61FIT3JSD
Fall 2023

Lecture 6

GUI programming (1)

Basic issues

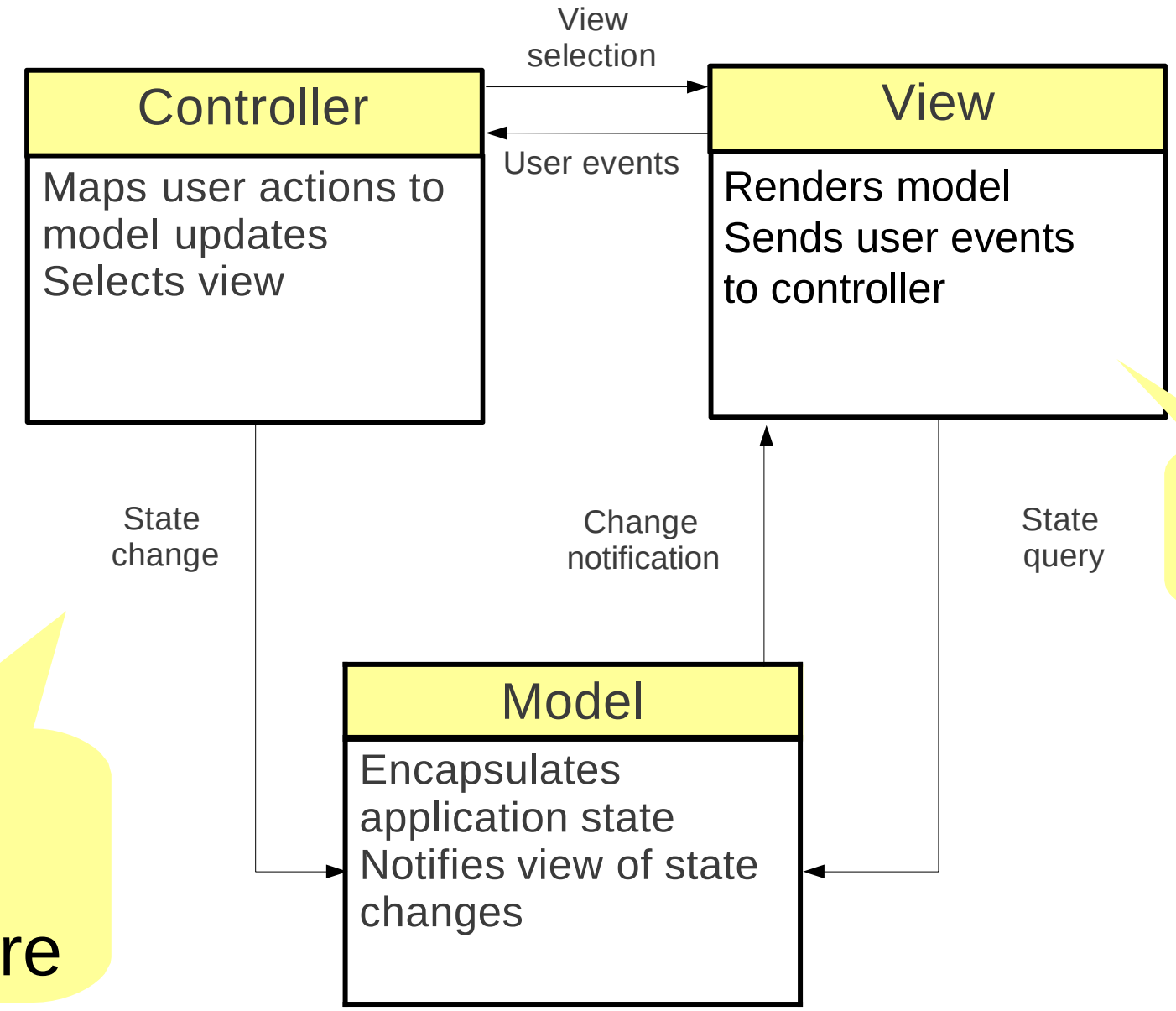
Lecture outline

- GUI & GUI application
- Event driven programming
- Java's support for GUI programming:
  - Swing, AWT
- Basic GUI application:
  - tasks and components

GUI & GUI application

- A **graphical user interface (GUI)** consists of display components of an application
- Display components present information/data graphically
- A **GUI application** is application + GUI:
  - must capture and handle user interactions on GUI

A general application architecture



Our GUI application scope

- 80% GUI:
  - display components & how to create them
  - 3 levels: basic, intermediate, advanced
- 15% user interaction:
  - event-driven programming
- 5% design: intermediate design based on MVC

Java's support for GUI programming (1)

- Two APIs: AWT & Swing
- **Abstract Window Toolkit (AWT)**:
  - (older) introduced since JDK 1.0
  - decouple display component specifications from their implementations
  - *heavy-weight*: uses a native peer class for each display component
  - the peer classes are fixed for given platform
  - has a modular interaction model (JDK 1.1)

Java's support for GUI programming (2)

- **Swing**:
  - introduced since JDK 1.2
  - improved from AWT
  - *light-weight*: Java's implementation of the display (no need for peer classes)
  - look & feel of the display components can be customised for a given platform
  - has enhanced display components
  - extends AWT 1.1's interaction model

Event-driven programming

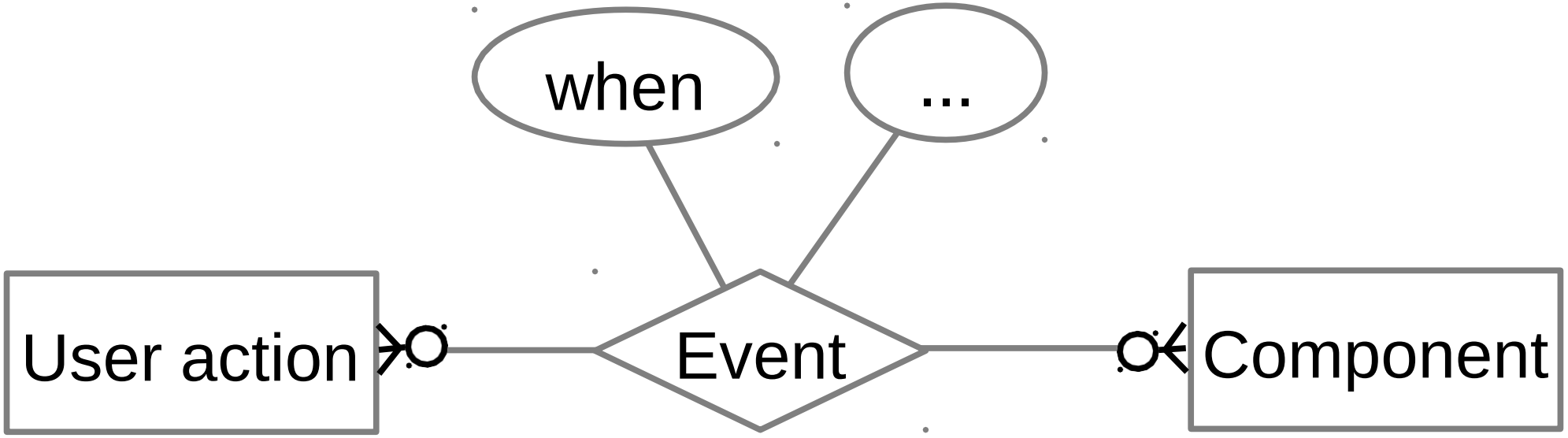
- User interaction model
- Event-driven programming (EDP)
- Java's support for EDP

What is EDP?

- Programming technique to *capture* and *handle* events caused by interactions with external entities
- EDP is *not* just for GUI applications
- EDP scope for GUI application:
  - user interactions on GUI components

Event-driven programming

User interaction model

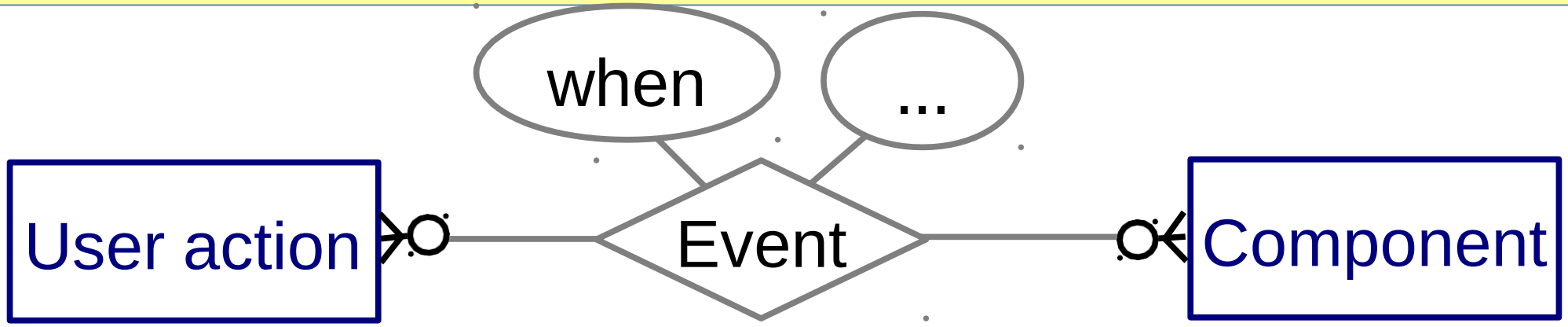


User interaction

- A user interaction consists of a sequence of actions
- *User action* is performed via an input device (mouse or keyboard)
- User action on a display component results in an *event*

Event-driven programming

User action & component



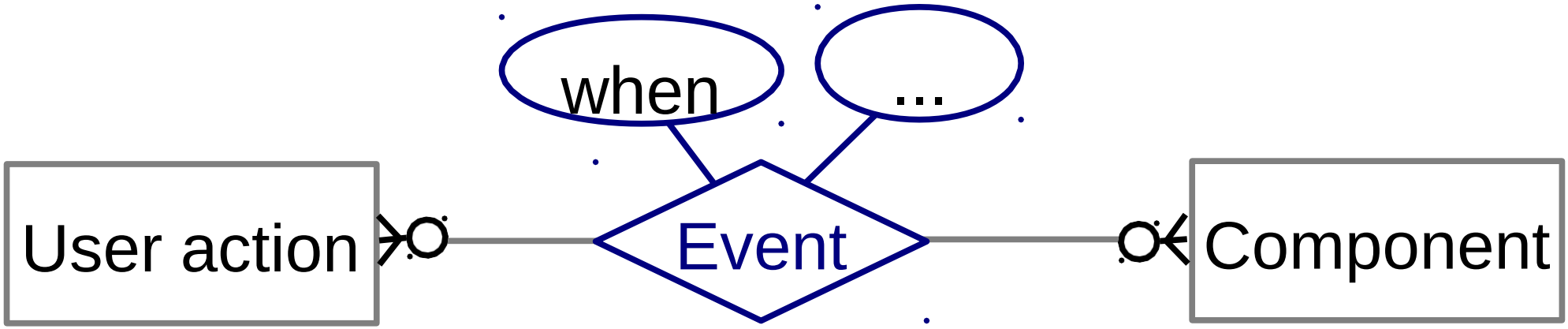
mouse:
- click
- double-click
- move
- ...

keyboard:
- key press
- key release
- ...

- window
- button
- label
- text field
- ...

Event-driven programming

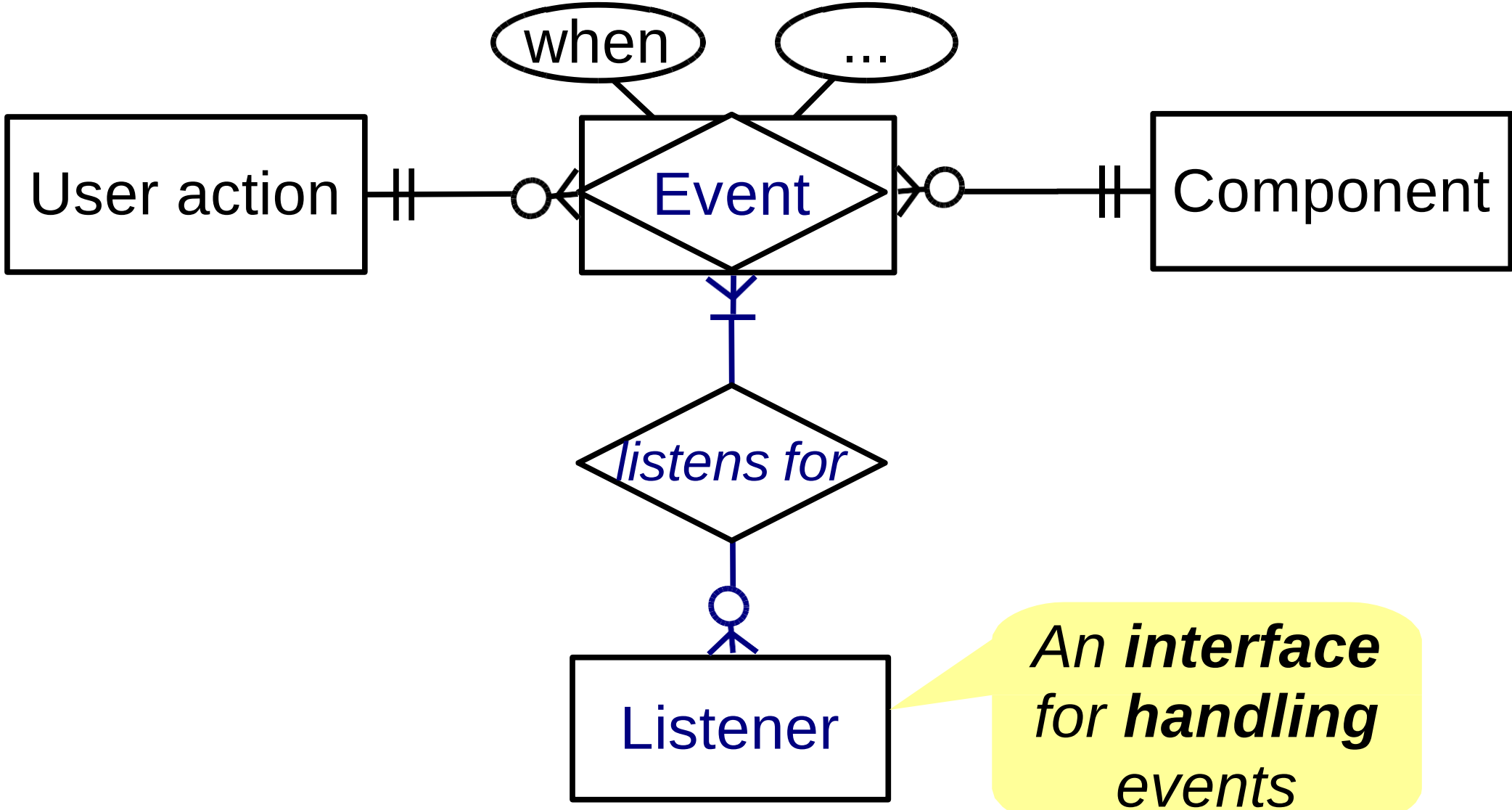
What is an event?



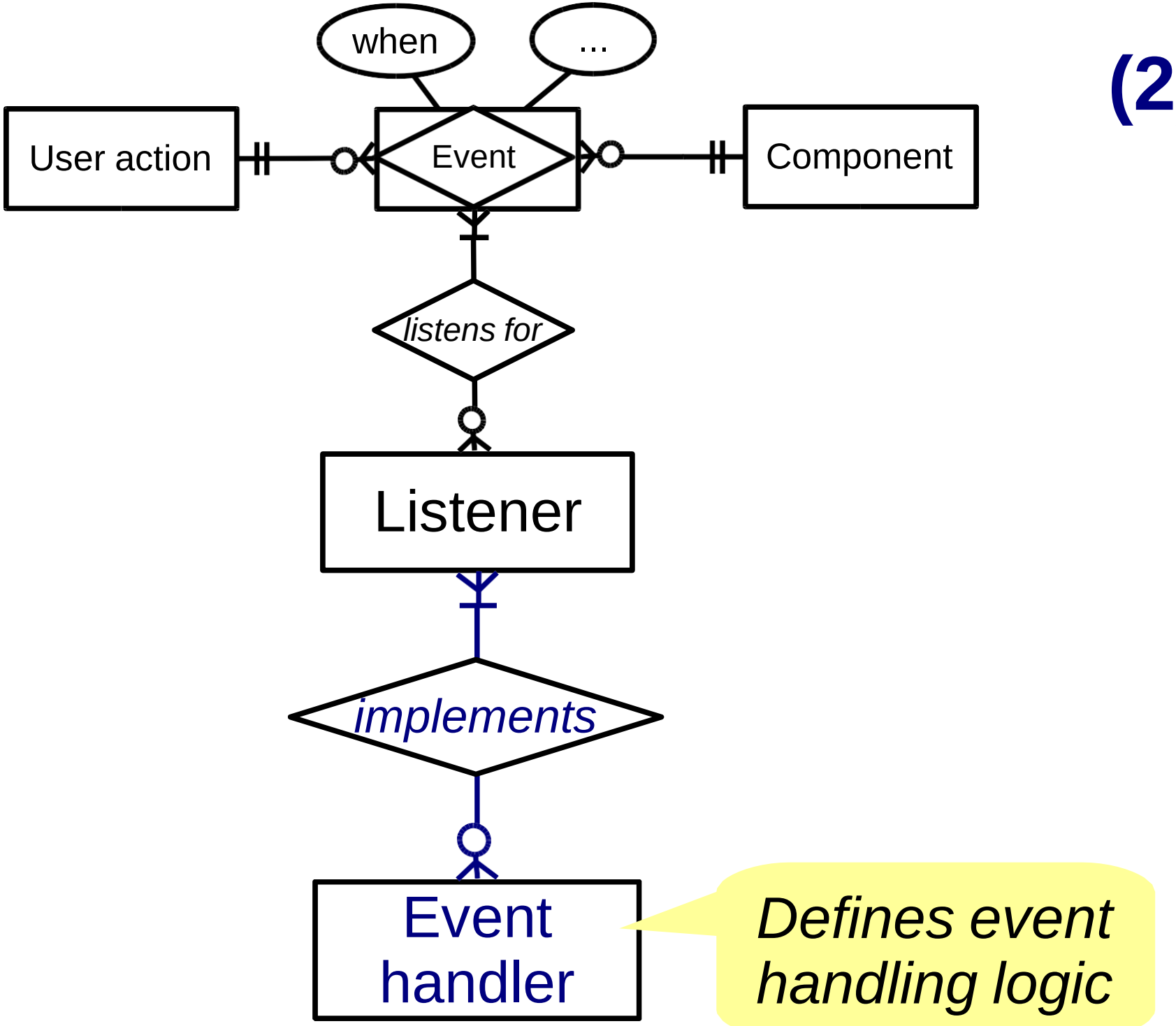
an occurrence of a user action on a display component

Event-driven programming

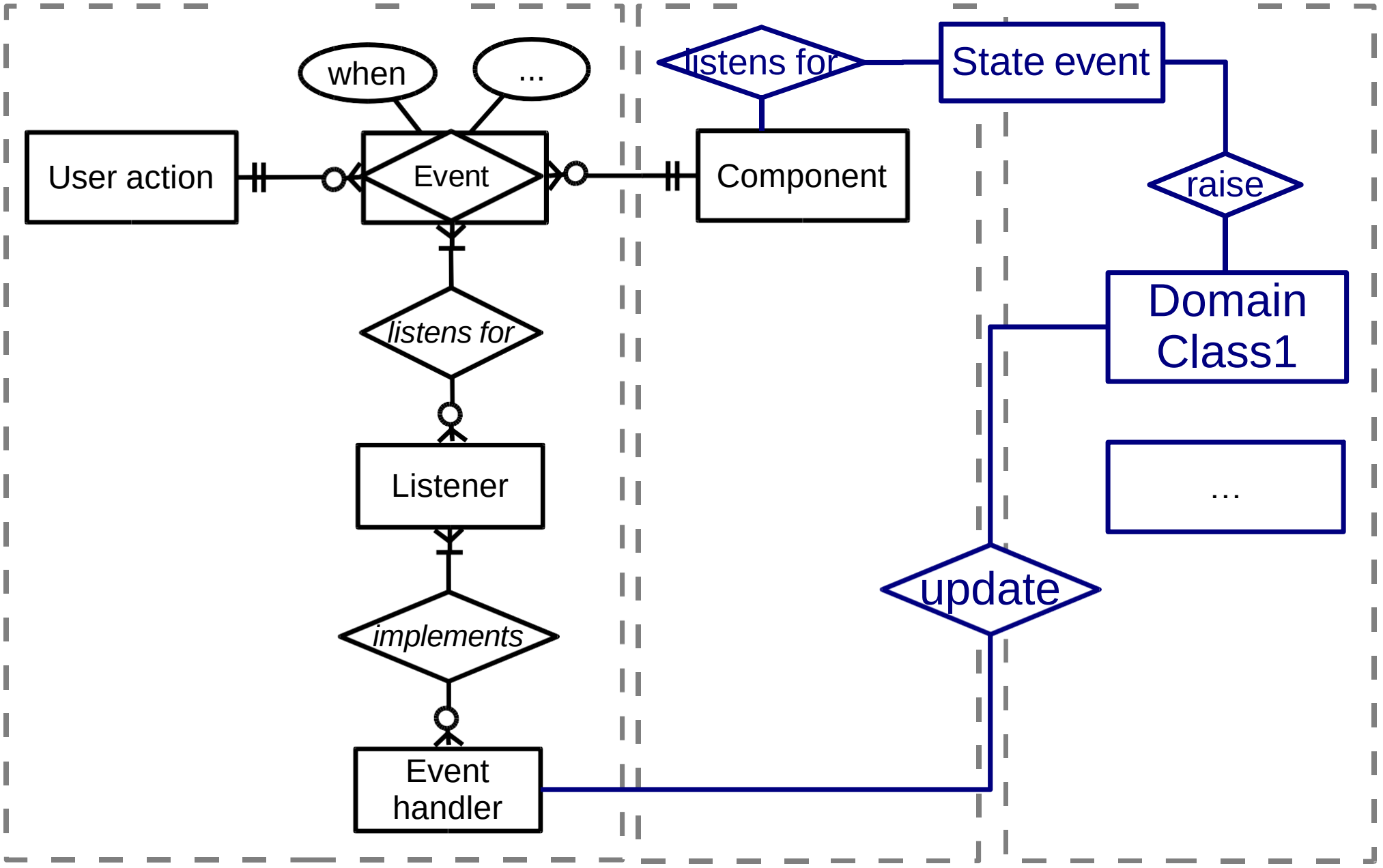
EDP model (1)



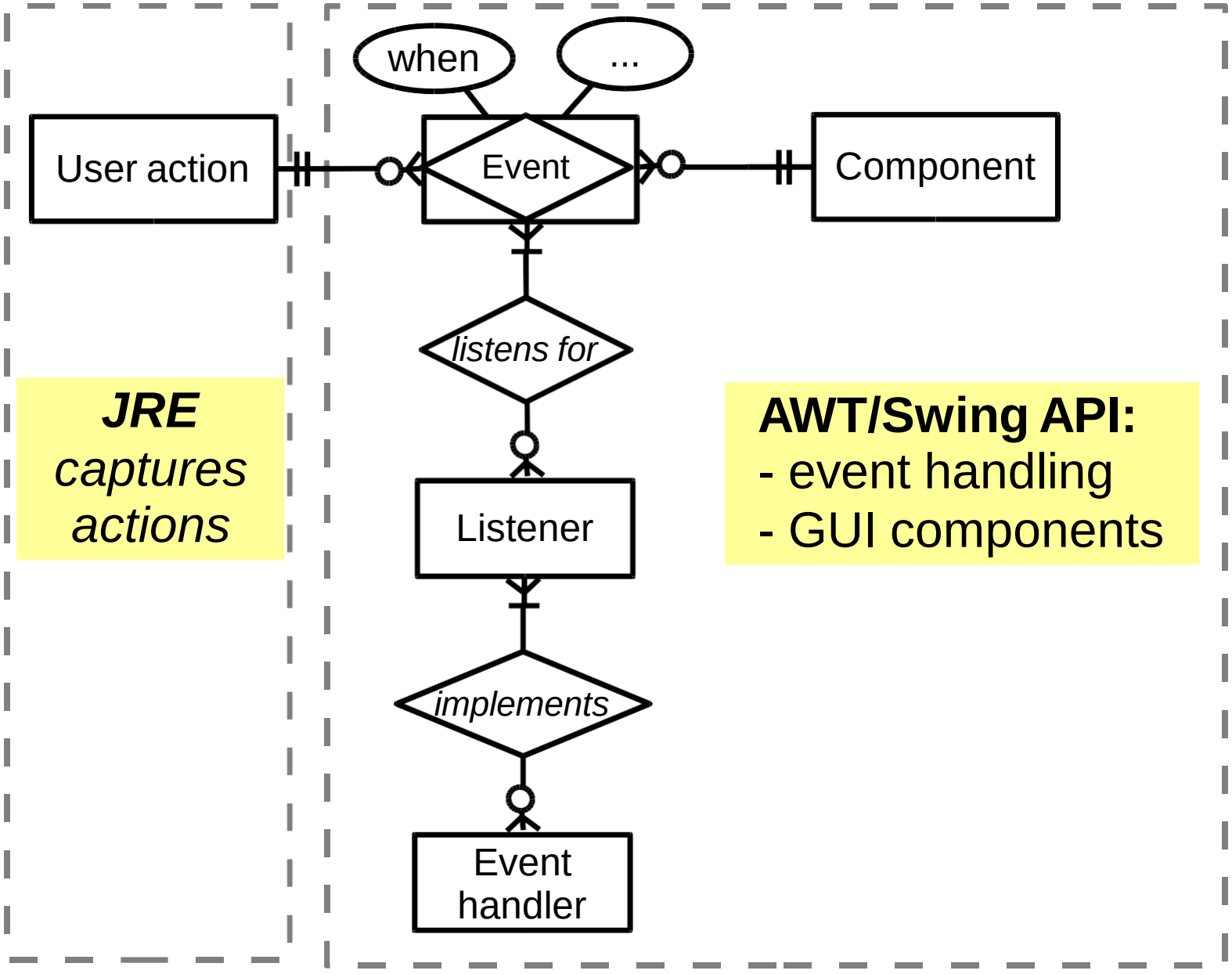
(2)


when
...
User action
Event
Component
listens for
Listener
implements
Event handler
Defines event handling logic

An event-driven GUI architecture



Java's support for EDP



Basic GUI application

- Tasks
- Components:
  - Window
  - Button
  - Color

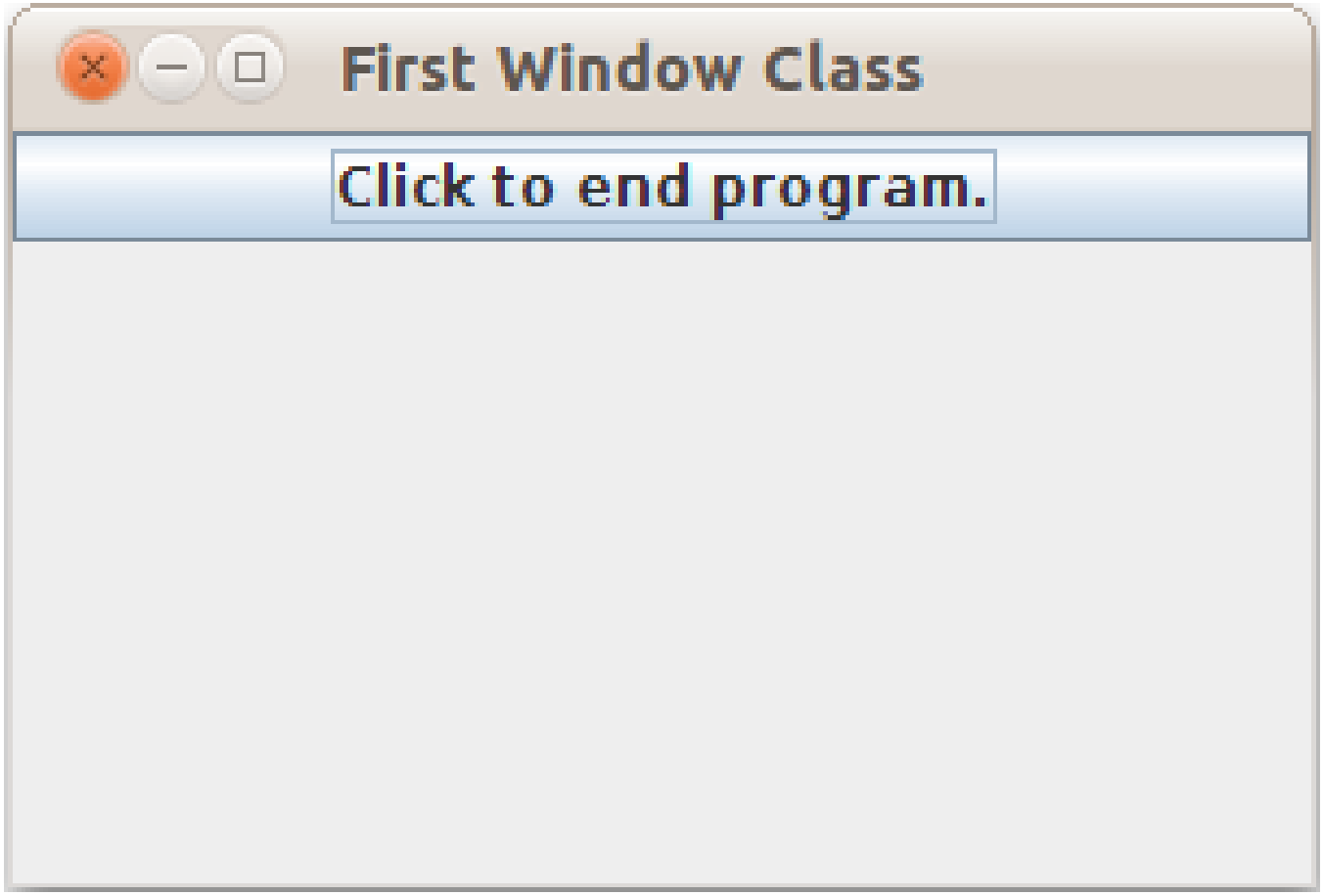
Basic GUI application

Tasks

- Task 1: Create a **window** object
- Task 2: **Set up** the window
- Task 3: Handle relevant window's **events**
- Task 4: For each display **component**:
  - Task 4.1: create
  - Task 4.2: set up
  - Task 4.3: handle the relevant events
  - Task 4.4: add to the window
- Task 5: **Display** the window

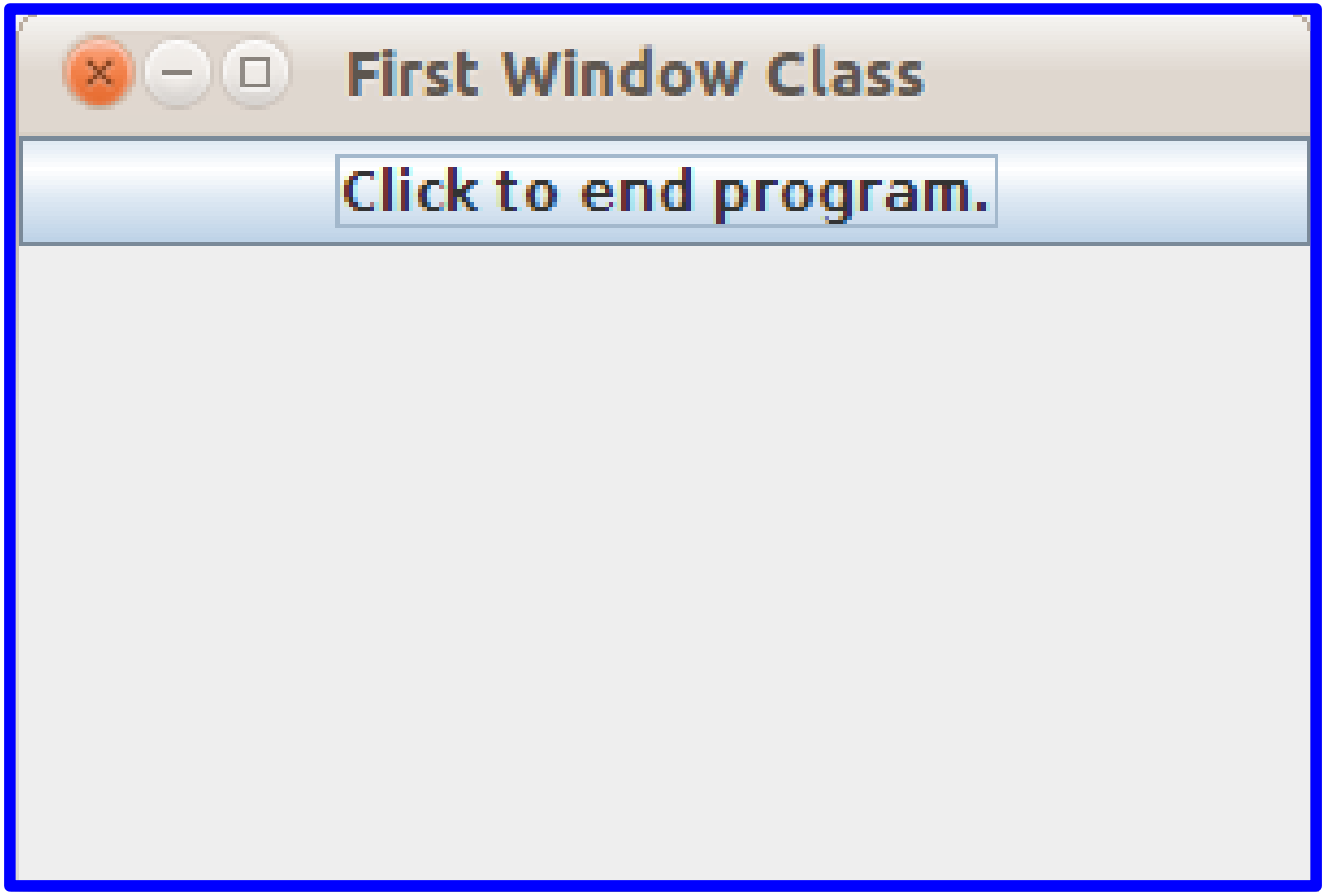
Basic GUI application

Example



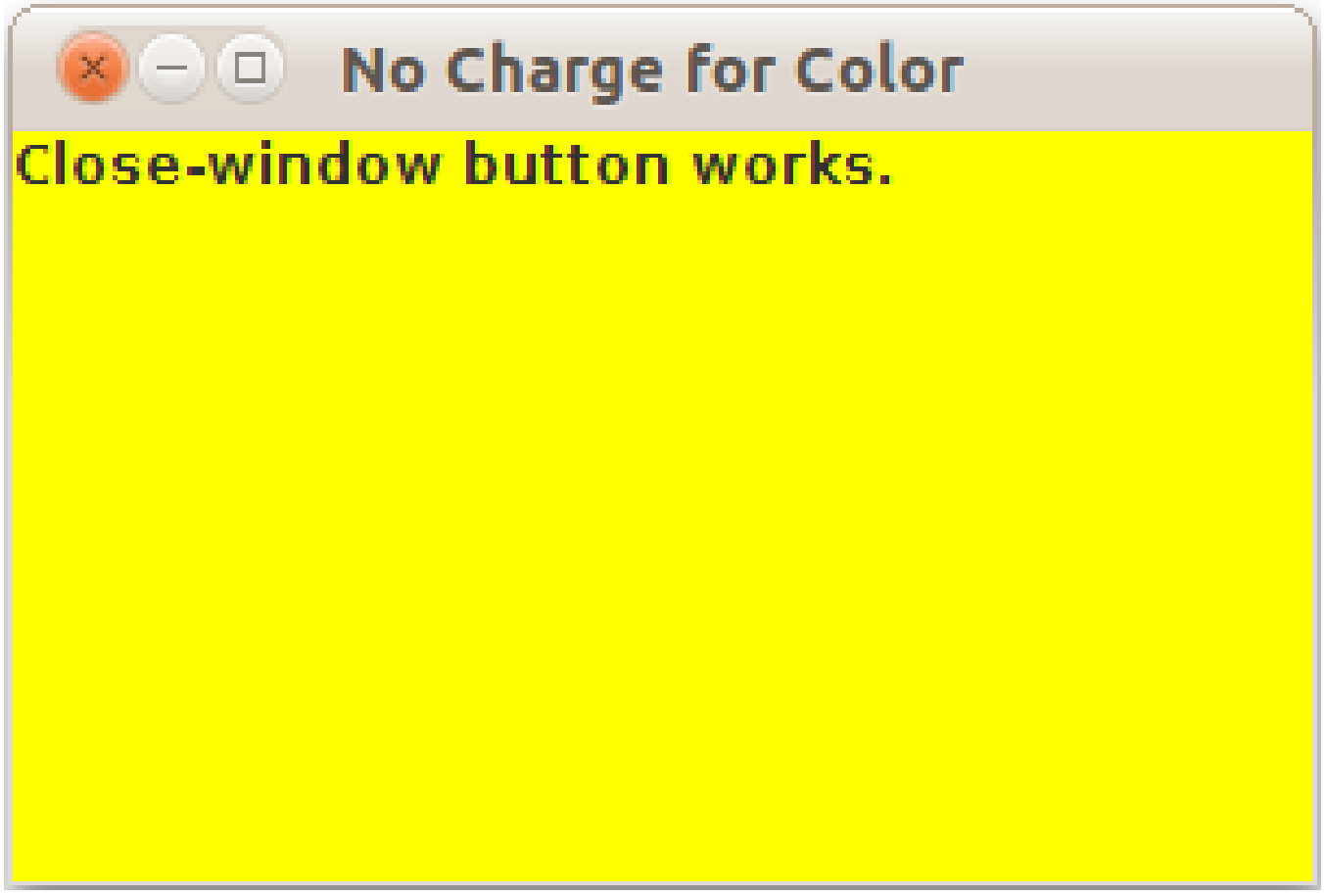
Basic GUI application

Window



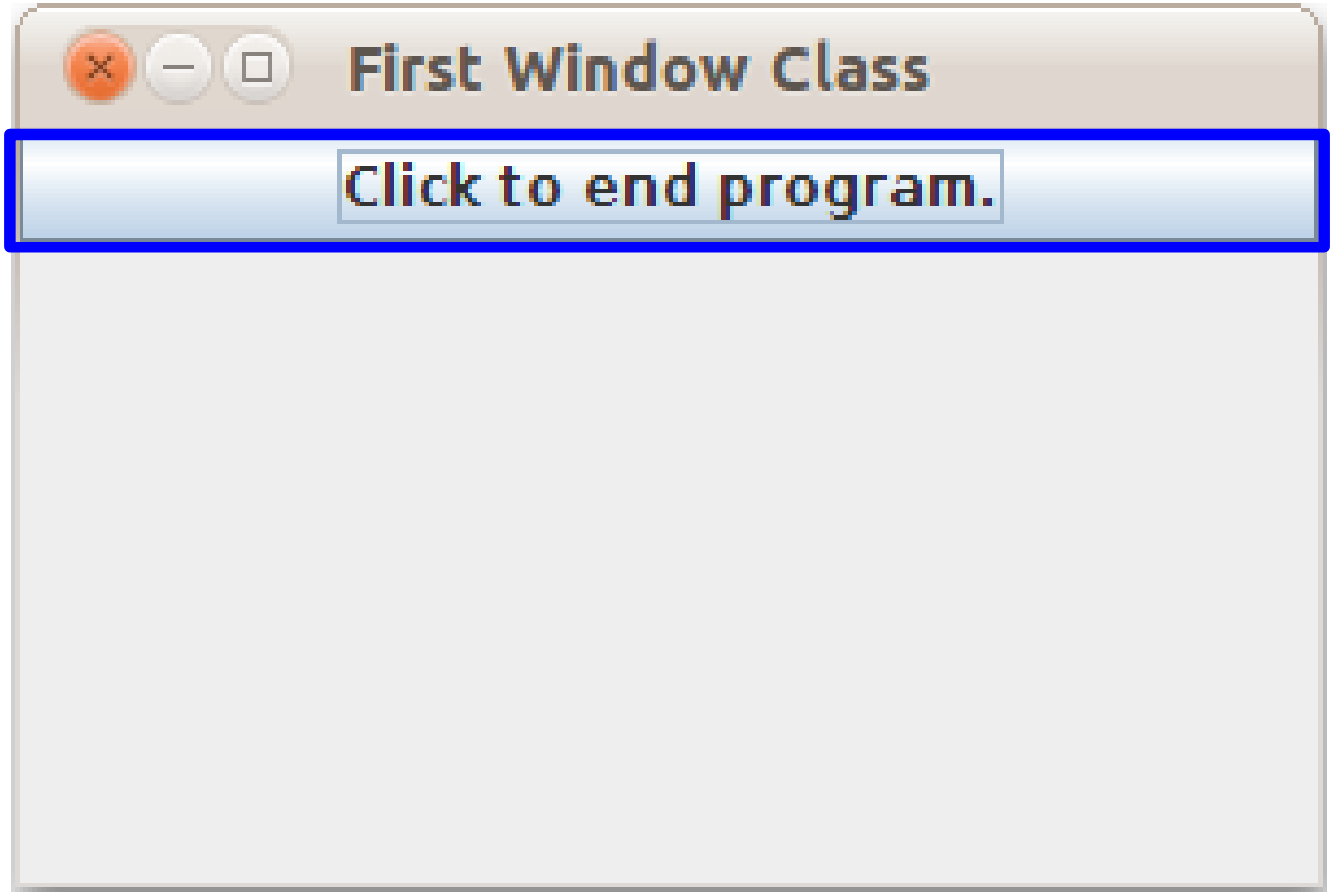
Basic GUI application

Coloured window



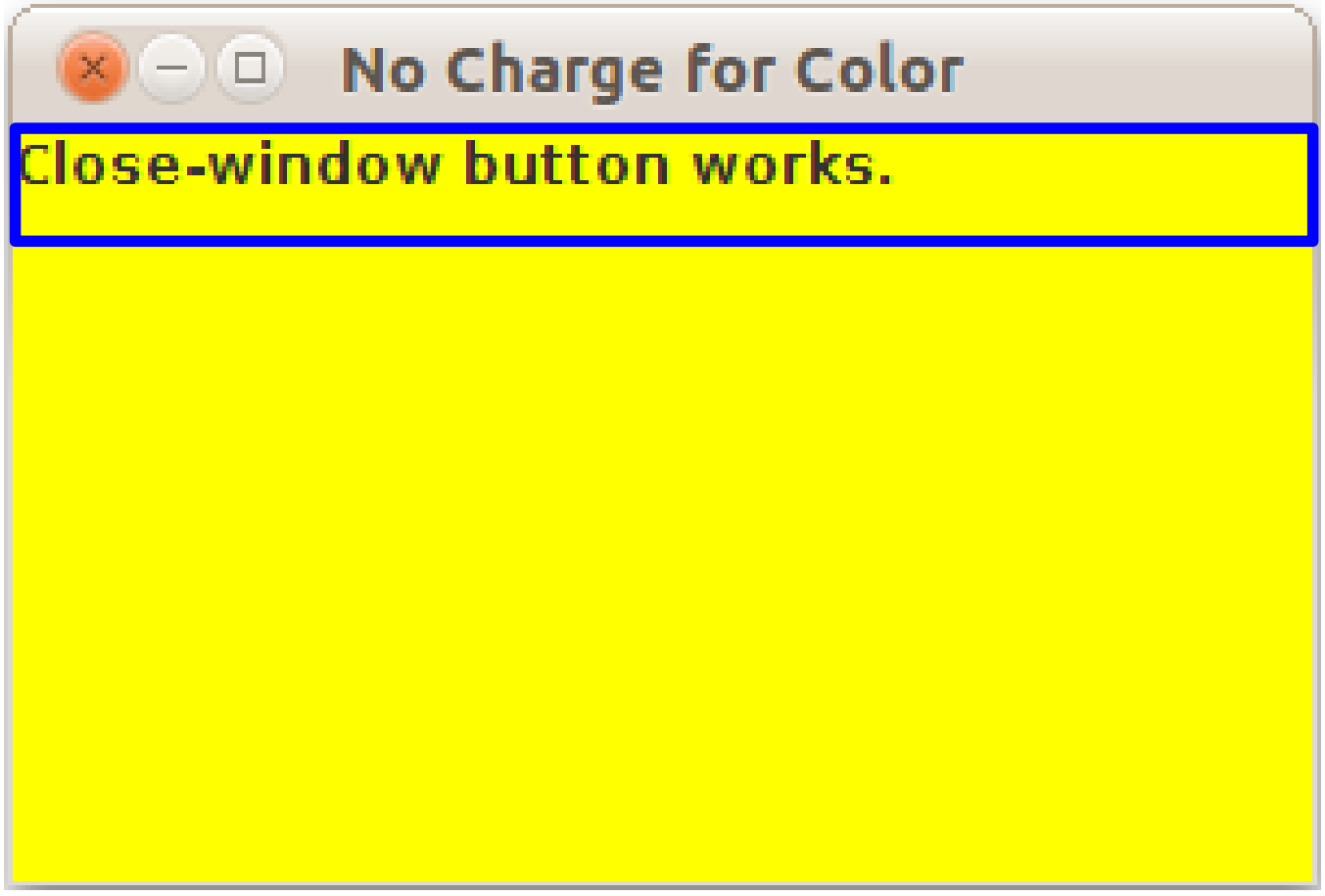
Basic GUI application

Button



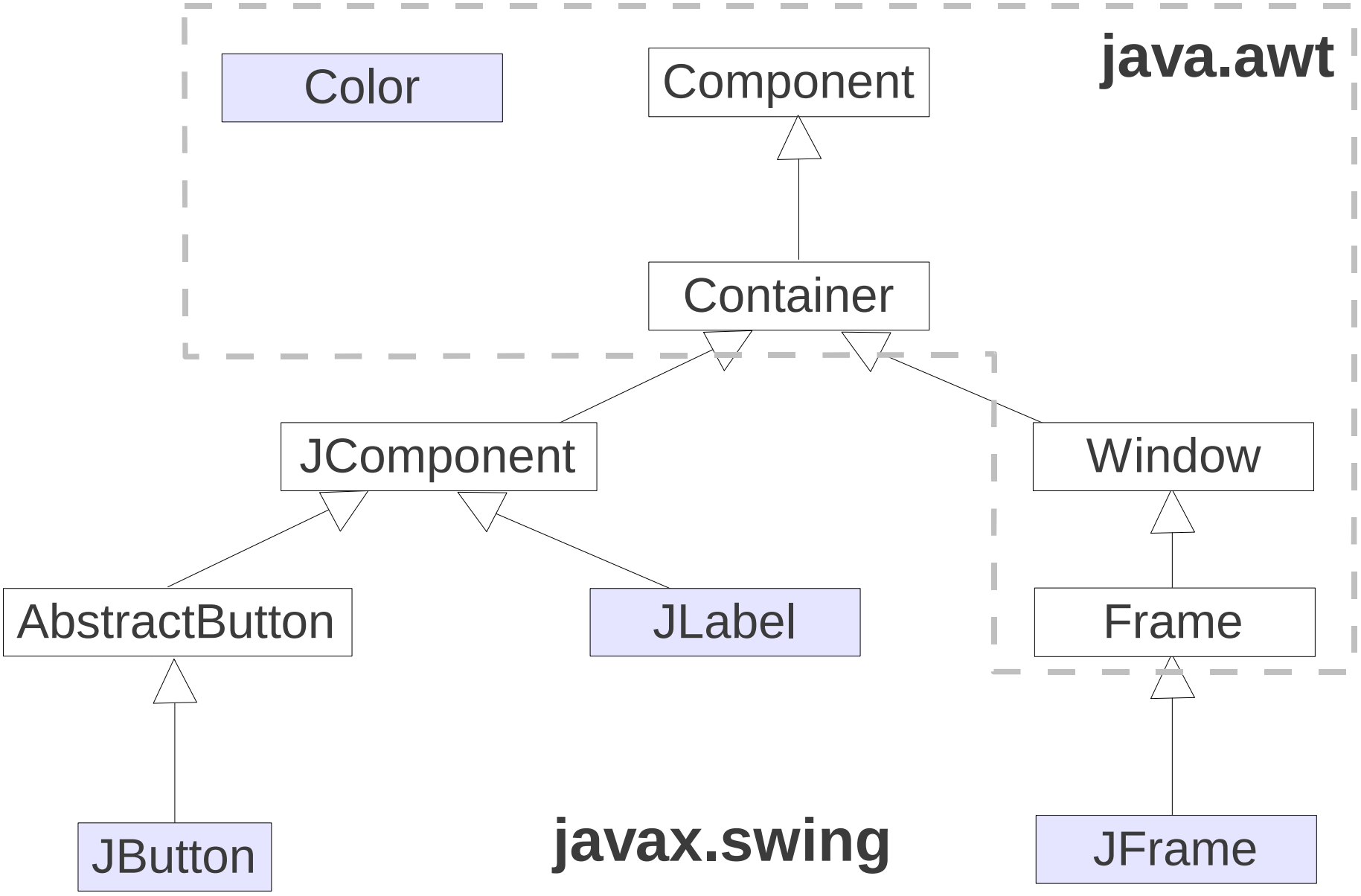
Basic GUI application

Label



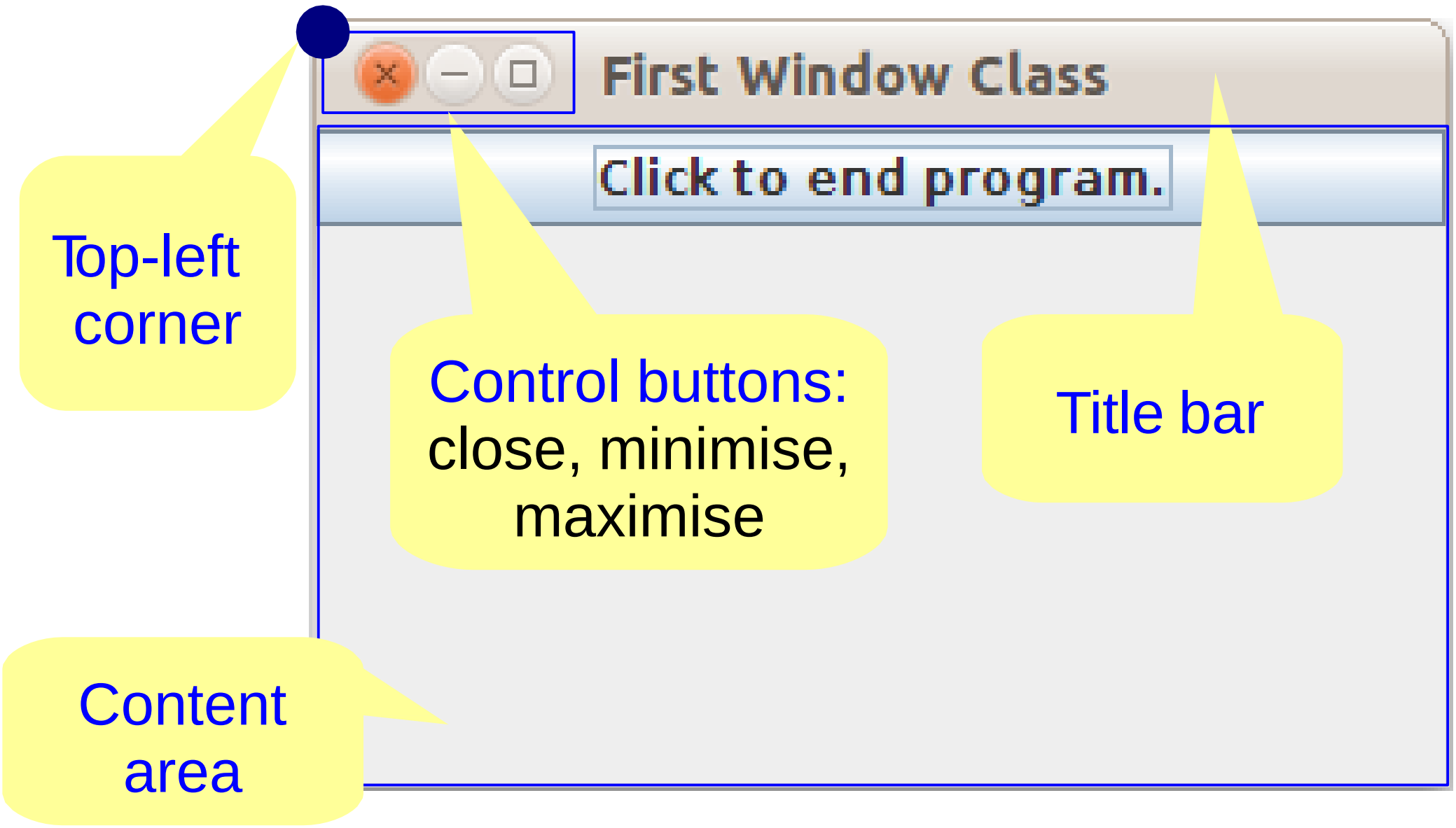
Basic GUI application

Component class hierarchy



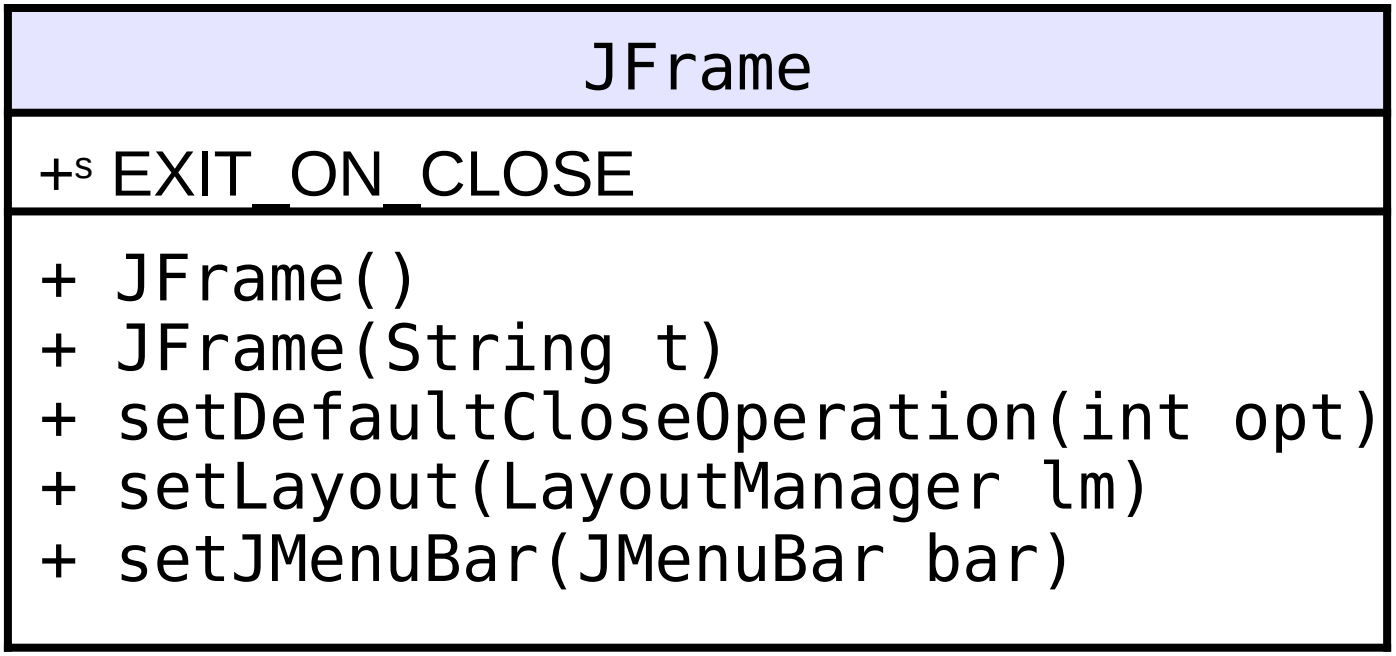
Basic GUI application

Window basic features



Basic GUI application

Window class



Windows class

Inherited methods

java.awt.Component
+ setSize(int,int) + getSize() + setBackground(Color) + setForeground(Color) + setLocation(int,int) + setVisible(boolean) + isVisible(): boolean

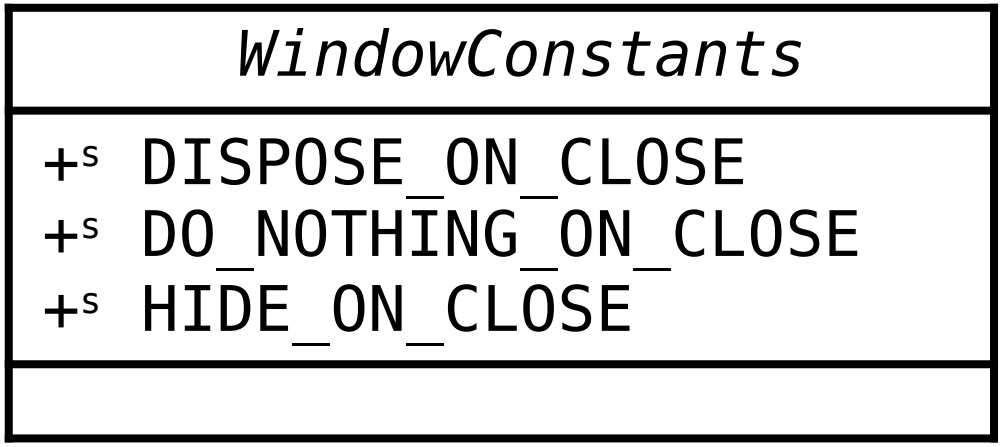
java.awt.Window
+ dispose()

java.awt.Frame
+ setTitle(String t) + getTitle(): String

java.awt.Container
+ add(Component)

Windows class

Inherited fields



Basic GUI application

Task 1: Create a window object

- **Directly** using the `JFrame` class:

```
JFrame w = new JFrame();
JFrame w1 = new JFrame("First Window Class");
```

- **Indirectly** via a sub-class of `JFrame`:

```
public class FirstWindow extends JFrame
{...}
JFrame w = new FirstWindow();
```

Task 2: Set up the window

- Set up window properties:
  - title
  - size: width and height
  - location: (x,y) coordinates of top-left corner
  - colour: fore/background
- Use the corresponding mutators:

```
w.setTitle("First Window Class");
w.setSize(300,200);
w.setLocation(100,100);
w.setBackground(Color.YELLOW);
```

Task 2: Set up the window

Set up colour

- A colour is an object of `java.awt.Color`
- Colour components (red, green, blue) can be specified as input
- `Color` has pre-defined colours, defined as public constants, e.g:

`BLACK`	`RED`	`ORANGE`	`...`
`BLUE`	`PINK`	`GREEN`	
`CYAN`	`YELLOW`	`GRAY`	

Task 3: Handle window events (1)

- API package: `java.awt.event`
- Window events are objects of `WindowEvent`
- Basic event listener is `WindowListener` interface:
  - defines methods for each window event (opened, closing, closed, activated, etc.)
  - methods take a `WindowEvent` object as argument
- Helper class `WindowAdapter` implements `WindowListener` to ease coding

Task 3: Handle window events (2)

- Three ways to implement event handlers:
  - implement `WindowListener`
    - must declare all methods
  - extend `WindowAdapter`
    - only declare methods that need to be overriden
    - only applicable if class is *not* in multiple inheritance
  - invoke a helper method:
    - for closing event only
    - no code, just invoke

Task 3: Handle window events

Example: using listener interface

```
public class MyWindowListener implements
                                        WindowListener {
  public void windowOpened(WindowEvent e) {}
   public void windowClosing(WindowEvent e) {
     // handling code
     System.exit(0);
   }
  public void windowClosed(WindowEvent e) {}
  public void windowIconified(WindowEvent e) {}
  public void windowDeiconified(WindowEvent e) {}
  public void windowActivated(WindowEvent e) {}
  public void windowDeactivated(WindowEvent e) {}
}
```

Task 3: Handle window events

Example: using adapter

```
public class MyWindowListener extends
WindowAdapter {
   public void windowClosing(WindowEvent e) {
      // handling code
      System.exit(0);
   }
}
```

Task 3: Handle window events

Example: using helper method

```
JFrame w = new JFrame("my window");
w.setDefaultCloseOperation(
                        JFrame.EXIT_ON_CLOSE)
// w.setDefaultCloseOperation(
//                    JFrame.DISPOSE_ON_CLOSE)
// w.setDefaultCloseOperation(
//                        JFrame.HIDE_ON_CLOSE)
// w.setDefaultCloseOperation(
//                JFrame.DO_NOTHING_ON_CLOSE)
```

Task 4: Create a component

- Import the component class
- Task 4.1: Create an object using a suitable constructor
- Example: button & label

```
JButton endButton = new JButton("Click to
  end program");

JLabel aLabel = new JLabel("Close-window
  button works");
```

Task 4: Create a component

Task 4.2: Setting up a component

- Set up the component's properties:
  - size
  - text: label or default text (for text fields)
  - icon (for button-type components)
  - fore/background colours
  - etc.
- Use the corresponding setters:

```
endButton.setText("Click to end program");
aLabel.setText("Hello world");
```

Task 4: Create a component

Task 4.3: Handle component events

- Event types are based on the component hierarchy:
  - (super) component: mouse, key, focus, etc.
  - container: component addition/removal
  - component-specific: **action**, change, item
- Action events are objects of `ActionEvent`
- Action event listener: `ActionListener`
  - event handling methods:
    `actionPerformed(ActionEvent e)`

Task 4.3: Handle component events

Handle button's action event (1)

- Corresponds to the button's pressed action:
  - caused by a mouse click or pressing Enter key
- `ActionEvent`
  - source: button object
  - action command: the button's action name (can be different from the text)

Task 4.3: Handle component events

Handle button's action event (2)

- Basic steps:
  - Step 1: Create an event handler from `ActionListener`
    - using normal or anonymous class
  - Step 2: Create an event handler object
  - Step 3: Register the handler to listen for the button's action event

Task 4.3: Handle component events

Step 1,2: Create an event handler (1)

- Using a normal class:

```
public class EndingListener
             implements ActionListener {
   public void actionPerformed(ActionEvent e) {
      System.exit(0);
   }
}
...
ActionListener l = new EndingListener();
```

Task 4.3: Handle component events

Step 1,2: Create an event handler (2)

- Using an *anonymous* class:

```
ActionListener l = new ActionListener() {
  public void actionPerformed(ActionEvent e) {
    System.exit(0);
  }
};
```

Task 4.3: Handle component events

Step 3: Register the event handler

- Invoke the `addActionListener` method on the button
- More than one listeners of the same event can be registered to the same component
- All listeners of the event are notified when it occurs
- Example:

```
endButton.addActionListener(l);
```

Task 4.4: Add component to window

- Non-menu components are added using method `Container.add`:
  - (optional) an index position can be specified
- Example: add a button to a window

```
// to a default position
w.add(endButton);
// to a specified index position
w.add(endButton, BorderLayout.NORTH);
```

Task 5: Display the window

- Windows are not displayed on the screen by default
- To display/hide, invoke method `setVisible` on the window object:
  - to display: invoke with argument `true`
  - to hide: invoke with the argument `false`

Basic GUI applications

- Basic
  - `lect06.DemoWindow`
  - `lect06.DemoColoredWindow`
- More about window events:
  - `lect06.WindowEventDemo`

Summary

- GUI application consists of controller, view and model components
- User actions are handled via an interaction model
- Event driven programming for GUI applications
- Java provides AWT and Swing API for GUI programming
- A basic GUI application contains window, button, and label

References

Savitch W., Absolute Java, 6th, Pearson, 2015
- Chapter 17