Graphical User Interface using widgets

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

1. GUI in Java
2. `java.awt`
3. `javax.swing`
4. Working with `JFrame`
5. GUI without actions

GUI (Graphical User Interface) in Java

- GUI libraries in general
  - X11 with c/c++, etc.
  - Generally OS dependent
- in Java
  - GUI libraries are distributed with JDK
  - OS independent
  - Working under OS dependent window managers

GUI programming as OOP

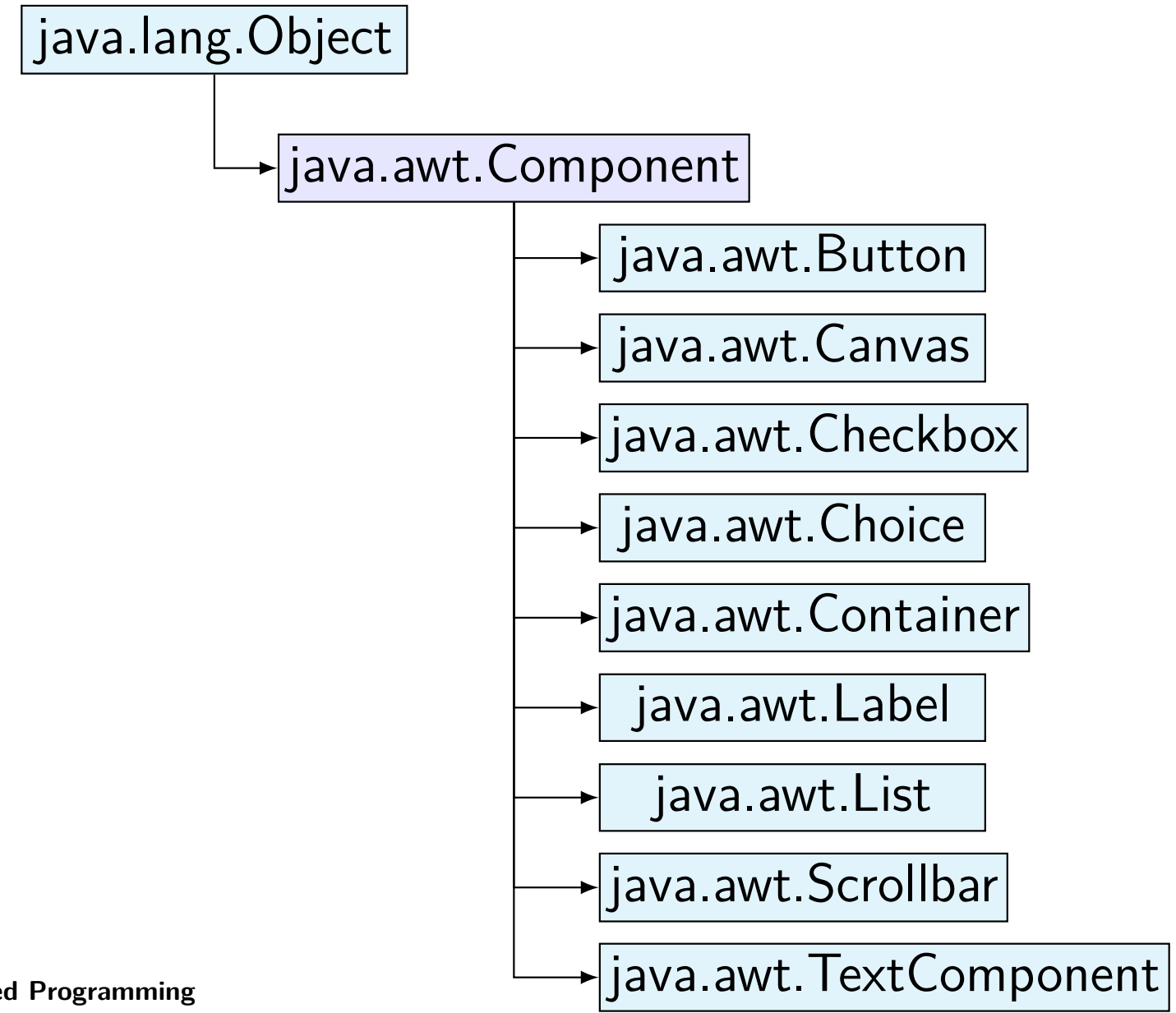
- GUI uses various widgets (*window gadgets)*
- Each widget has own properties and methods
  - Properties: color, size, etc.
  - Methods: action, property change, visible, etc.
- Widgets communicates other widgets through methods.
- Fundamental widgets are used for applications by extensions.
  - GUI applications by extending `JFrame`
  - Widget containers by extending `JPanel`

java.awt: Abstract Windows Toolkit

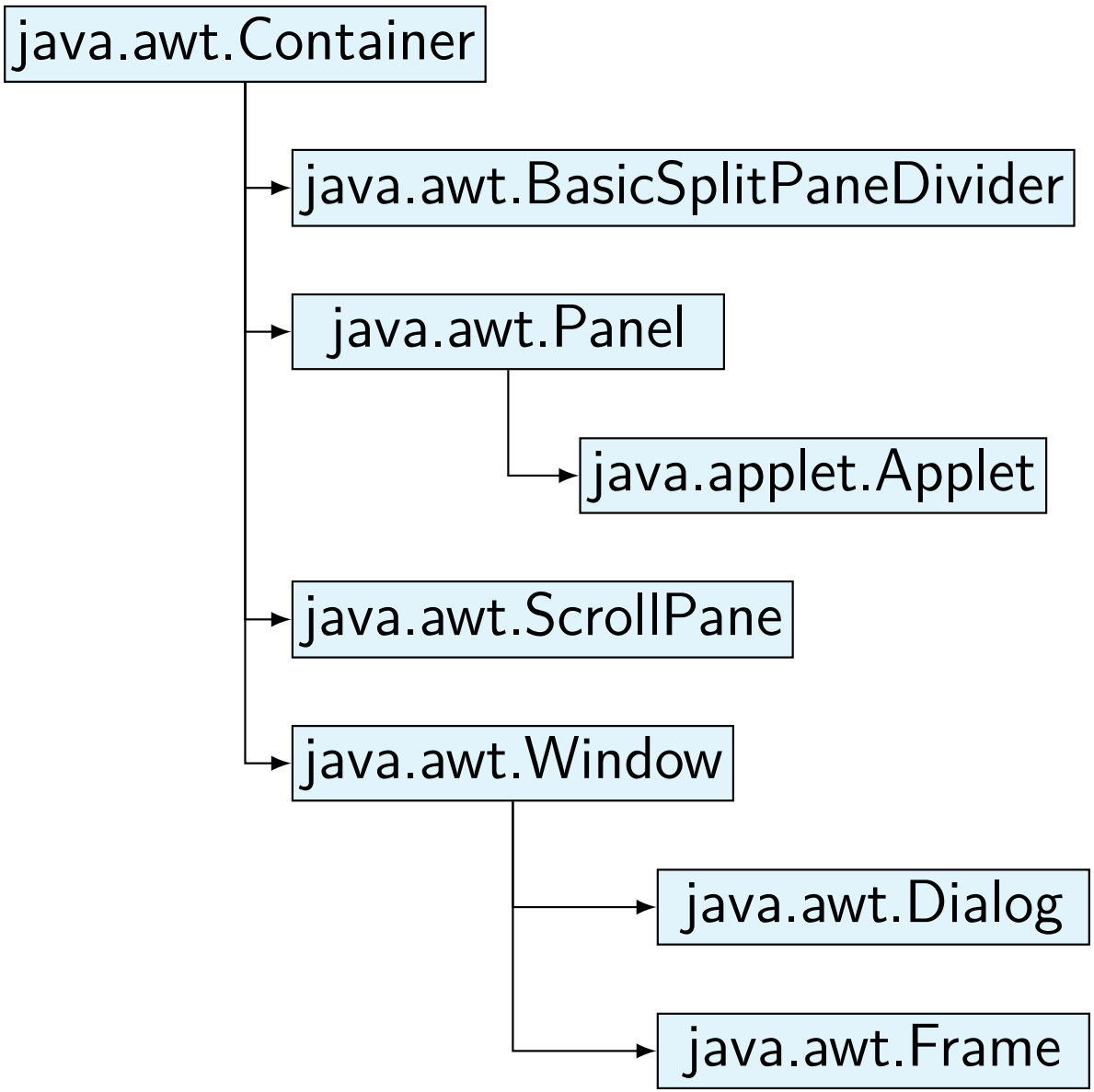
java.awt provides

- Fundamental graphical properties
  - Color, BasicStroke, Font, etc.
- Fundamental widgets
  - panels, buttons, etc.
- Fundamental events
  - mouse, keyboard, property changes, etc.

Hieralchy of java.awt



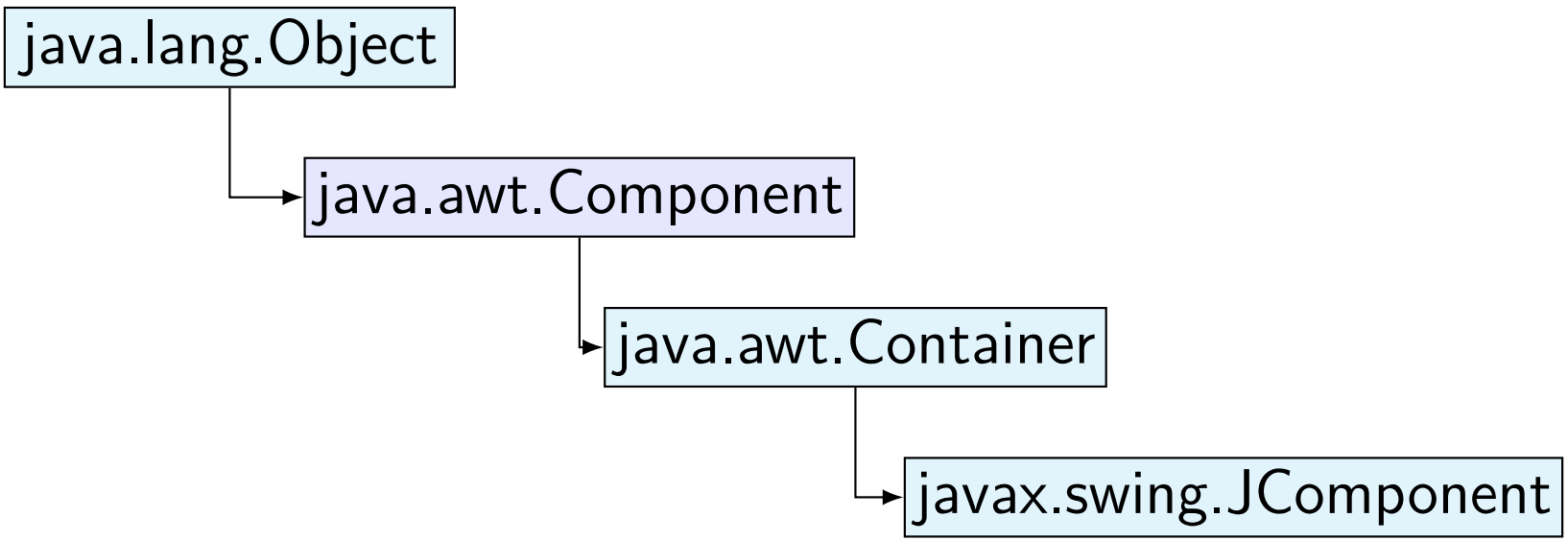
Hieralchy of java.awt: cont.



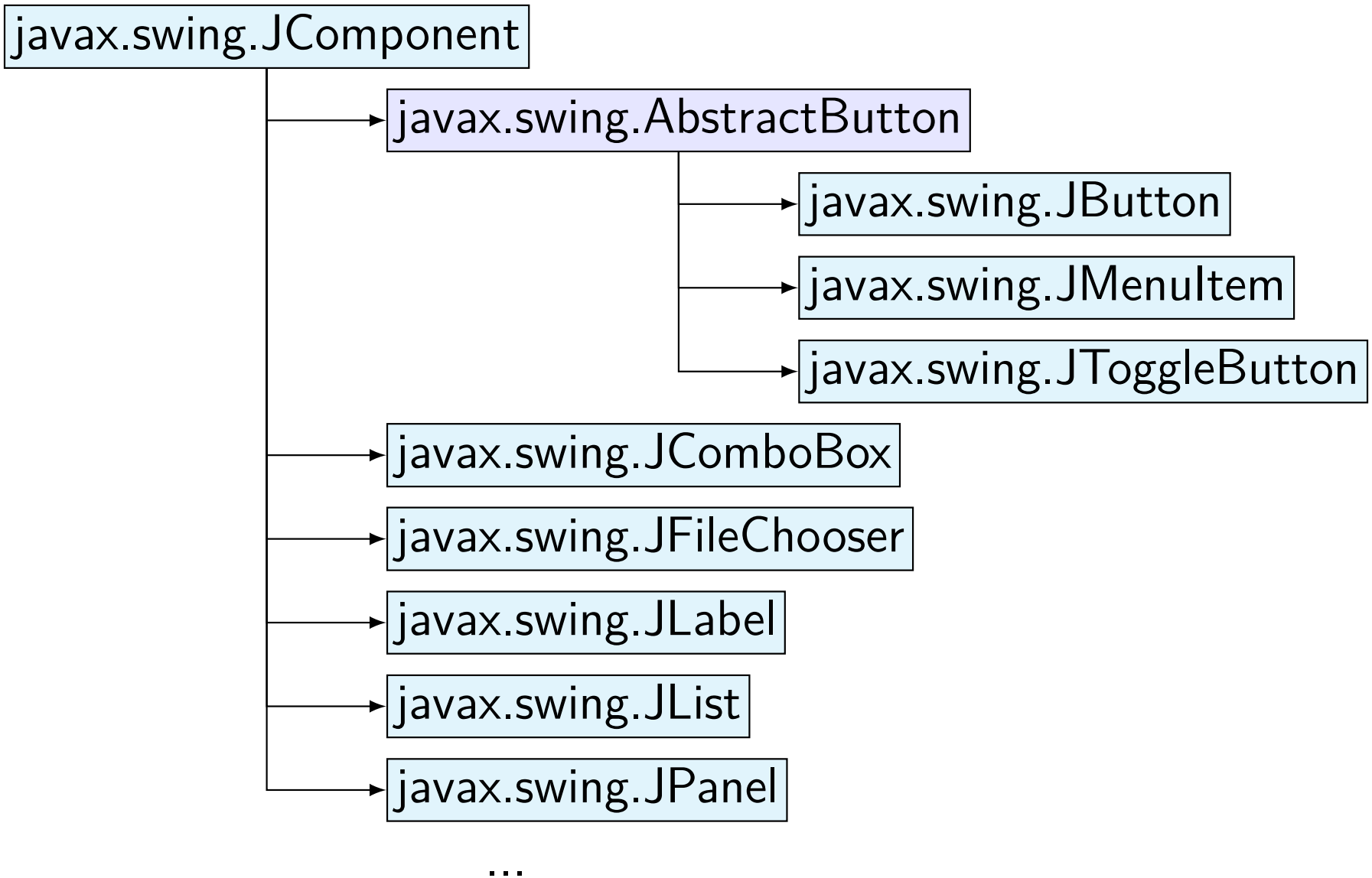
javax.swing

- Extensions of `java.awt`
- Enriching widgets
- Completely OS independence
  - Control under OS window manager
  - Separate Look-and-Feel
- Lightweight
- Running as threads

Hieralchy of swing widgets



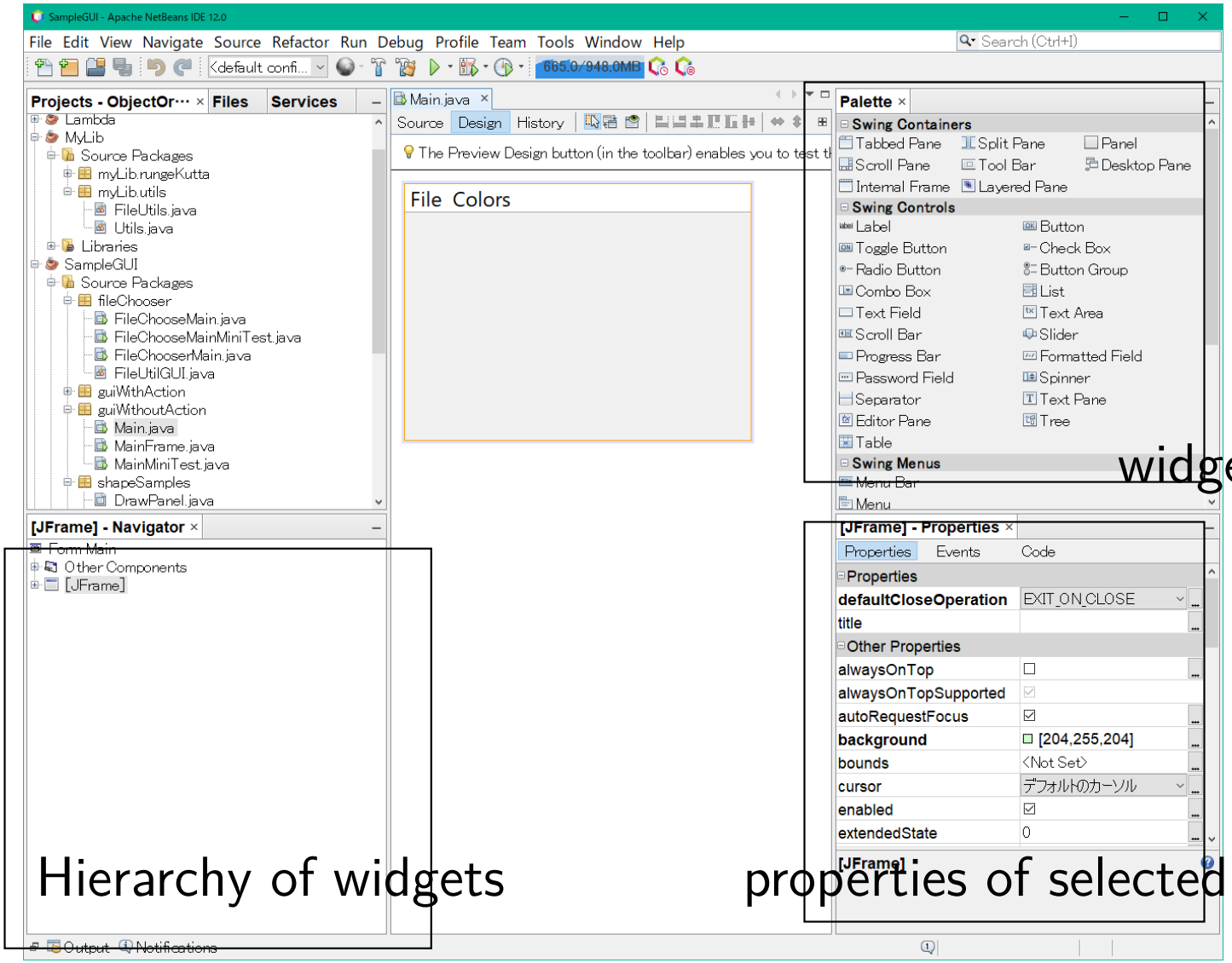
Hieralchy of swing widgets: cont.



swing components

- javax.swing.JFrame
  - Main window of applications
  - Put JPanel and JMenuBar instances onto this component
- javax.swing.JPanel
  - Put widgets on this components
  - Used for drawing
- javax.swing.JMenuBar
  - Menu bar at the top of applications
  - Put javax.swing.JMenu instances on this component

Layout Design in NetBeans



Constructing GUI in NetBeans

- Start a project as usual applications
- Create `JFrame` form for *Main* class of the application
  - `New`→`JFrame form`
  - At widget hierarchy: `Set Layout`→`BorderLayout`
  - The *Main* class is defined as a new class by extending `JFrame`

Configuring widgets

- Configuring widgets using mouse
  - In `Navigation`: Drag a component from the palette
- Creating `JMenuBar`
  - Two `JMenu` instances `File` and `Edit` are added initially.
  - Add `JMenu` and `JMenuItem`

Exercise

- Create a new `JFrame` instance.
- Add a `JMenuBar` instance
- Add a `JMenuItem` instance to the `JMenuBar`
- Set some attributes to `JMenuBar` and `JMenuItem` instances

Notice at creating new JFrame instances

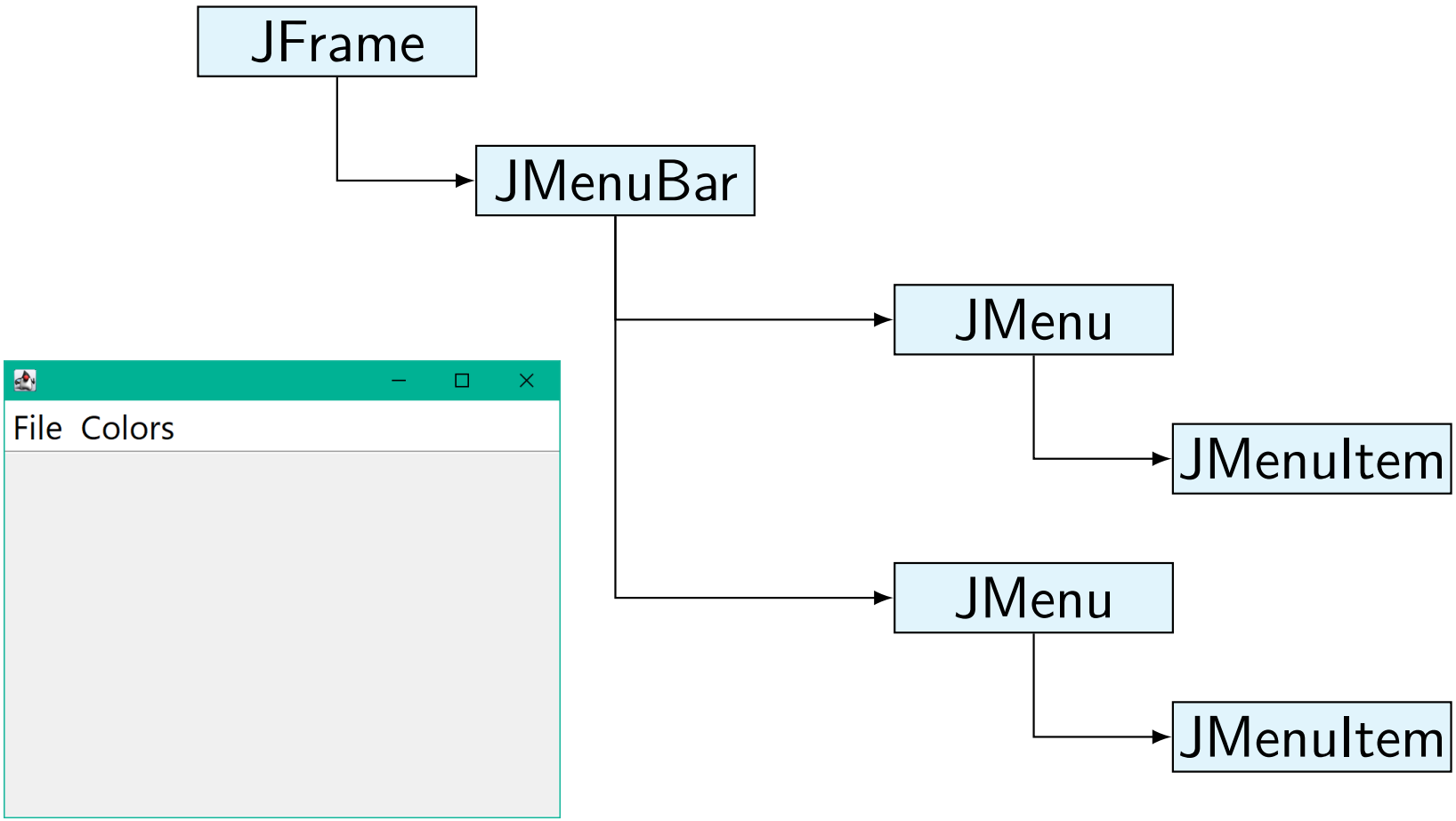
- Properties and layout are stored in `*.form` file.
  - Parts of source files are not allowed to edit, because some configurations are stored in `*.form` files.

Today's sample program

https://github.com/oop-mc-saga/GUI1

- guiWithoutAction
- guiWithAction
- fileChooser
- simpleTimer

GUI without actions



Two menus in this application

- The first menu `fileMenu`
  - has menu item `exit`,
  - which is added through the *design* interface of NetBeans.
- The second menu `selectColors`
  - has menu items for selecting a color defined in the enum type `ColorItem`.
  - Those menu items are added in the constructor

Main part

```
1  public class Main extends javax.swing.JFrame {
2
3      public enum ColorItem {//Define colors as enum instance
4          ORANGE(Color.ORANGE), YELLOW(Color.YELLOW),
           ↪  GREEN(Color.GREEN);
5          private final Color color;
6
7          ColorItem(Color color) {  this.color = color;  }
8
9          public Color getColor() {  return color;  }
10     }
11
12     public Main() {
13         initComponents();
14         Font font = new Font("MS UI Gothic", 0, 24);
15         for (ColorItem m : ColorItem.values()) {
16             JMenuItem item = new JMenuItem(m.toString());
17             item.setFont(font);
18             selectColors.add(item);
19         }
20     }
21     ...
22 }
```

`initComponents()`

- Generated automatically with `form` file through NetBeans
- What `initComponents()` does is
  - Inserting widgets and laying out them
  - Setting properties of widgets
- Operations in `initComponents()` are defined through the *design* interface of NetBeans.

`enum` type

- `enum` allows us to define a set of named constants.
- Items in `enum` can have properties and methods.
- `enum` types are useful for `switch-case` clauses.

Example 5.1: enum

```
1  public class EnumExample {
2      public static enum ColorName{
3          RED, GREEN, BLUE;
4      }
5
6      public static void main(String[] args) {
7          ColorName colorName = ColorName.BLUE;
8          String colorCode = null;
9          switch(colorName){
10             case RED -> colorCode = "#FF0000";
11             case GREEN -> colorCode=  "#00FF00";
12             case BLUE -> colorCode = "#0000FF";
13             default -> {
14             }
15         }
16         System.out.println(colorCode);
17     }
18
19 }
```

Exercise

Add a new menu for selecting color (see quiz).