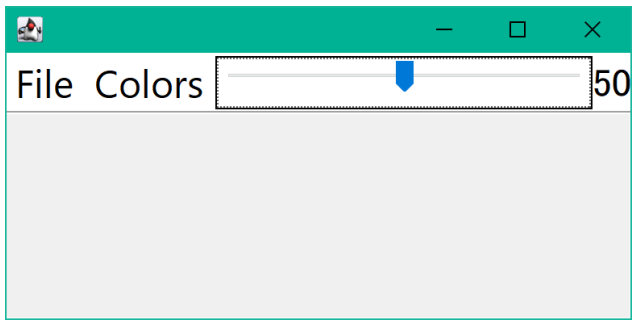
Graphical User Interface adding actions to widgets

Object Oriented Programming
2024 First Semester
Shin-chi Tadaki (Saga University)

Events and widgets

- GUI applications wait events and respond to them.
  - Generally GUI applications are in an infinite loop waiting events.
  - Events induce actions in the applications.
  - These mechanisms are called *event-driven*.
- Events
  - mouse: click, move, drag
  - keyboard: keycode
  - widgets: actions, state change
- Widgets have listener functions of events
  - Those functions receive events and invoke actions

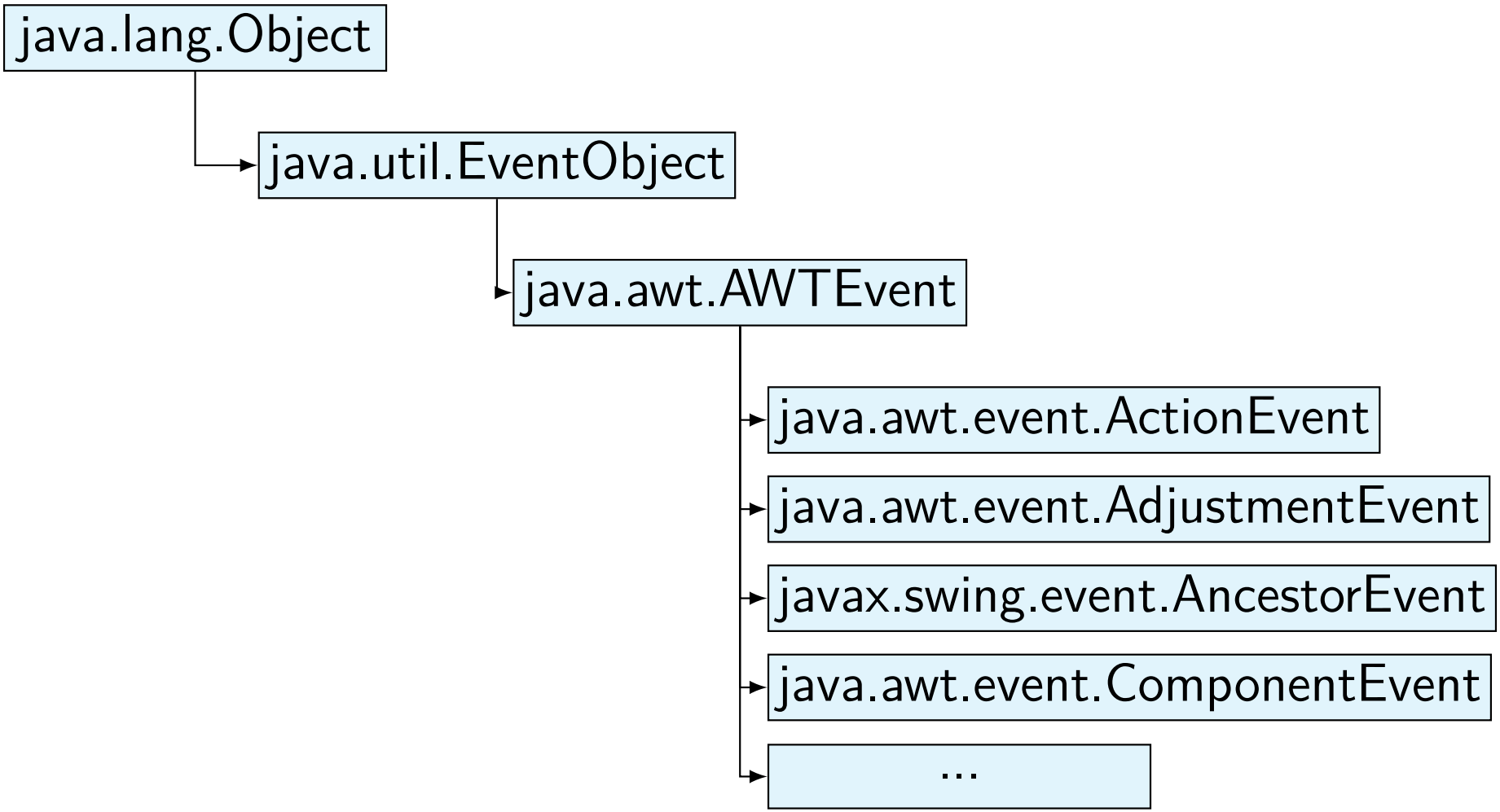
Today's theme: Devine actions in GUI



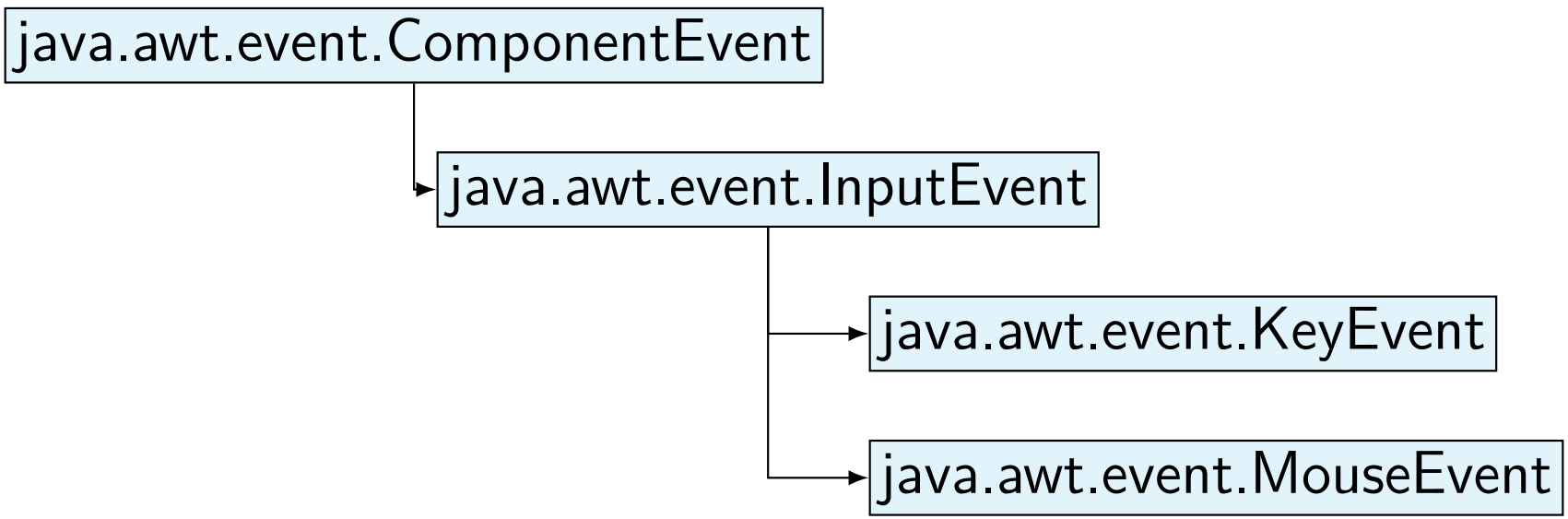
`guiWithAction.Main`

- Add actions to widgets
  - Actions for the menu items
  - Actions for the slider

Hierarchy of `java.awt.AWTEvent`



Hierarchy of `java.awt.event.inputEvent`



Define actions for buttons

- At the *Design* page in NetBeans
  - Double clicking a widget for defining actions
- Example: `actionPerformed` for `exit` menu item

```
1  private void exitActionPerformed(java.awt.event.ActionEvent evt)
2  {
3      dispose();
4  }
```

- The listener function is automatically defined in `initComponents()`
  - This function combines the widget `exit` and its action `exitActionPerformed()`.

```
1  exit.addActionListener(new java.awt.event.ActionListener() {
2      public void actionPerformed(java.awt.event.ActionEvent evt) {
3          exitActionPerformed(evt);
4      }
5  });
```

Example: Define actions for Color menu

Set color to the panel by selecting colors in selectColors menu.

```
1  for (ColorItem color : ColorItem.values()) {//Add colors to the menu
2      JMenuItem item = new JMenuItem(color.toString());
3      item.setFont(font);
4      //Add an action to the item
5      item.addActionListener(e -> colorItemPerformed(color));
6      selectColors.add(item);
7  }
```

```
1  private void colorItemPerformed(ColorItem c) {
2      System.out.println(c.toString());
3      panel.setBackground(c.getColor());
4  }
```

Add slidebar to the menubar

- NetBeans supports to add only `JMenu` to `JMenuBar`
- You can manually add any widgets to `JMenuBar`.
- The following source code shows how to add `JSlider` to the menu bar.

```
1  //Create slider instance
2  slider = new JSlider();
3  slider.setFont(font);
4  slider.setPaintTicks(true);
5  slider.setBorder(BorderFactory.createLineBorder(Color.BLACK));
6  slider.setBackground(Color.white);
7  //Define slider response
8  slider.addChangeListener(e -> sliderStateChanged(e));
9  menuBar.add(slider);//Add slider to the menu
```

```
1  private void sliderStateChanged(javax.swing.event.ChangeEvent evt) {
2      int v = slider.getValue();
3      label.setText(String.valueOf(v));
4  }
```

File Chooser: an example

- This application provides functions for opening and saving files
- It uses the `JFileChooser` class included in Java Swing.
- Functions
  - Open a file
  - Show the text
  - Save the text to a file
  - Show error dialogs

Action for open button

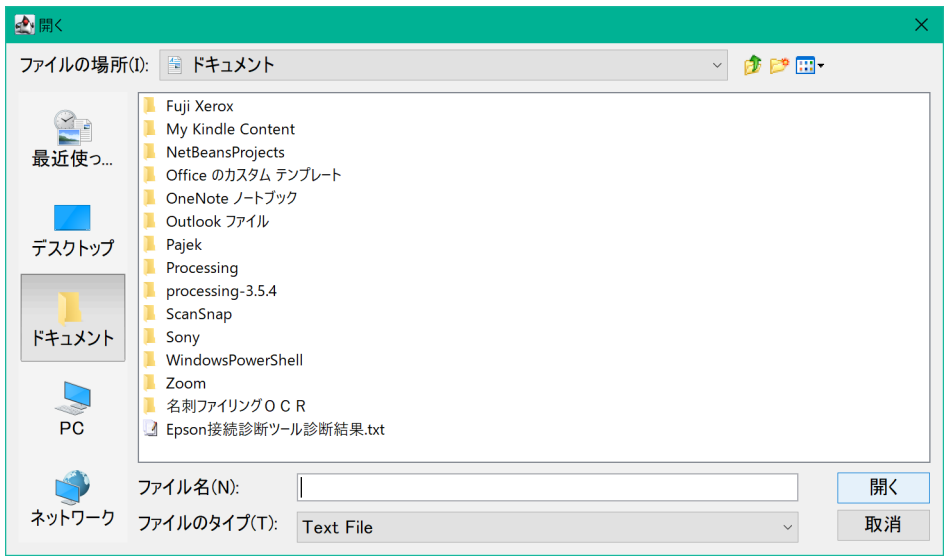
```
1  private void openMenuActionPerformed(java.awt.event.ActionEvent evt) {
2      //Create file chooser and set filter for text files
3      JFileChooser chooser = new JFileChooser();
4      chooser.setCurrentDirectory(dir);
5      chooser.setFileFilter(
6              new FileNameExtensionFilter("Text File", "txt"));
7
8      int returnVal = chooser.showOpenDialog(this);//Show dialog
9      if (returnVal == JFileChooser.APPROVE_OPTION) {
10         File file = chooser.getSelectedFile();//Selected file
11         //Show text in textArea
12         textArea.setText(FileUtilGUI.openFile(file));//
13         textArea.setVisible(true);
14         filename = file.getName();
15         setTitle(applicationName+" "+filename);
16         dir = file.getParentFile();
17     }
18 }
```

Action for save button

```
1  private void saveTextActionPerformed(java.awt.event.ActionEvent evt) {
2      //Create file chooser and set filter for text files
3      JFileChooser chooser = new JFileChooser();
4      chooser.setCurrentDirectory(dir);
5      chooser.setFileFilter(new FileNameExtensionFilter("Text File",
       ↪  "txt"));
6
7      int returnVal = chooser.showSaveDialog(this);
8      if (returnVal == JFileChooser.APPROVE_OPTION) {
9          File file = chooser.getSelectedFile();
10         FileUtilGUI.saveFile(file, textArea.getText());
11         filename = file.getName();
12         this.setTitle(applicationName+" "+filename);
13         dir = file.getParentFile();
14     }
15 }
```

JFileChooser class

- Provides the standard widget used for file selections
- Return values
  - status of file selection
  - properties of selected file
- `FileNameExtensionFilter` restricts files by extensions.



FileUtilGUI class

- Reading text from the file
- Saving text into the File
- Check whether the file is writable
- Show error and confirmation dialogs
- Obtain file extensions

Show dialogs

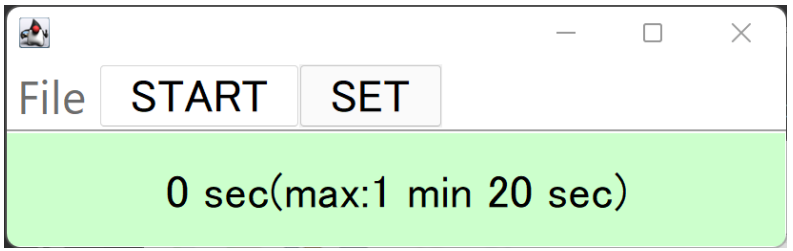
```
1   static public void showError(String message) {
2       JOptionPane.showMessageDialog(
3               new JFrame(), message, "Error",
4               JOptionPane.ERROR_MESSAGE);
5   }
6
7   static public void showMessage(String message) {
8       JOptionPane.showMessageDialog(
9               new JFrame(), message, "Message",
10              JOptionPane.INFORMATION_MESSAGE);
11  }
```

Confirming writability

```
1  static public boolean checkOverwrite(String filename) {
2      boolean b = true;
3      String message = filename + "exists. Do you overwrite?";
4      int answer = JOptionPane.showConfirmDialog(
5              new JFrame(), message, "Confirm overwrite",
6              JOptionPane.OK_CANCEL_OPTION);
7      if (answer != JOptionPane.OK_OPTION) {
8          b = false;
9      }
10     return b;
11 }
```

```
1  static public boolean checkWritable(File file) {
2      boolean isWritable = true;
3      if (file.isFile()) {//Confirm the file existing
4          if (!file.canWrite()) {//Overwritable?
5              showError("Can not write to " + file.getName());
6              return false;
7          } else {
8              if (!checkOverwrite(file.getName())) {
9                  return false;
10             }
11         }
12     } else {//New file
13         try {
14             if (!file.createNewFile()) {//Create new file
15                 showError("Can not create " + file.getName());
16                 return false;
17             }
18         } catch (IOException ex) {
19             showError(ex.getMessage());
20             return false;
21         }
22     }
23     return isWritable;
24 }
```

Simple Timer: the second example



- Main menu provides three functions
  - Toggle button for start / save
  - `Set` button setting time limit
  - `Exit` button closing the applications
- Main body of the timer application
  - Is the extension of `JLabel` class
  - Shows the current time
- A separate Panel
  - Is invoked by pressing the `Set` button
  - Limits time with minute and second
  - Is shown inside `JOptionPane`

Running timer as a thread: `Timer` class

- Running as a separate thread using `Runnable` Interface
- Checking the difference between the start and current times

```
1  public void run() {
2      while (running) {
3          setTime();
4          try {
5              Thread.sleep(100);
6          } catch (InterruptedException ex) {
7          }
8      }
9  }
```

setTime() method

```
1  public boolean setTime() {
2      Calendar c = Calendar.getInstance();
3      //Get duration (sec) from the beginning
4      int d = (int) (c.getTimeInMillis() - startDate.getTimeInMillis()) /
       ↪  1000;
5      setTimeString(d);
6      if (d >= max) {//Exceed the limit
7          setForeground(foregroundOver);
8          return false;
9      }
10     setVisible(true);
11     return true;
12 }
```

START / STOP toggle button

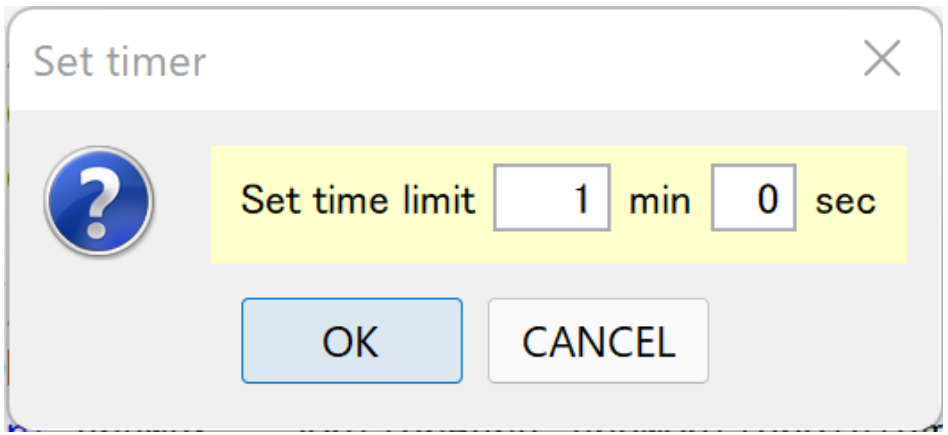
```
1   //Create START/STOP toggle button
2   toggle = new JToggleButton("START");
3   toggle.setFont(font);
4   toggle.addActionListener(evt -> toggleActionPerformed(evt));
5   menuBar.add(toggle);
6
7   //Create button for popping up SetTimerPanel
8   setButton = new JButton("SET");
9   setButton.setFont(font);
10  setButton.addActionListener(e -> setTimeActionPerformed(e));
11  menuBar.add(setButton);
```

Action for setTime

```
1  private void setTimeActionPerformed(java.awt.event.ActionEvent evt) {
2      //Stop timer
3      toggle.setSelected(false);
4      toggle.setText("START");
5      timerLabel.stop();
6      //Show dialog for setting
7      Object[] options = { "OK", "CANCEL" };
8      int answer = JOptionPane.showOptionDialog(
9              new JFrame(), setTimePanel,
10             "Set timer", JOptionPane.OK_CANCEL_OPTION,
11             JOptionPane.QUESTION_MESSAGE, null, options, options[0]);
12     if (answer == JOptionPane.OK_OPTION) {
13         //Set time limit by pressing OK
14         int m = setTimePanel.getMinute();
15         int s = setTimePanel.getSecond();
16         timerLabel.setMax(60 * m + s);
17     } else {
18         setTimePanel.setDefault();
19     }
20 }
```

SetTimePanel

- Message object in `JOptionPane`
- Text form setting minute and second
- Close dialog by `OK` button



Exercise

Add a new function for changing font style of a text area (see quiz).