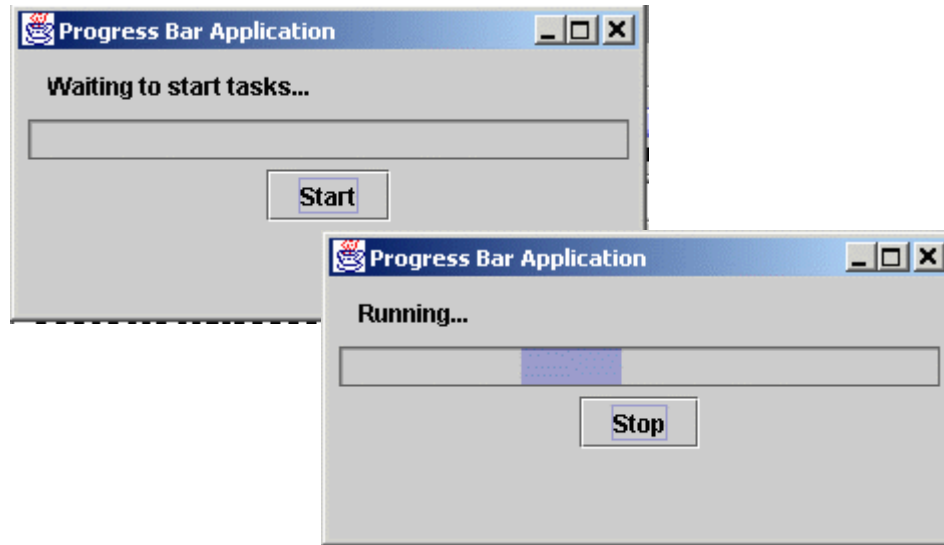
# Positioning Progress Bars

- If you noted in `MyProgressBar.java`, the progress bar was not simply “added” to the container. It used `setBounds()`.
- Also note that we used the default layout for the `JPanel`, which supports drawing objects at their preferred size
- Lesson is that progress bars are typically a layout intensive component that take a little more planning than some others.

# Indeterminate Progress Bars

- NEW in JDK 1.4
- By default, every progress bar comes up determinate, you set it to be indeterminate with
  - `setIndeterminate(true)`
  - This immediately starts bar into repaint cycle
- Control animation
  - Access UIManager properties
  - "ProgressBar.repaintInterval" in milliseconds
  - "ProgressBar.cycleTime"
    - must be an even multiple of the repaint interval
- Example
  - `UIManager.put("ProgressBar.repaintInterval", new Integer(250));`
  - `UIManager.put("ProgressBar.cycleTime", new Integer(6000));`

# Indeterminate.java



# Scroll Bars

- Used to find and view information that takes more space than the allotted display space
- Consists of an elongated rectangular container consisting of
  - Scroll area
  - Slider box
  - Arrows or anchors at either end
- May be horizontal or vertical
- Advantages
  - Permits viewing data of unlimited size
- Disadvantages
  - Consumes screen space
  - May be awkward to use
- Do not use to set values

# Scroll Bars in Java

- JScrollBar
- Bad reputation from AWT days, often used for setting values which it should not have been used for
- Usually used for interfaces which do not require a scale for the bar
  - If a scale is required, use JSlider instead
- Used in JScrollPane
- With new Swing components, use of JScrollBar by itself is assigned to specialized applications.
  - You will rarely use this by itself
  - May use it indirectly when customizing JScrollPane

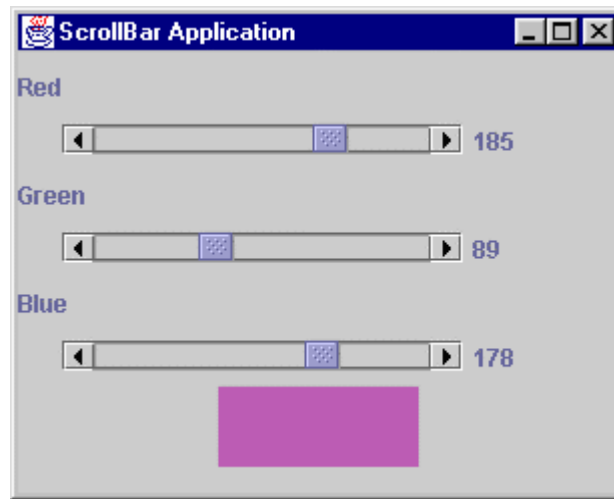
- Constructors

```
JScrollBar()
```

```
JScrollBar(int orientation)
```

```
JScrollBar(int orientation, int value, int extent,  
            int min, int max)
```

# Code Example, MyScrollBar.java



# Sliders

- Used to make a setting when a continuous qualitative adjustment is acceptable. It is useful to see the current value relative to the range of possible values
- Description
  - A scale exhibiting more or less of a quality on a continuum
  - Includes the following
    - A shaft or bar
    - A range of values
    - An arm indicating relative settings through its position on the shaft
  - May include:
    - A pair of buttons to permit incremental movement
    - A text box for typing and/or displaying an exact value
    - Detents or snap positions

# More on Sliders

- Advantages
  - Spatial representation of relative setting
  - Visually distinctive
- Disadvantages
  - Not as precise as an alphanumeric indication
  - Consumes screen space
  - Usually more complex than other controls
- Best usage
  - To set an attribute
  - When an object has a limited, continuous range of possible settings
  - When the graduations are relatively fine
  - When the choices can increase or decrease in some well-known, predictable, and easily understood way
  - When a spatial representation enhances comprehension and interpretation

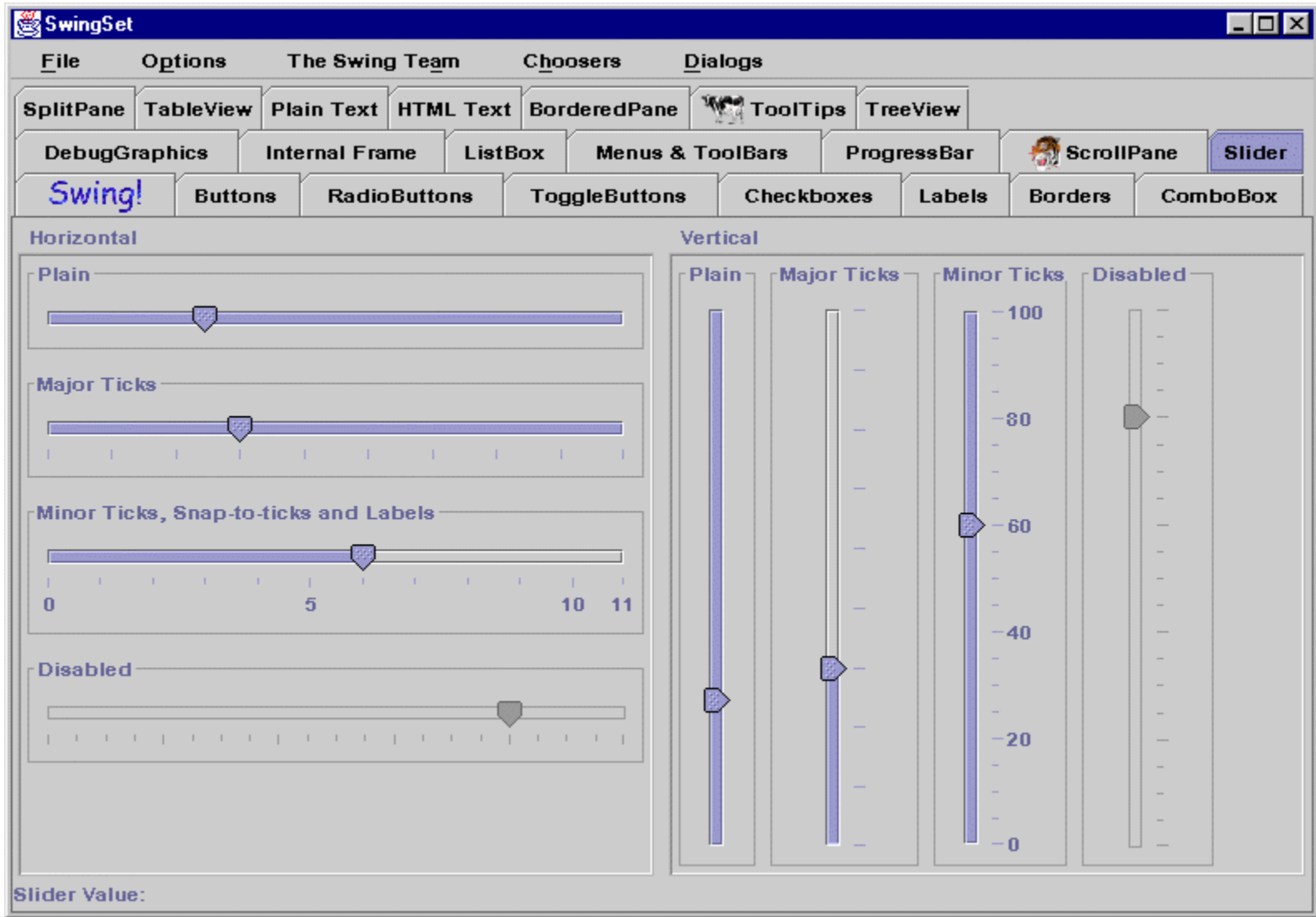
# Slider Scales

- Show the complete range of choices
- mark the low intermediate and high ends of the scale
- Provide scale interval markings, where possible
- Provide consistent increments
- Permit the user to change the units of measure
- If the precise value of a quantity represented is important, display the value set in an adjacent text box.

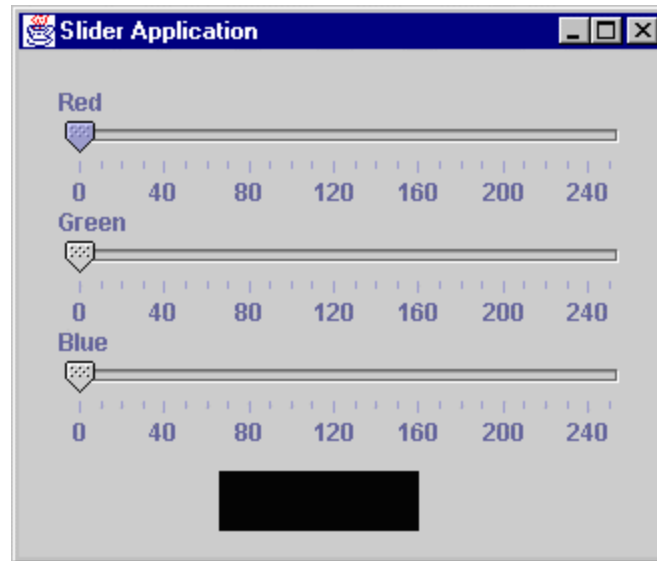
# Sliders in Java

- JSlider
- Previously, scroll bars were used for sliders
  - Didn't support precise positioning
  - Need some type of label to indicate value
- JSliders have improved:
  - appearance
  - support value scale
    - min, max
    - minor, major ticks
    - tick labels
    - snap to ticks
- Unfortunately, a JSlider is only a basic slider, it has no control buttons or TextFields associated with it.

# JSliders



# Code Example, MySlider.java



# JSliders, Attributes and Methods

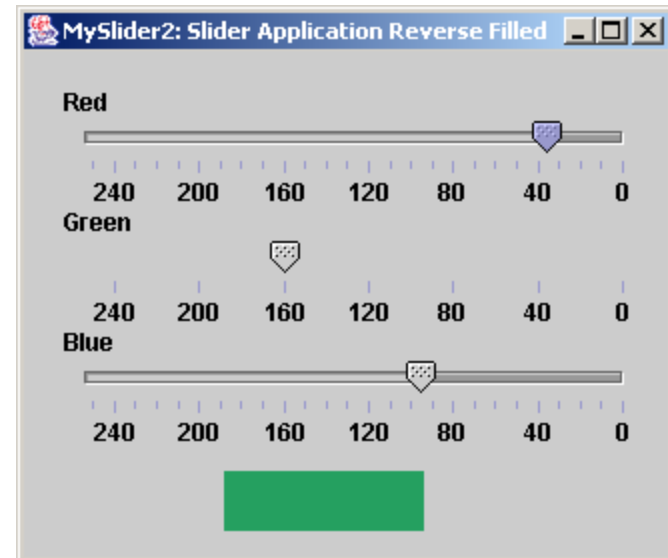
- **Constructors**
  - `JSlider()`
  - `JSlider(int min, int max)`
  - `JSlider(int min, int max, int value)`
  - `JSlider(int orientation, int min, int max, int value)`
  - `JSlider(BoundedRangeModel brm)`
- **get/set pairs for**
  - Value
  - Extent
  - Minimum
  - Maximum
- **JSlider implements ChangeListener**
  - Each slider then has an `addChangeListener()` method

# JSlider ticks and labels

- **Tick methods**
  - `getMajorTickSpacing`
  - `getMinorTickSpacing`
  - `getSnapToTicks`
  - `setMajorTickSpacing`
  - `setMinorTickSpacing`
  - `setSnapToTicks`
- **Label methods**
  - `public void setLabelTable(Dictionary labels)`
  - `public Dictionary getLabelTable()`
  - `protected void UpdateLabelUIs()`
  - `public Hashtable createStandardLabels(int increment, int start)`

# Customizing JSliders

- `setInverted(boolean true)`  
Reverses the direction of the Jsliders
- Filling sliders
  - Early versions of JSlider had “filled” portions of the bar
  - In JDK 1.4 it isn't filled
    - `slider.putClientProperty("JSlider.isFilled", Boolean.TRUE);`
- Drawing track
  - `setPaintTrack(boolean)`
- Code `MySlider2.java`



# Customizing methods

- Combining with other components
  - Since a JSlider should usually be used with at least a text field.
  - SliderPanel.java
    - Several Listeners to sync the slider and the text area
      - Focus
      - Change
      - Action
    - Several methods to control layout and access internal JSlider

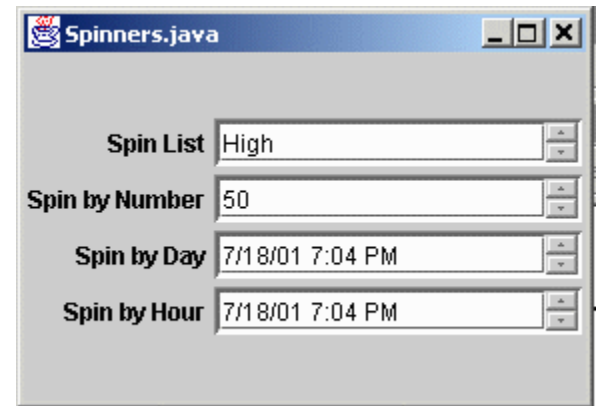


# JSpinner

- New in JDK 1.4
- Single line input field that lets the users "click" through a series of values that are displayed, one at a time, in the field
  - Use mouse
  - Keyboard up/down arrow keys
- JSpinner may allow user input (like in JComboBox)
- Sequence is defined by its SpinnerModel
  - SpinnerListModel
  - SpinnerNumberModel
  - SpinnerDateModel
- Constructors
  - JSpinner() (*default SpinnerNumberModel*)
  - JSpinner(SpinnerModel)
- Default editor is a JFormattedTextField
- You can monitor changes by using `addChangeListener(ChangeListener)`

# JSpinner Methods

- Common Methods:
  - getNextValue()
  - getPreviousValue()
  - getValue()
  - setEditable()
- Idiosyncrasies
  - JSpinner bases its size on its initial value...It is up to you to set a correct one
  - JSpinner has a beginning and an end, it does not roll over
    - You can extend SpinnerListModel though, to implement rolling over of a list
    - You can provide your own renderer to show icons instead of text
  - As with JTextFields, no labels are included
- Code Example, Spinners.java



# Custom Spinners Data Model

- Easy to create your own date model
- You can create Rollover models that wrap
- Code Example, `CustomSpinnerModel.java`

