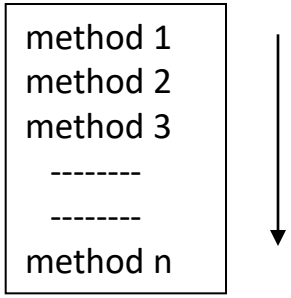
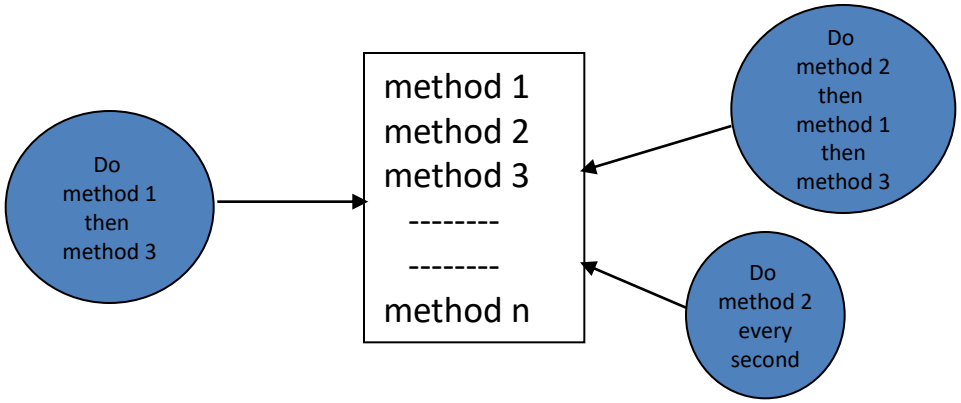
CS102 – GUI

Programming forms

- *Conventional/Procedural programming*
  - code is always executed in same sequence.
    characterised by input/process/output



- *Event-driven programming*
  - code is executed upon activation of events.
    sequence changes depending on order of events



GUI using AWT

- AWT - Abstract Window Toolkit
- Must base
  - desktop programs on **Frame**
    - constructor, paint, …
  - browser programs on **Applet**
    - init, start, paint, stop, destroy, …
- Can convert, but
  - better to base code on **Panel**
  - then it add to Frame or Applet

GUI using AWT

- Two steps

  (1) Create the interface
  - By add components & containers
  - & using layout managers

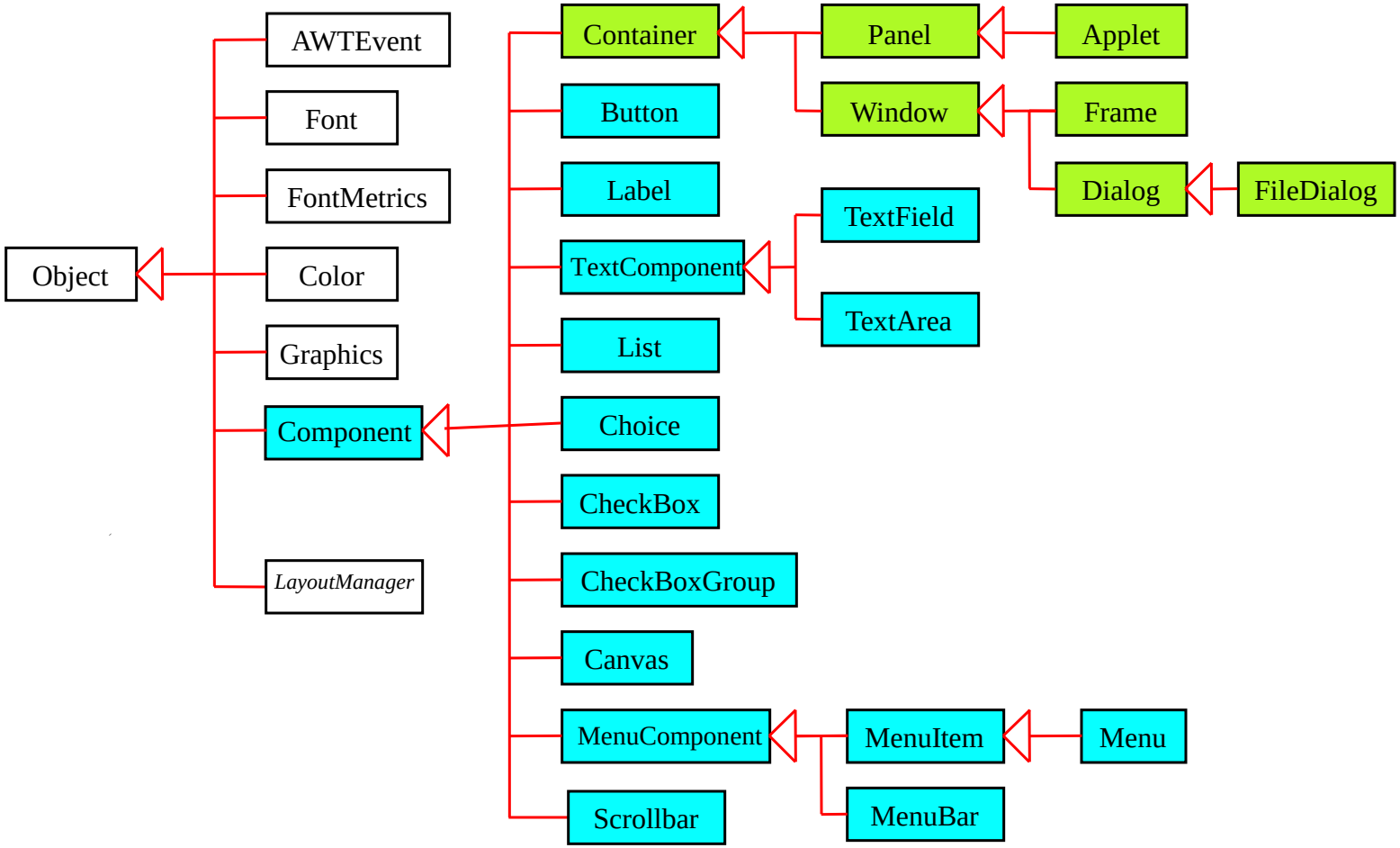
  (2) Add interaction
  - Create event listeners
  - & “Wire-up” events

(1) Create the interface…

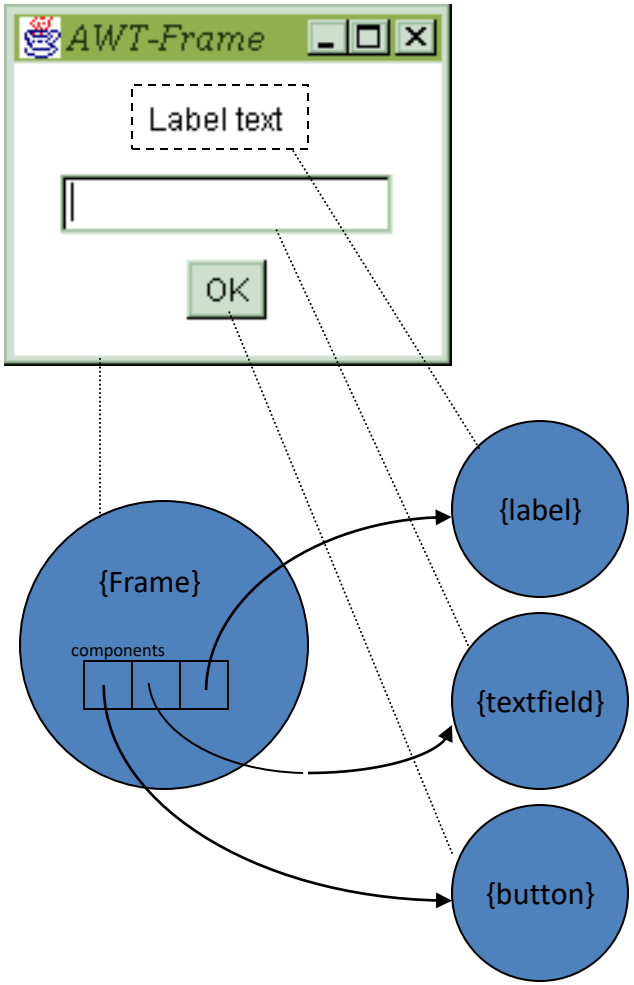
AWT Applications - Frame

- Frame is a container for components

AWT classes


Object
AWTEvent
Font
FontMetrics
Color
Graphics
Component
LayoutManager
Container
Button
Label
TextComponent
List
Choice
CheckBox
CheckBoxGroup
Canvas
MenuComponent
Scrollbar
Panel
Window
TextField
TextArea
MenuItem
MenuBar
Applet
Frame
Dialog
FileDialog
Menu

Understanding the GUI



- UI-containers
  - have list of UI-components
- Each UI-component
  - is a class
  - with paint method
  - & lists of Event listeners

Setting up the GUI

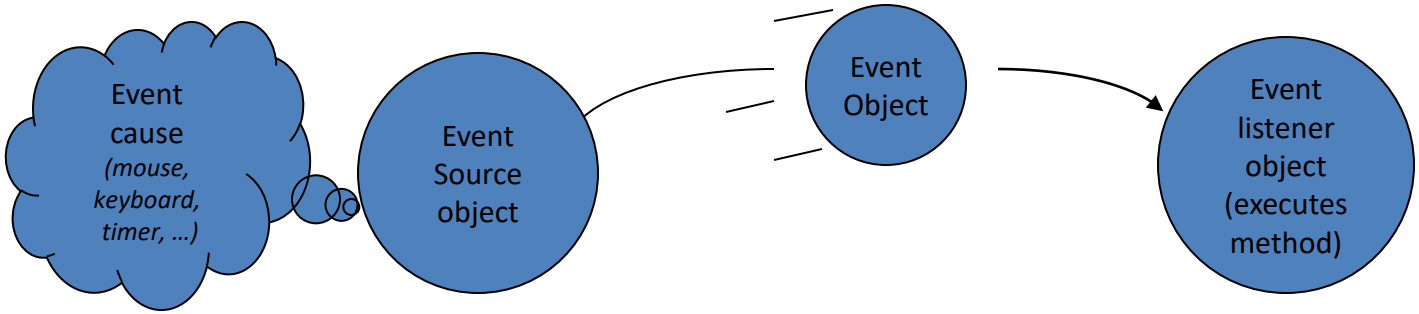
- Extend Frame class
  - In constructor
    - Create instances of containers & add them to Frame
    - Create instances of components & add them to containers or Frame
  - Possibly override paint method

Hint: Employ layout managers to arrange components in containers

- UI-components added to components list
- Painting Frame
  1. paints Frame borders
  2. calls Frame paint method
  3. calls paint method of each object in component list

(2) Add interaction…

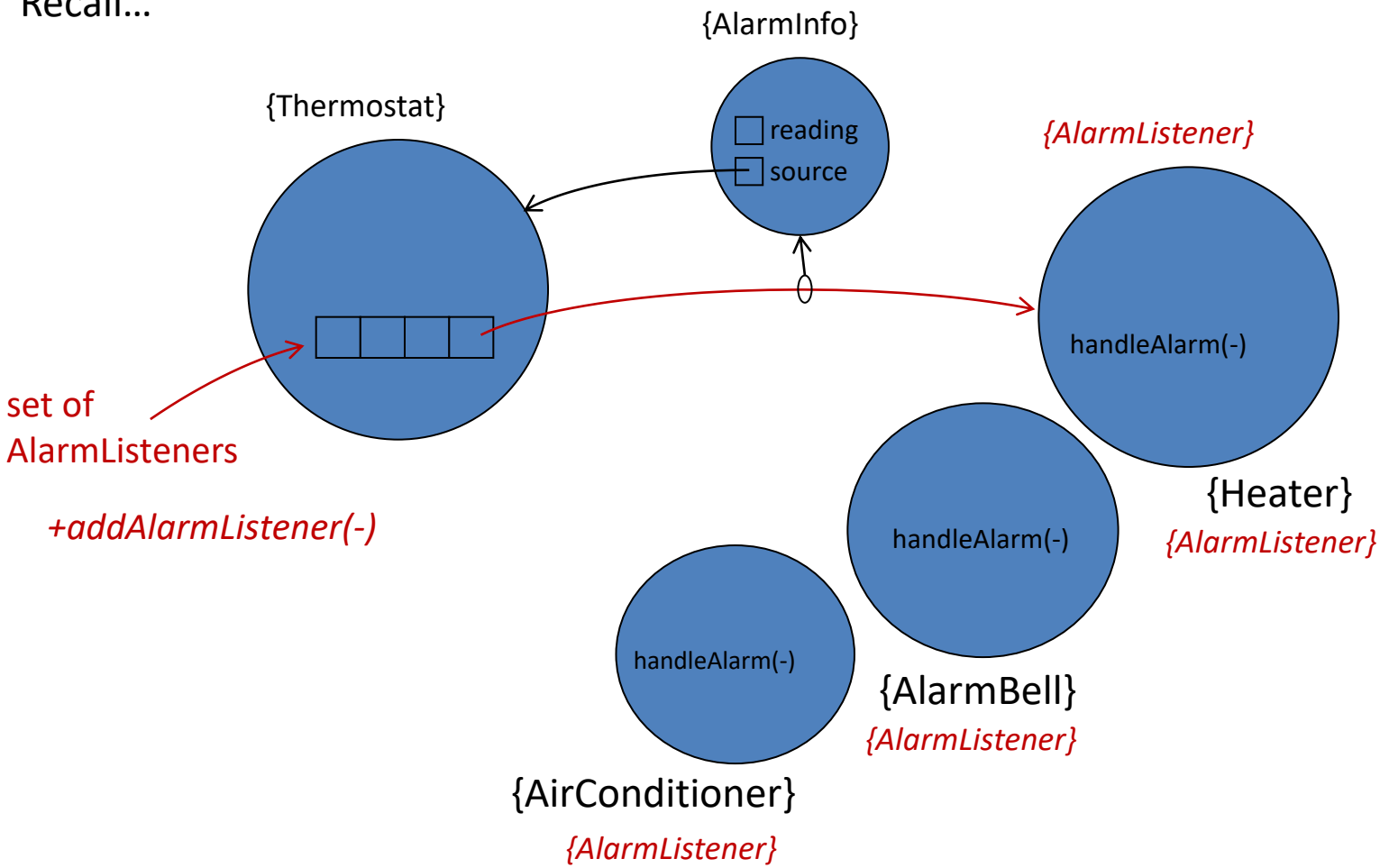
Events & Event Handling



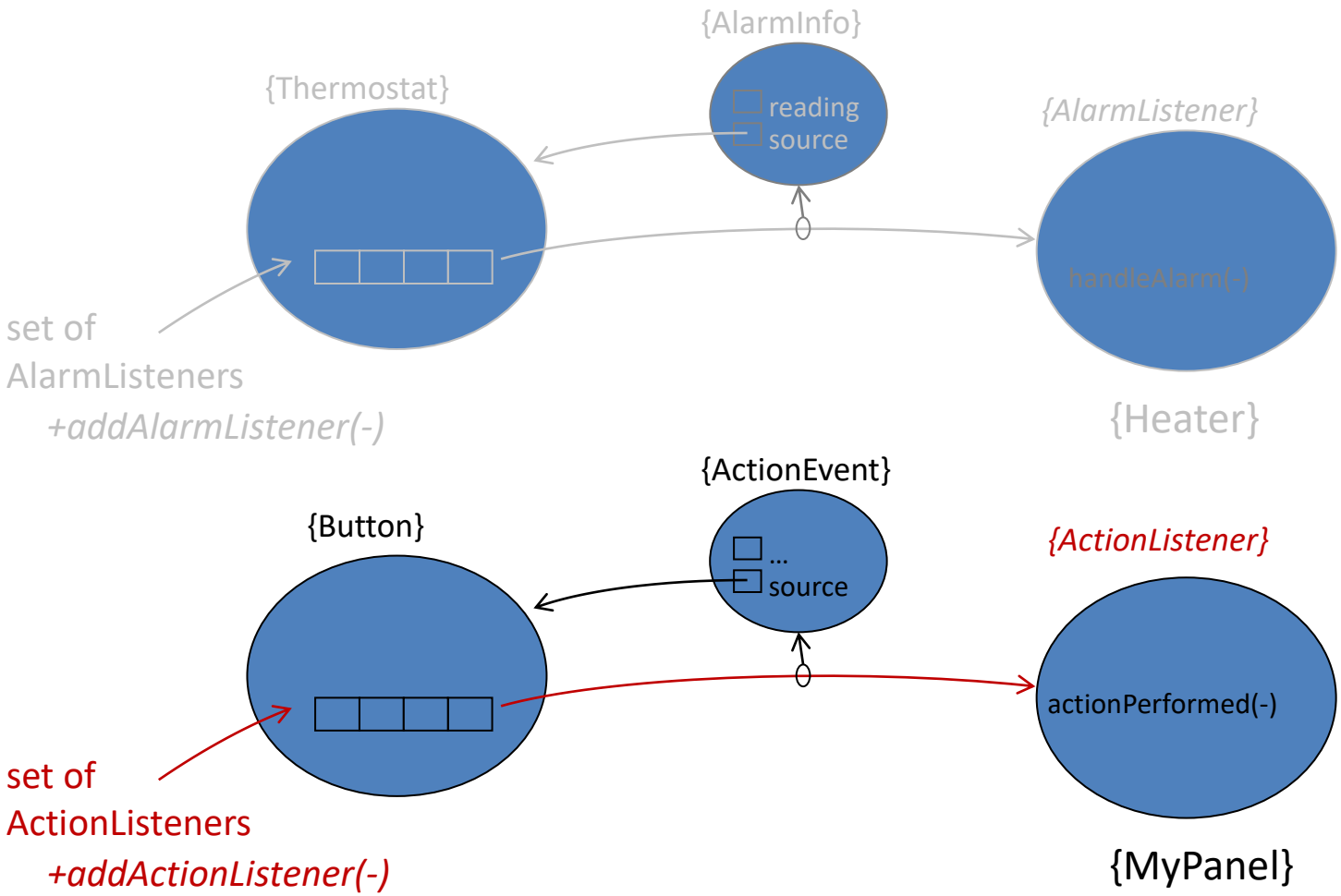
- Example…
  - User clicks on a button
  - Button is source of event object
  - Event object passed to associated listener object
  - Listener object executes associated method to perform desired task (save file, quit program, …)

Event handling…

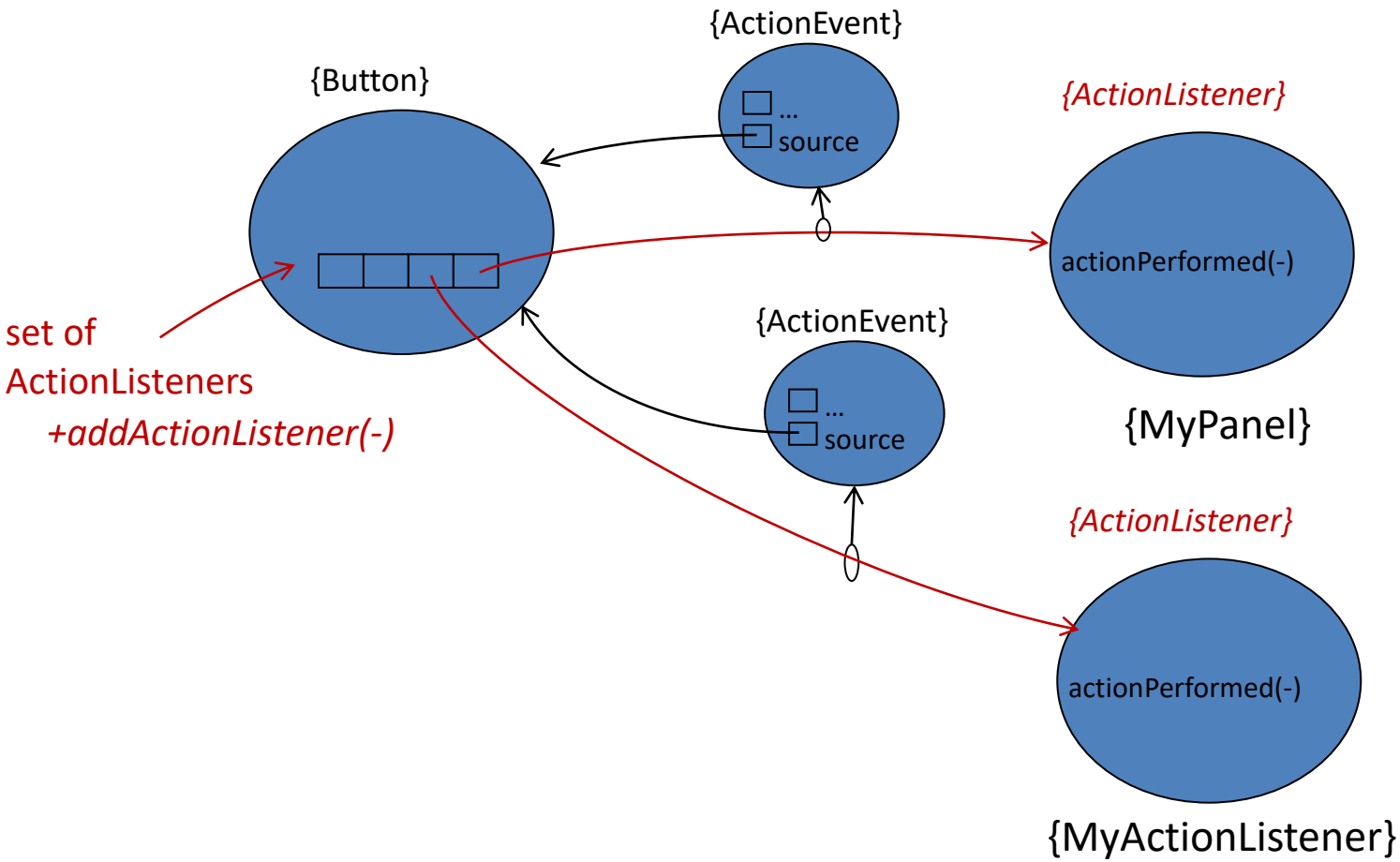
Recall…



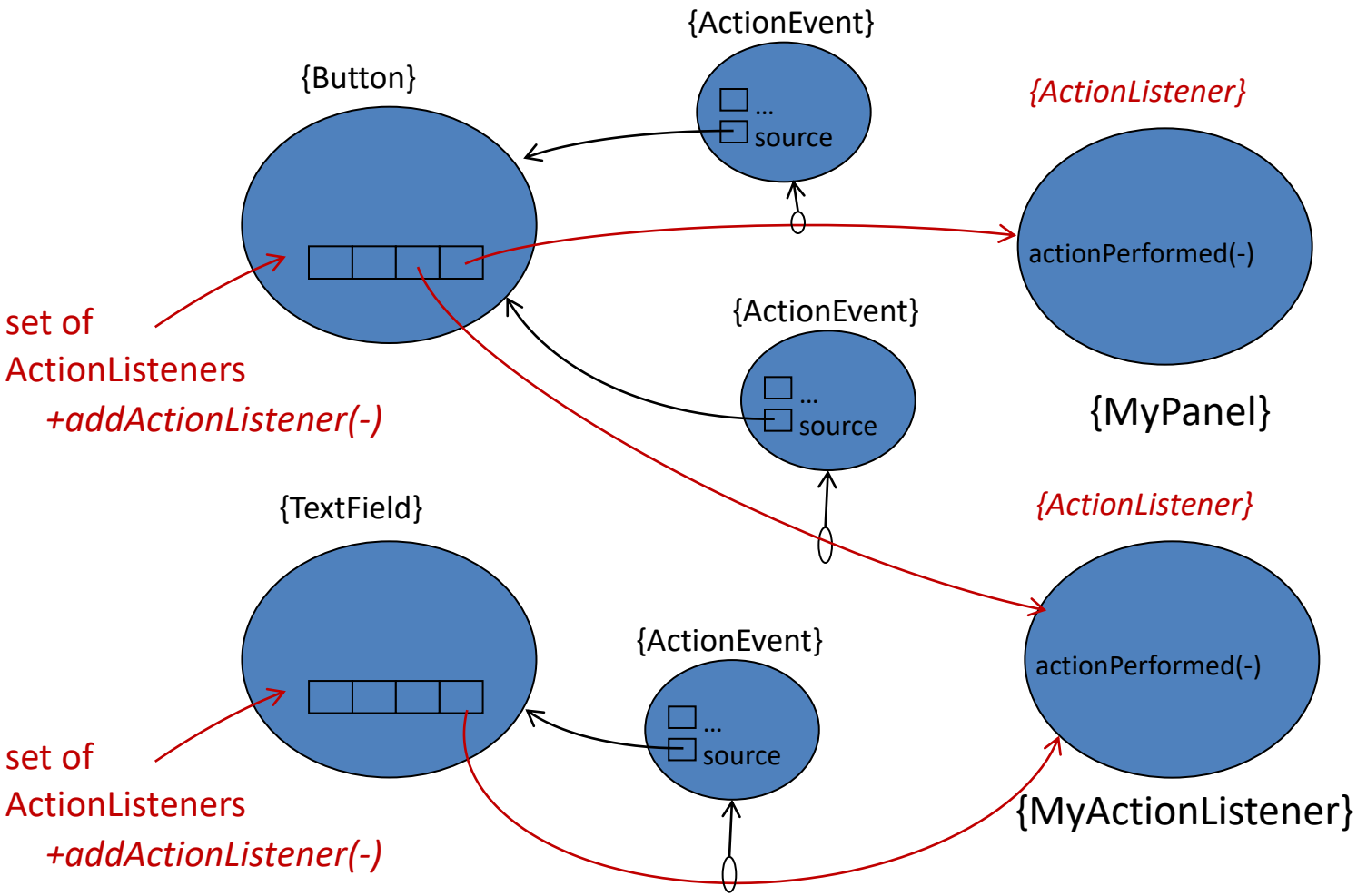
Event handling…



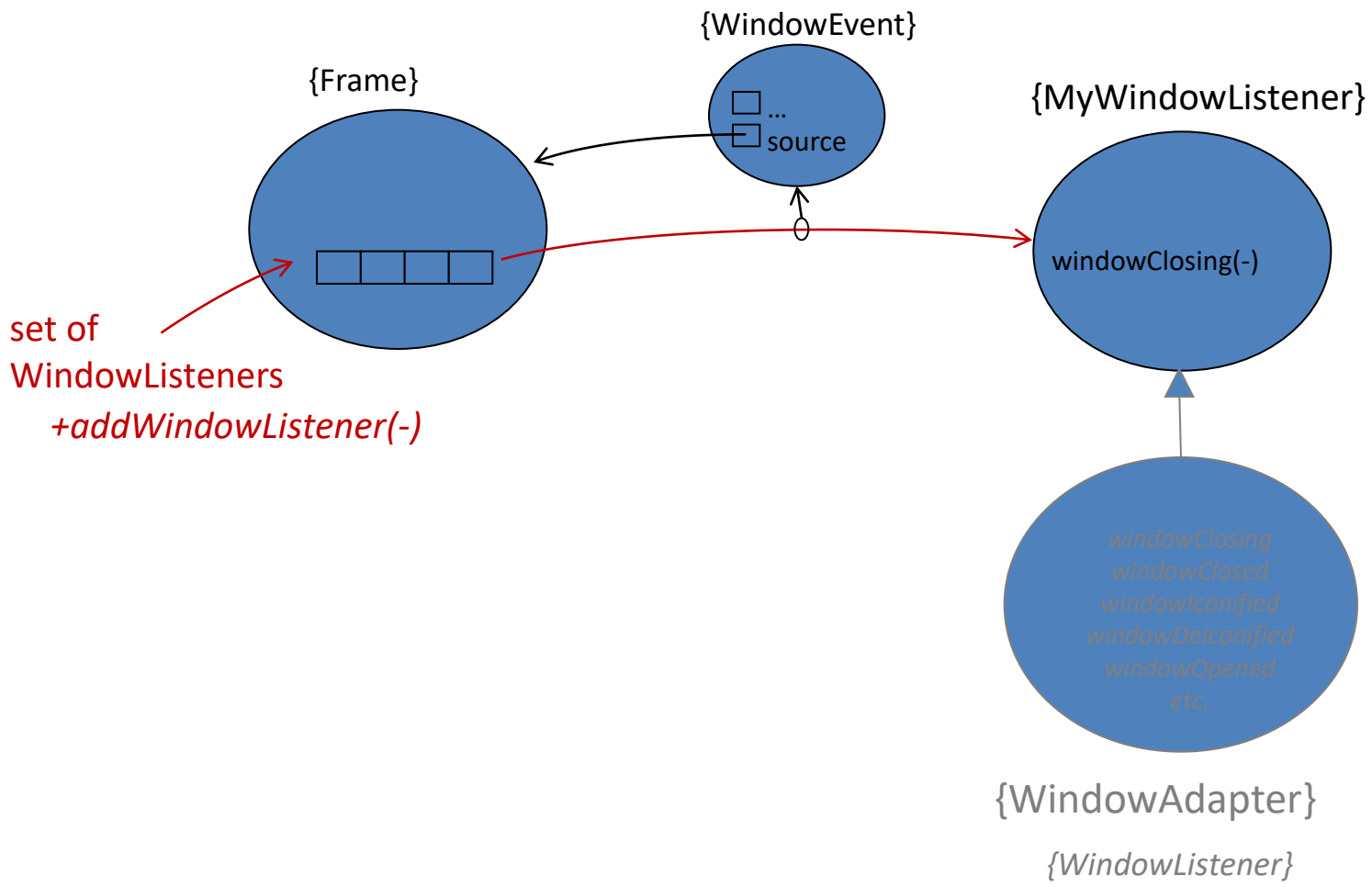
Event handling…



Event handling…



Event handling…



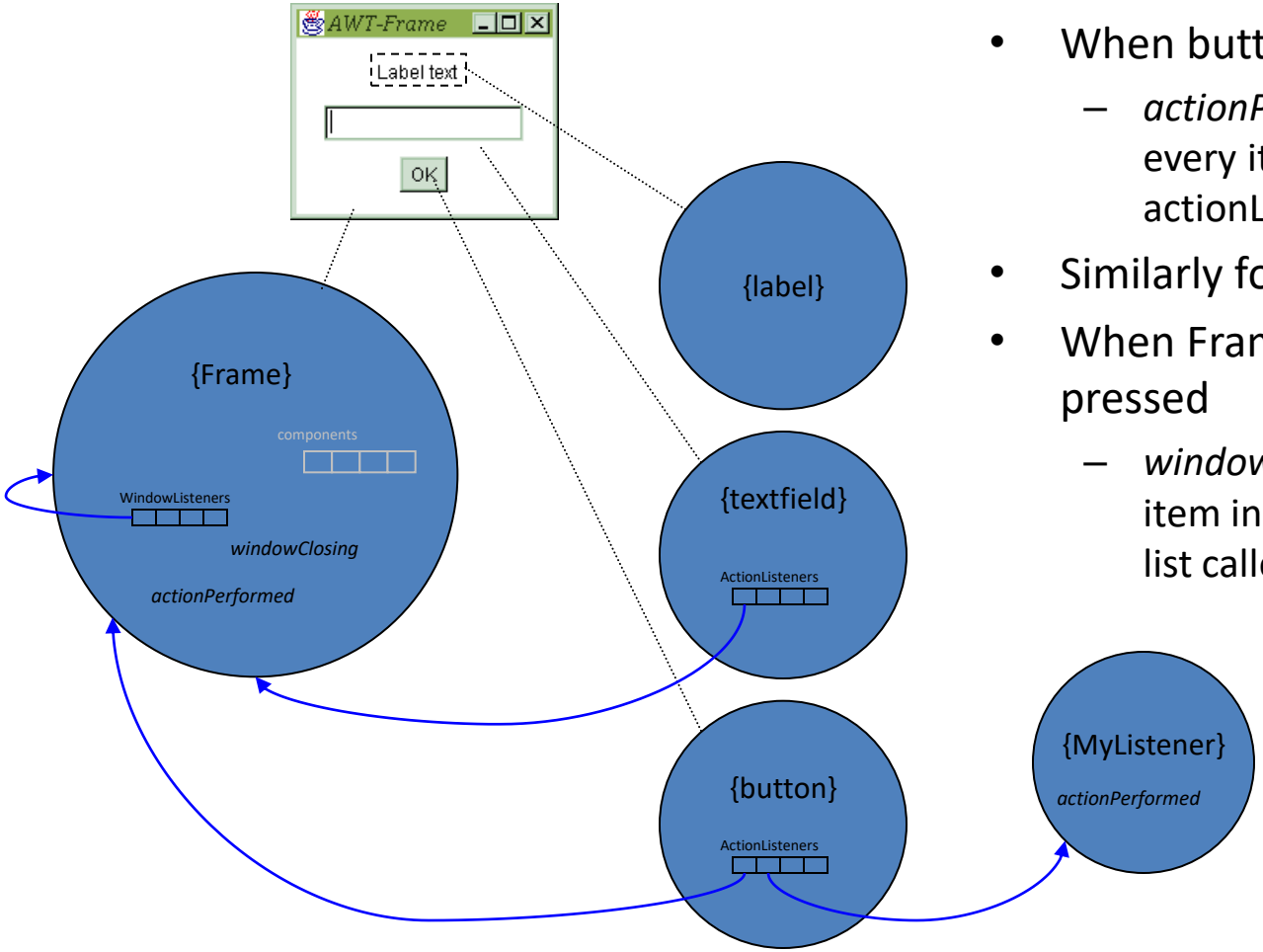
Setting up Event Handling

- Create listener class
  - Using new or existing class, *simply*
  - Implement desired event listener interface
  - Putting code for desired action in its methods

- In application *(e.g. Frame)*
  - Create instance of listener class
  - Add as listener of source object

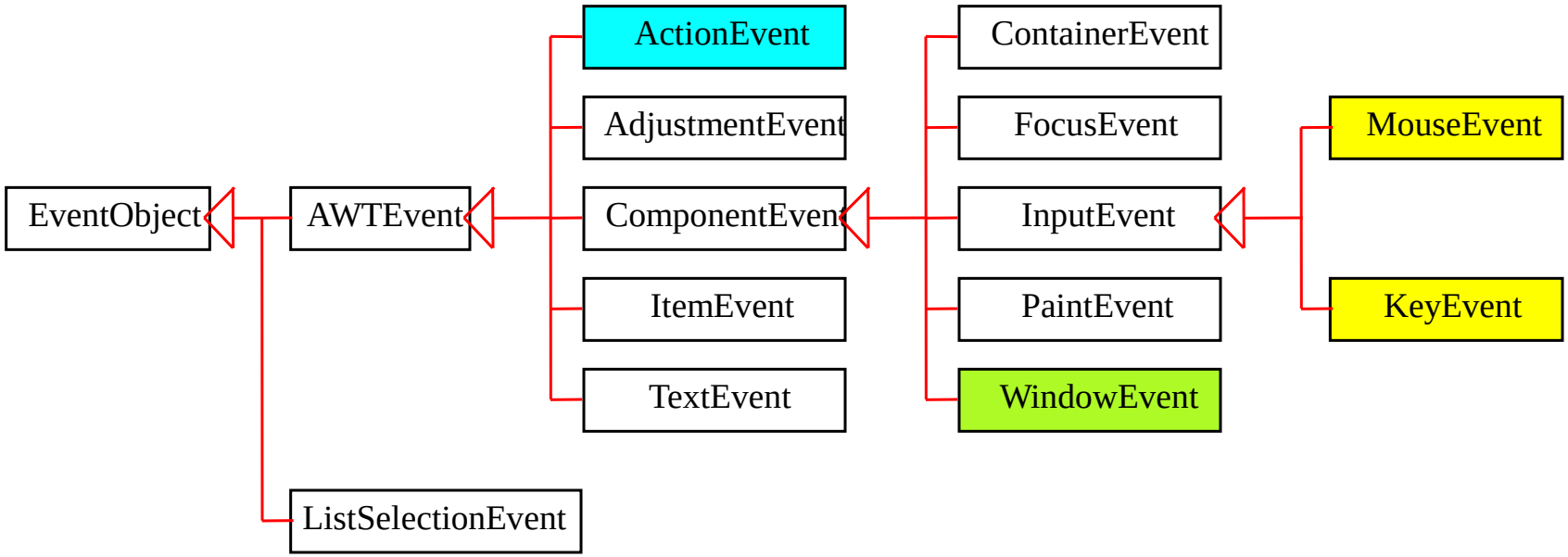
- can have any number of listeners for each event
- Source & listener can be same object!

Understanding Events



- When button is pressed
  - *actionPerformed* method of every item in button's actionListeners list called
- Similarly for textfield
- When Frame close button is pressed
  - *windowClosing* method of every item in Frame's windowListeners list called

Event Classes



GUI USING SWING

GUI using Swing

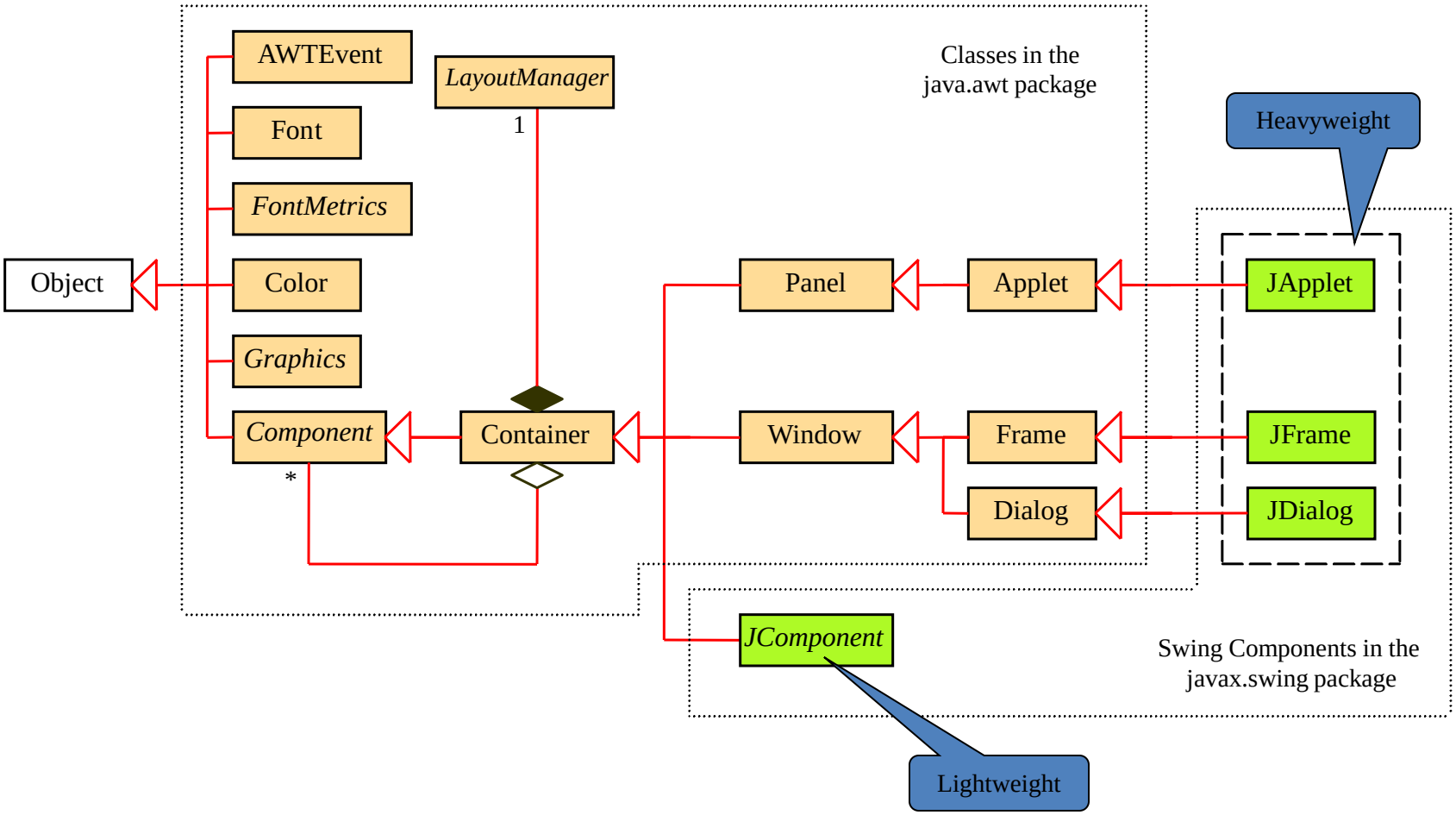
- Advantages
  - OS independent
  - Prettier!
  - More sophisticated components & options
    - Pluggable “Look & feel”
    - Borders, Tooltips, etc.
    - Drag ‘n Drop
    - File & ColorChoosers, Tables, editors, etc.

- Conceptually same as AWT
- Still uses AWT events package

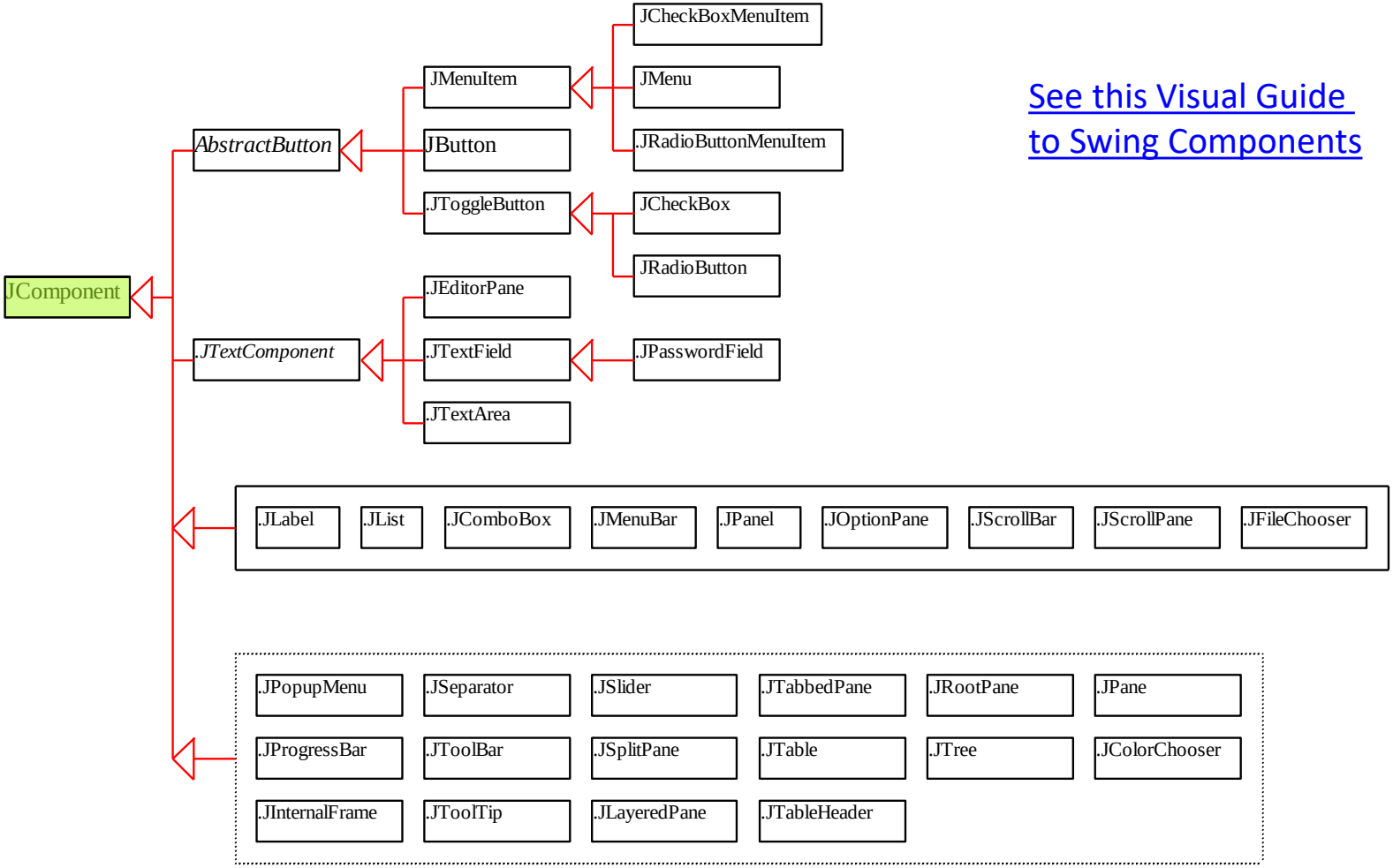
GUI using Swing

- Few differences (from AWT)
  - Use javax.swing package
    (equivalent Swing components start with “J”)
  - Frames can close automatically
    (well sort of…!)
  - Add components to JFrame’s contentPane
    (v1.5+ no longer explicitly needed)
  - Override paintComponent, not paint
    (except for Jframe, JApplet & JDialog)
    (also, must call super.paintComponent)

AWT & Swing classes



Swing - JComponents



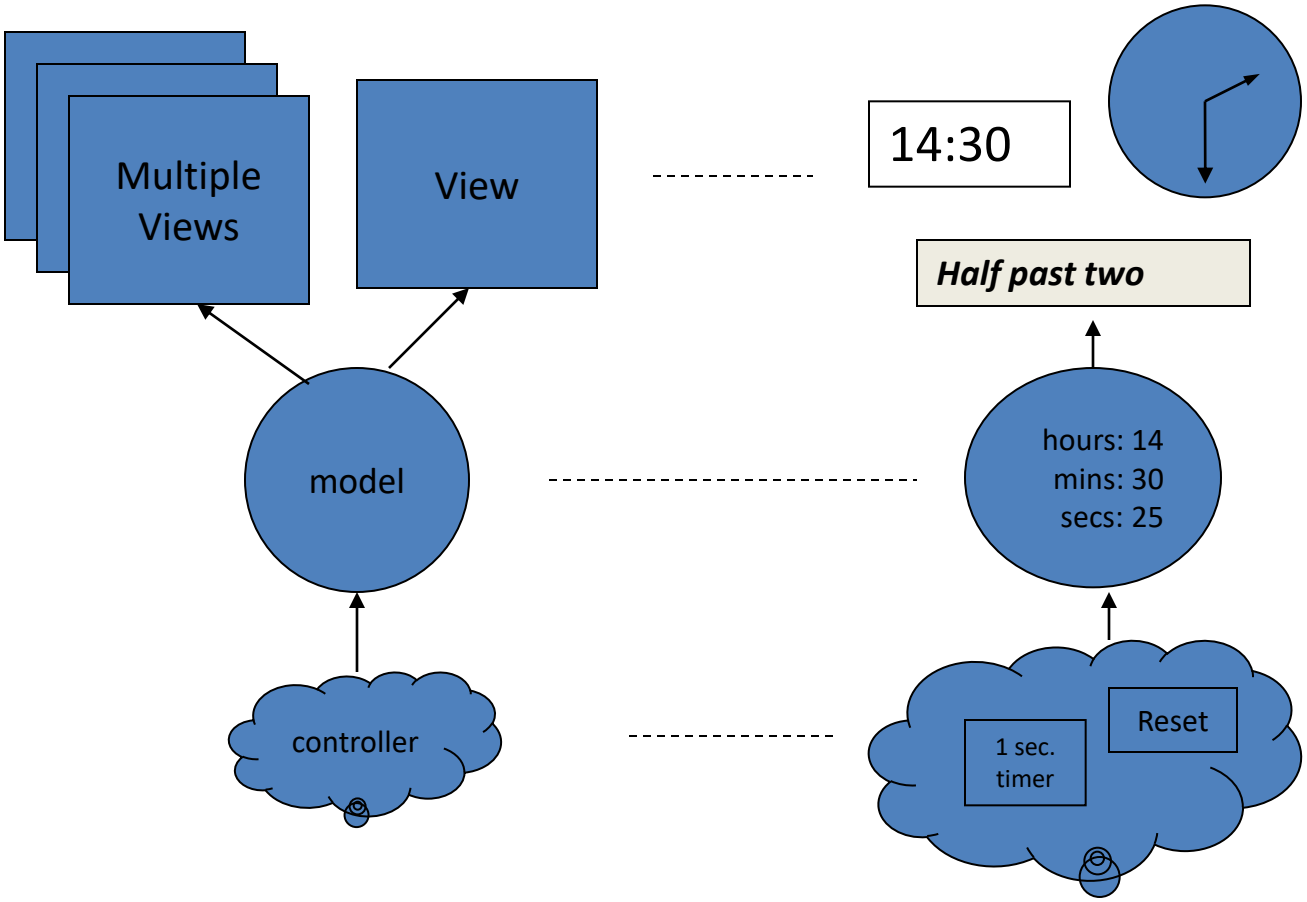
See this Visual Guide to Swing Components

DESIGNING GUI’S…

Design Tips

- GUI code can get very messy
  - Do not put everything in one class (as many Visual IDE's do)
  - Quick & dirty = impossible to change!
  - Employ design patterns, e.g. MVC
- Think Design first...

MVC - Design Pattern



Design Tips

- Think & design first
  - Use layout managers
  - Use OOP
    - What do you want?
    - What existing class is closest?
    - Extend it!

- begin with Panel rather than Frame/Applet
- then add instance(s) to whichever you want

- digital clock view
  - centered text in plain box, extend label
- analogue clock view
  - graphics in plain box, extend panel

Alarm Clock

Product Idea…

- An alarm clock!

Brainstorm requirements…

Design User Interface…

Detailed Design…

Implement & Test…

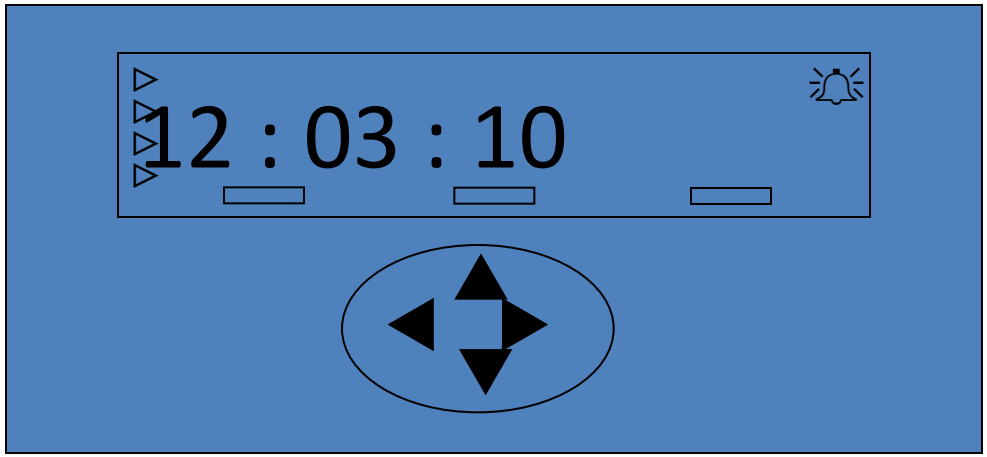
Brainstorm requirements…

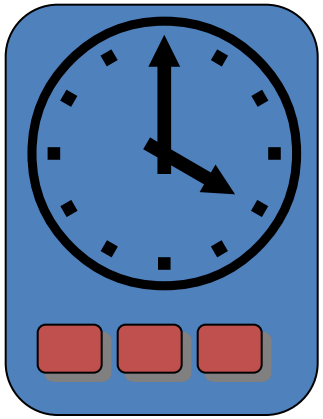
- Show time – hours/mins/secs
- 12 hour (am/pm) and/or 24 hour format
- Set alarm time – hours/mins/secs
- Disable alarm
- Stop alarm ringing

- Show alarm time – hours/mins/secs
- Set time – hours/mins/secs

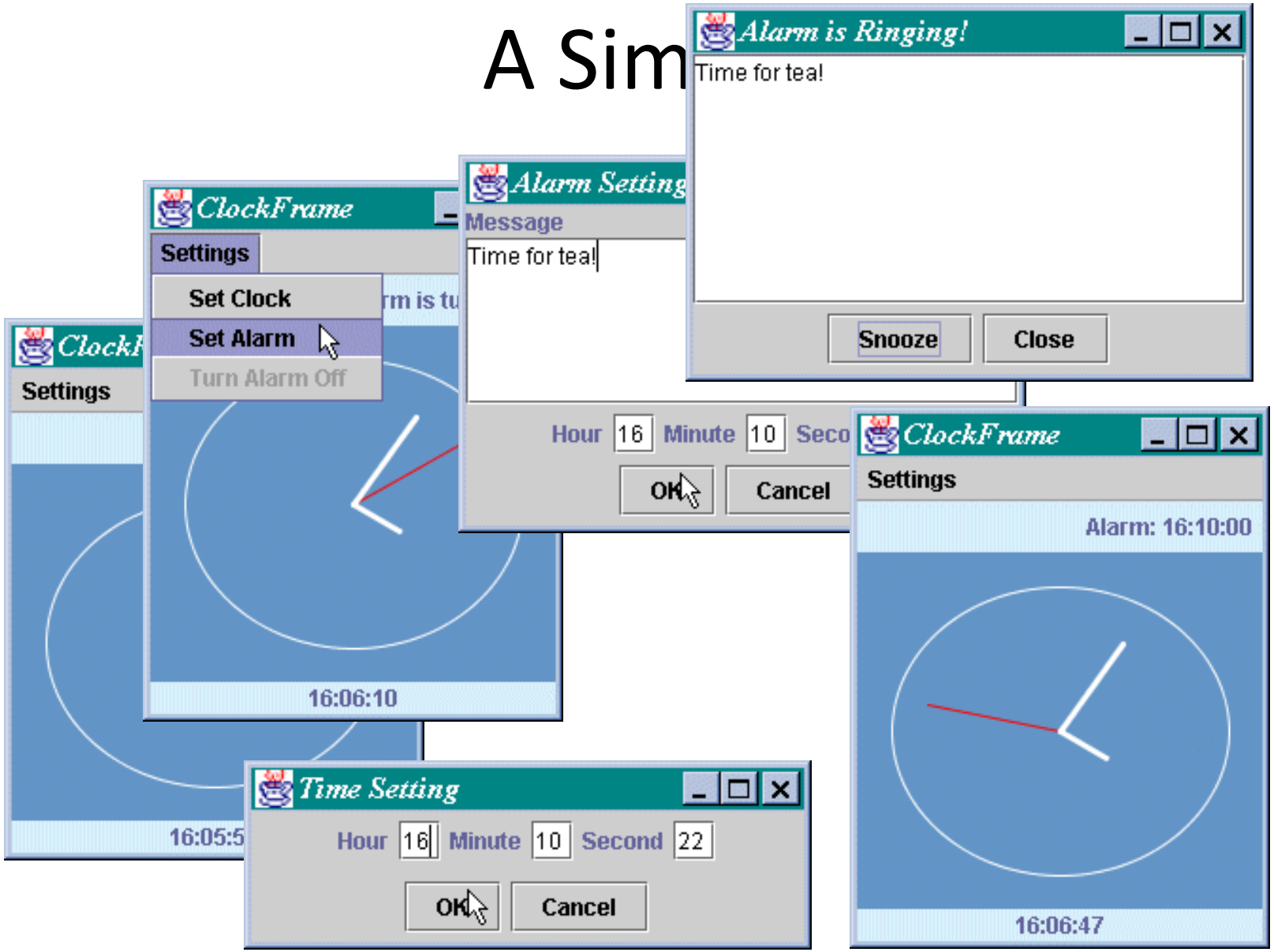
- Snooze option
- Multiple alarms, with different sounds
- Text description for each alarm
- International offsets for travelers

Design User Interface…

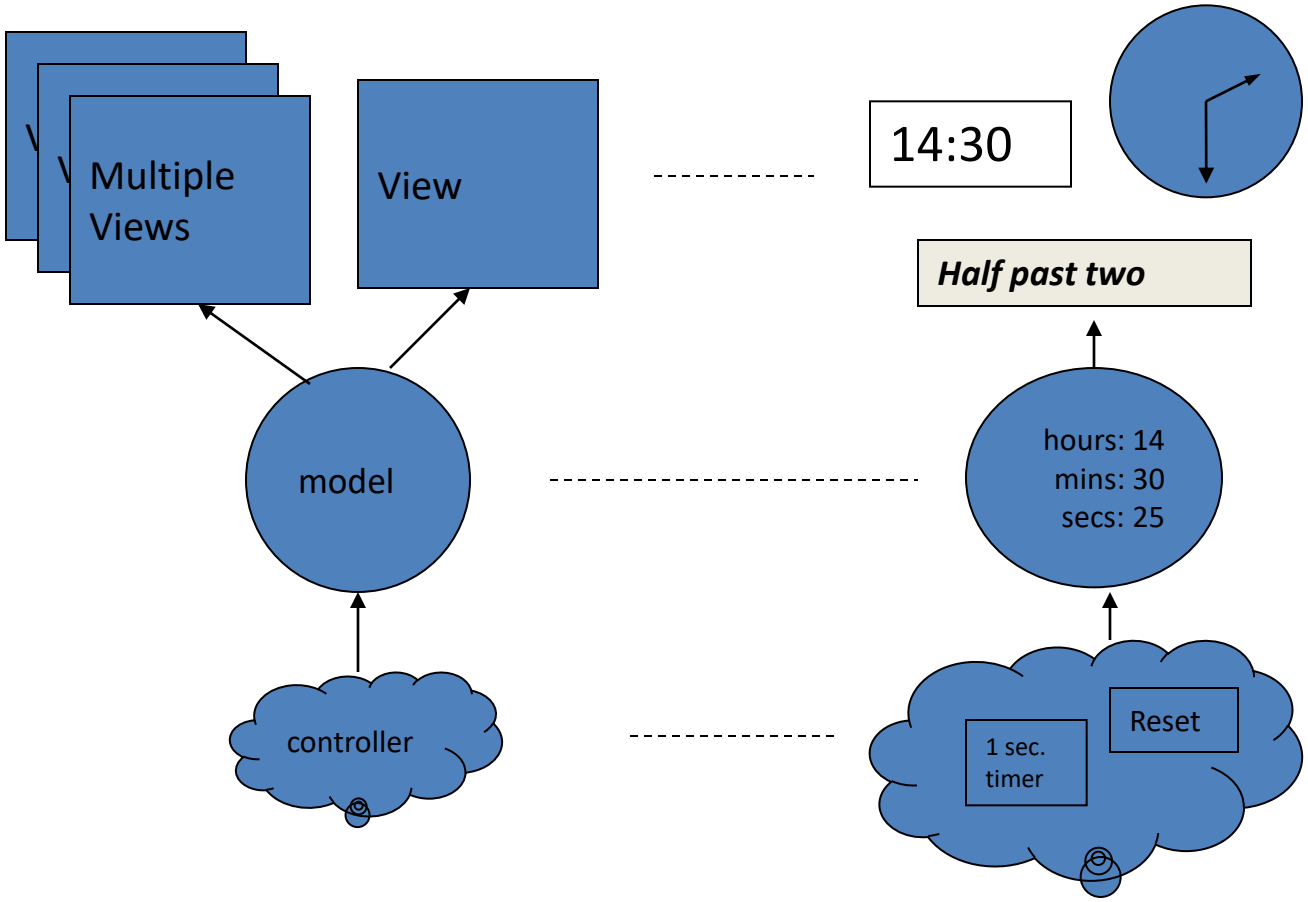




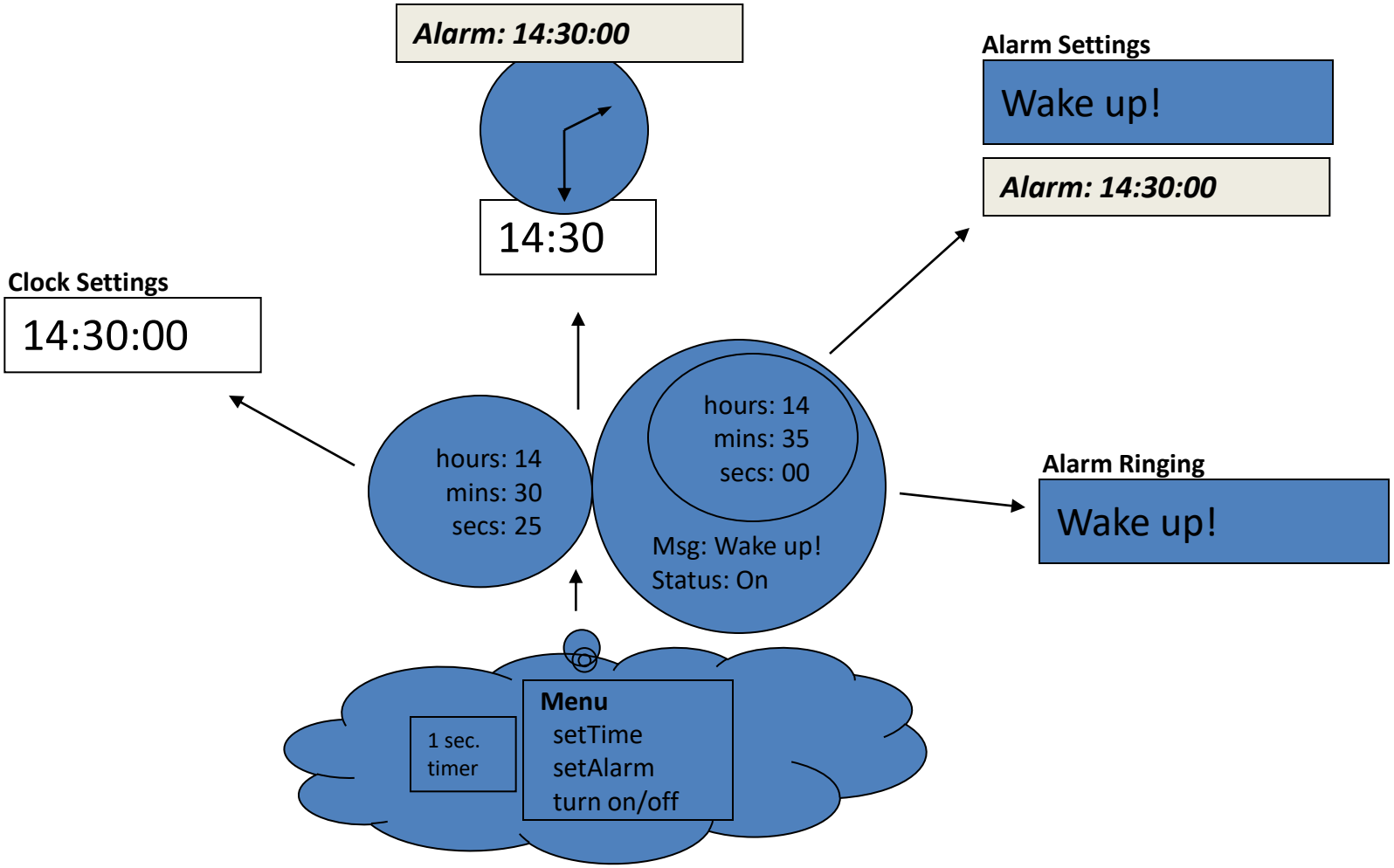
A Sim


Alarm is Ringing!
Time for tea!
Snooze
Close
ClockFrame
Settings
Set Clock
Set Alarm
Turn Alarm Off
16:06:10
Alarm Setting
Message
Time for tea!
Hour 16 Minute 10 Seco
OK
Cancel
ClockFrame
Settings
Alarm: 16:10:00
16:06:47
Time Setting
Hour 16 Minute 10 Second 22
OK
Cancel

MVC - Design Pattern

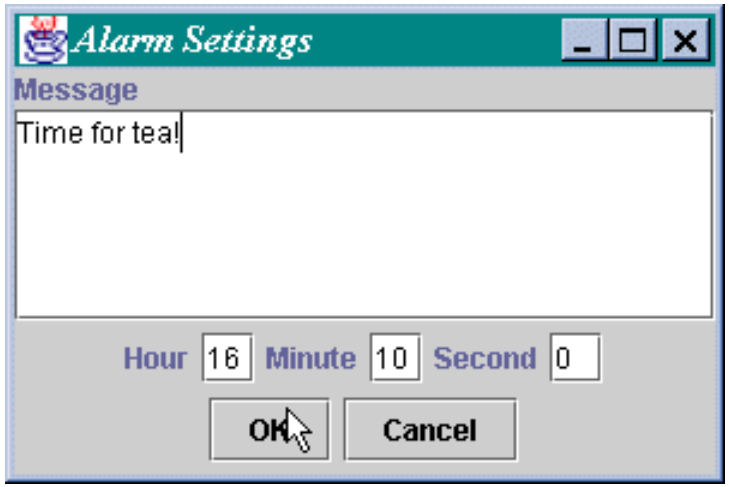


MVC – Alarm Clock



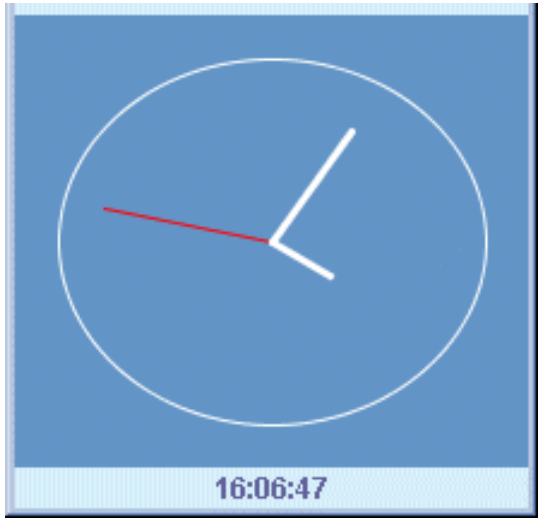
Alarm Class

- **Constructors**
  - Alarm(Time, Msg)
  - Alarm(Time)
- **Methods**
  - isAlarmTime(theTime)
  - getAlarmTime()
  - setAlarmTime(time)
  - getAlarmMsg()
  - setAlarmMsg(msg)
  - snooze()
  - isOn()
  - setStatus(status)



- **Alarm**
  - alarmTime
  - alarmMsg
  - status

AlarmClock Class



- **AlarmClock**
  - time
  - alarm

- **Constructors**
  - AlarmClock()
- **Methods**
  - turnAlarmOn()
  - turnAlarmOff()
  - isAlarmOn()
  - setAlarm(time)
  - setAlarmMsg(msg)
  - setTime(time)

  - ?getDisplayPanel()
  - ?update()
    - time.tick() & if isOn notify alarm listeners

Model View Controller

What is MVC

- *MVC is a design pattern for user interface programs.*
- *The controller changes the model which then informs/updates the view(s).*

Example

- Design a simple user-interface for a GUI application that will compute the circumference of a circle of given radius and also compute the radius given the circumference.
- Explain how you would "wire-up" the interface so that it functioned correctly.

Version 2

- Moving the knowledge of Circles out to a separate class allows it to be used elsewhere.
- Having radius & circumference properties is not normally a good idea, but in some cases such dependent properties are needed (if it takes too long to recompute, for example.)
- MVCa

Version 3

- Revised version using a reference to UI class
- MVCb

Version 4

- Revised version using Observable class to update ui & another observer (console)
- http://docs.oracle.com/javase/7/docs/api/java/util/Observable.html
- Observer Interface
- void update(Observable o, Object arg) This method is called whenever the observed object is changed. An application calls an Observable object's notifyObservers method to have all the object's observers notified of the change.
- MVCc