CS20004: Object Oriented Programming using Java

Lec- 27

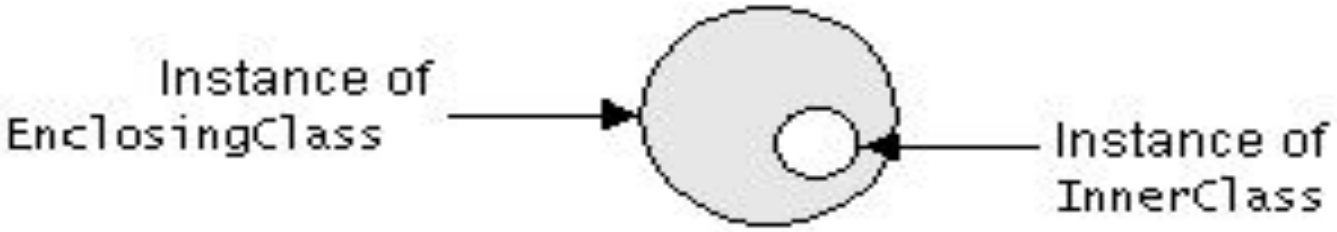
In this Discussion . . .

- Swing
  - Inner Classes
  - Model-view Controller (MVC) Architecture
  - Delegation Event Model
  - Inheritance Vs. Containment Hierarchy
  - Threads and Swings

Inner Classes (Revisited)

- **Nested class:** A class defined inside of another class.
- **Usage**:
  - Inner classes are hidden from other classes (encapsulated).
  - Inner objects can access/modify the fields of their outer object.

Event listeners are often defined as nested classes inside a GUI.



Model-View Controller (MVC) Architecture

Swing uses the model-view-controller architecture (MVC) as the fundamental design behind each of its components.

- Essentially, MVC breaks GUI components into three elements. Each of these elements plays a crucial role in how the component behaves.
- The Model-View-Controller is a well known software architectural pattern ideal to implement user interfaces on computers by dividing an application intro **three interconnected** parts

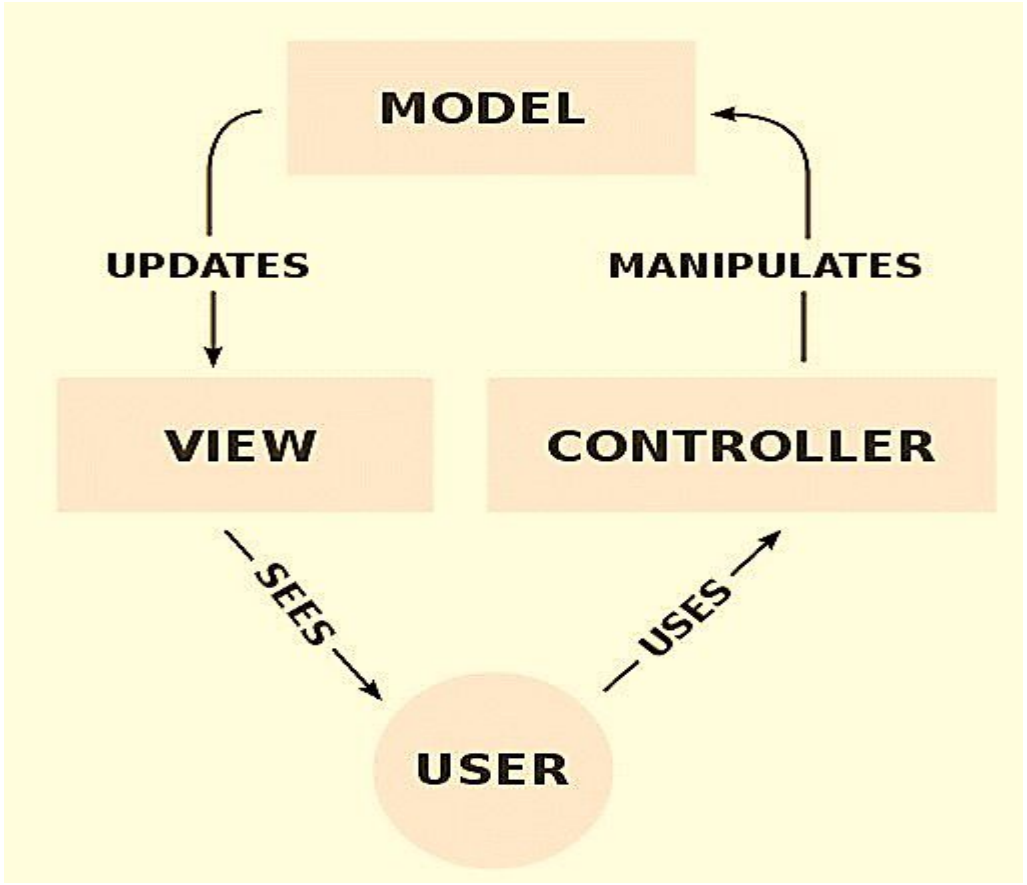
Model-View Controller (MVC) Architecture (Contd.)

Main goal of MVC, is to separate internal representations of an application from the ways information are presented to the user.

- Initially, MVC was designed for desktop GUI applications but it's quickly become an extremely popular pattern for designing web applications too.
- MVC pattern has the **three components**:
  - **Model** that manages data, logic and rules of the application
  - **View** that is used to present data to user
  - **Controller** that accepts input from the user and converts it to commands for the Model or View.

Model-View Controller (MVC) Architecture (Contd.)

- The MVC pattern defines the interactions between these three components like you can see in the following figure :



Model-View Controller (MVC) Architecture (Contd.)

- The Model receives commands and data from the Controller. It stores these data and updates the View.
- The View lets to present data provided by the Model to the user
- The Controller accepts inputs from the user and converts it to commands for the Model or the View.

Event Handling in Swing: Delegation Event Model

- Java foundation introduced "**Delegation Event Model**" i.e. describes how to generate and control the events.
- The **key** elements of the **Delegation Event Model** are **source** and **listeners**.
- The listener should have registered on source for the purpose of alert notifications.
- All GUI applications are event-driven

Swing is not thread-safe

Most Swing object methods are not thread safe , invoking them from multiple threads risks thread interference or memory consistency errors

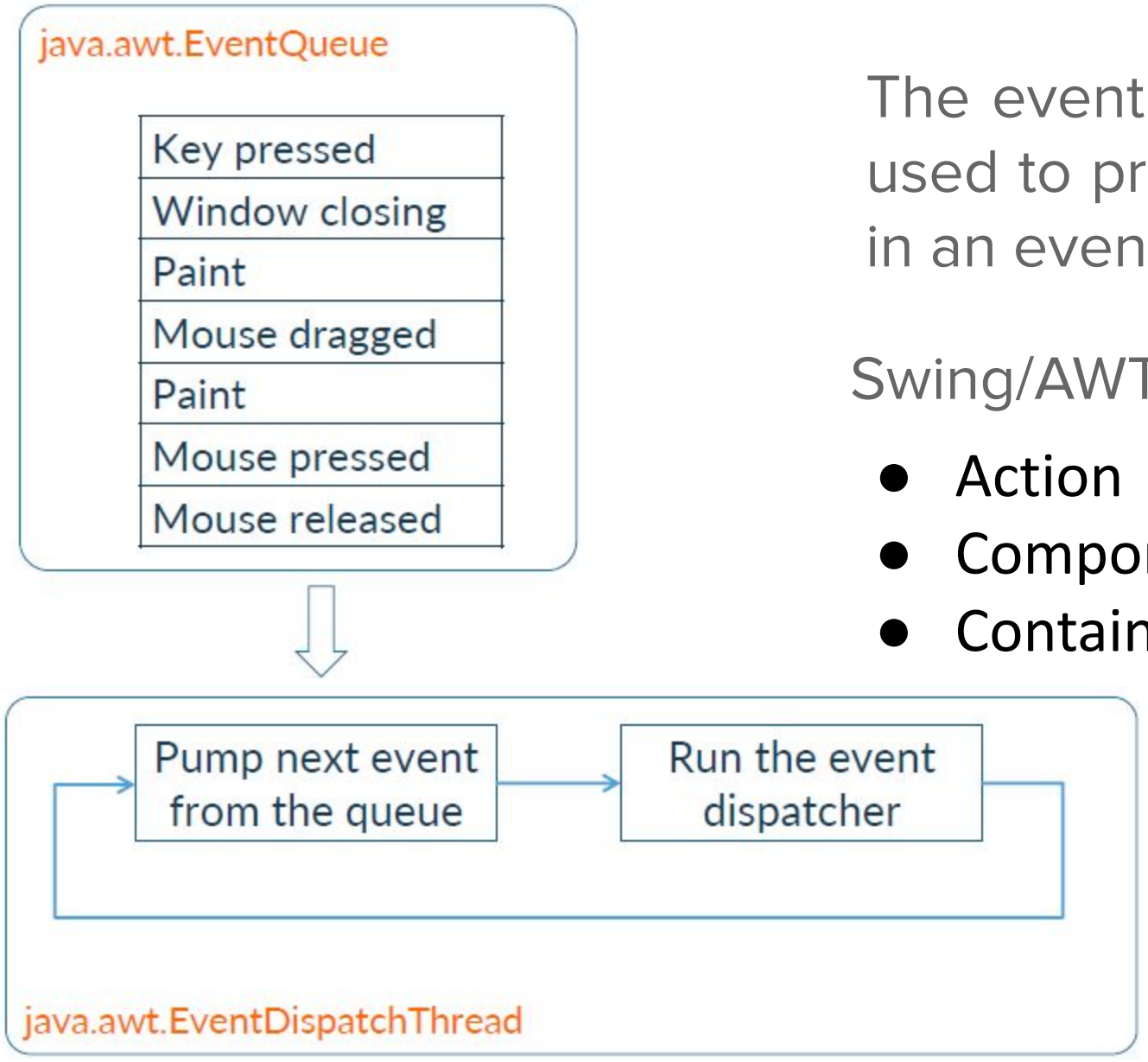
Swing event handling code runs on a special thread known as the event dispatch thread (EDT) and most of the code that invokes Swing methods also runs on this thread

Swing is not thread-safe

Some Swing component methods are labelled thread safe in the API specification; these can be safely invoked from any thread. All other Swing component methods must be invoked from the event dispatch thread.

Programs that ignore this rule may seems to run correctly most of the times but are subject to unpredictable errors that are difficult to reproduce.

The event queue & event dispatch thread



The event dispatch thread is a thread used to process the events enqueued in an event queue

Swing/AWT has several types of events:

- Action
- Component
- Container
- Mouse
- Key
- Window

- Focus
- Text
- MouseWheel
- etc.

Using the event dispatch thread

The code that handles Swing events is invoked from the event dispatch thread

If you need to determine whether your code is running on the event dispatch thread, invoke javax.swing.SwingUtilities.isEventDispatchThread

Using the event dispatch thread

Tasks on the event dispatch thread must finish quickly; if they don't, unhandled events back up and the user interface becomes unresponsive

Takeaways:

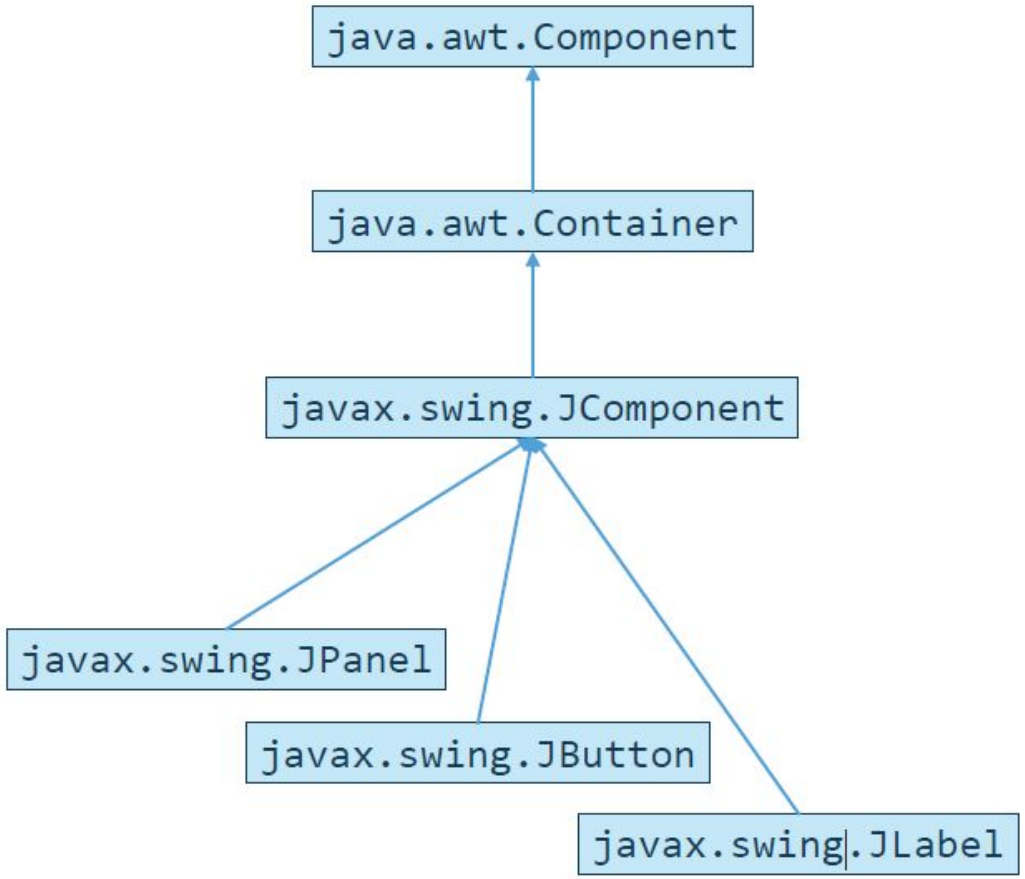
- Thread unsafe methods must be invoked from the event dispatch thread

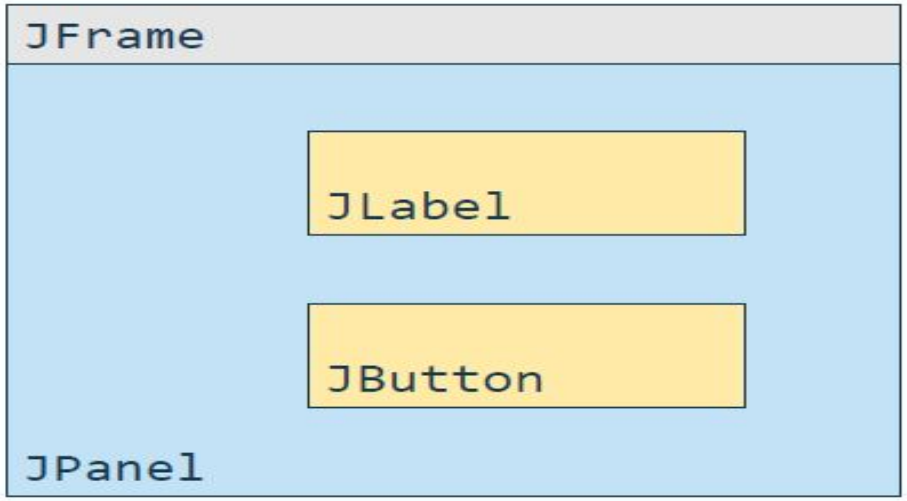
Takeaways:

- Swing is not thread safe
- Swing documentation indicates what methods are thread safe

Longer tasks should run in background, i.e., without blocking the GUI by using a SwingWorker

Inheritance Vs. Containment Hierarchy





Label and button are child components of a panel

Don't get confused!