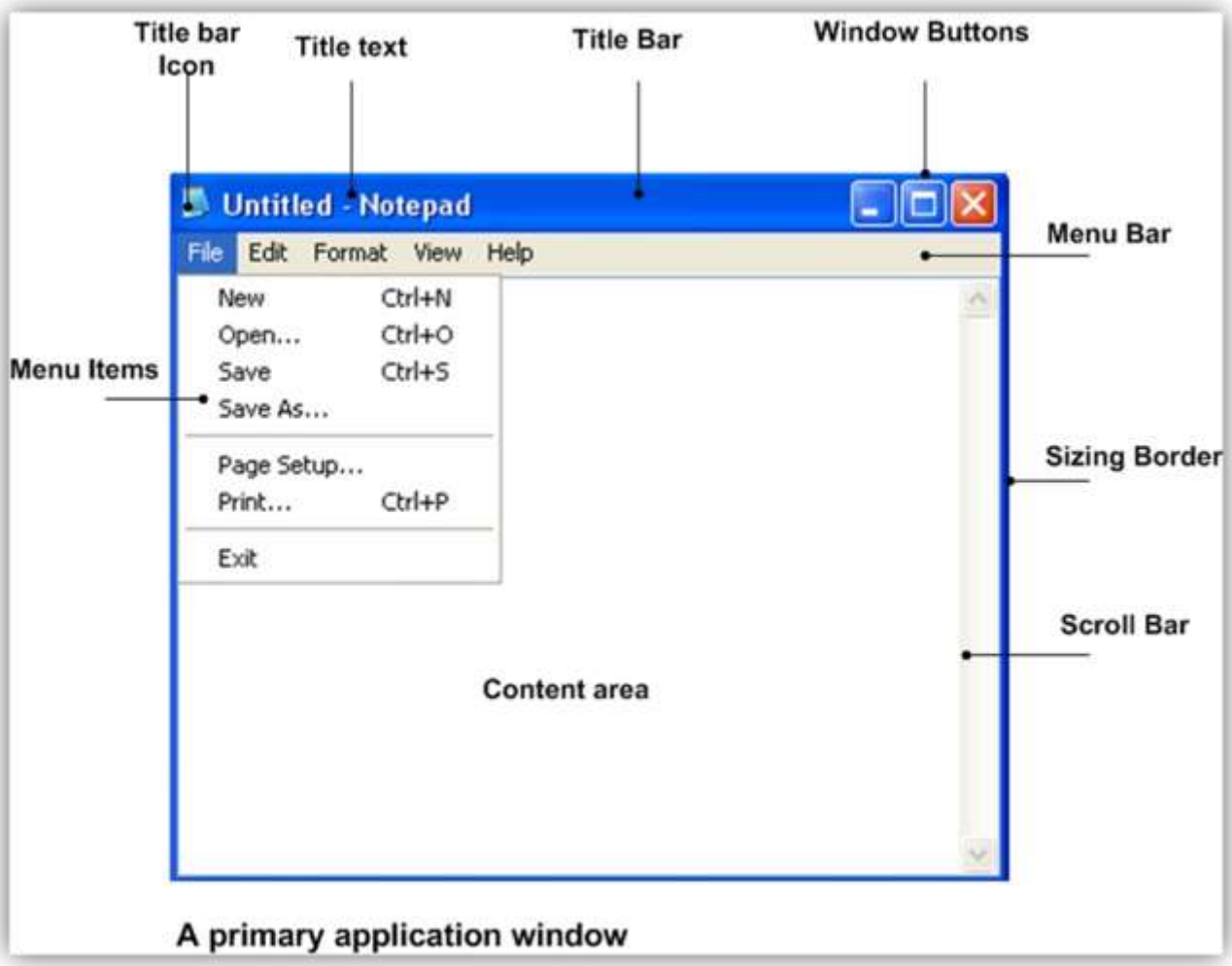
# CST8221 – Java Application Programming

# Unit 1

## Graphical User Interface (GUI) Fundamentals

**Graphical User Interface is a Human-Machine Interface used for interaction between humans and computers. The GUI consists of two basic parts: input devices and an application window. The basic computer input devices are keyboard and mouse. The application window is an image (that explains the term graphical) displayed on a computer screen. Using the input devices the user manipulates different parts of the application window. This interaction is made possible by the operating system (platform). Usually the operating system provides the main building components of the application window.**

**In order to create a GUI, programmers use some kind of API provided by the language library they use for writing their applications. If they work directly with the underlying windowing system provided by the operating system, this kind of programming is called native window programming.**



# Basic Principles of Good Interaction Design

- **Try to create a** *clear mental model* **of what the users are interacting with. In other words the users should understand what will happen when they take some action.**

- **A well designed system should have a** *reassuring feedback***, so that the users know what they have done when they have done it. When a user clicks on a button the button must change somewhat its appearance to indicate that it has been pressed.**

- *Navigability* **is also very essential, particularly with screen interfaces. The users need to know where they are in the system, what they can do there, where they can go next, and how to get back.**

- *Consistency* **is equally crucial. A certain command in one part of the system should have the same effect in another part. The *Copy* command in the Edit menu should have the same effect as the** *Copy* **command in the right-click pop-up menu.**

- *Intuitive interaction* **minimizes the burden of conscious thought needed to operate the system. Like a car, we do not spend too much time thinking about how the car operates: we think about where we are heading and what we want to do.**

- **When you design a computer-based system you are designing not just what it looks but how it** *behaves***. You are designing the quality of how the user and the system interact. When you move your mouse, for instance, does it feel sluggish, or nippy and sprightly? When you manipulate your smart phone or tablet does the combination of feel and sound, as well as informing you what you are doing is subtle and satisfying?**

- **The quality of the interaction should be appropriate to the context. An adventure game needs an interaction different from the one offered by the word processor.**

**The interactions can be catigorized acording to its "dimensions": 1-D, 2-D, 3-D, and 4-D. 1-D includes words. 2-D includes paining and icons. 3-D includes tree dimensional components. 4-D includes sound and animation. Think of the Recycle bin – you see and hear how you file disapear.**

# Guidelines for Interface Design Process

**Here are some guidelines for the user interface (GUI) design process outlined in the book "The Essential Guide to User Interface Design 2ed" by Wilbert O. Galitz, Wiley Computer Publishing.**
**The 14th development steps typically followed in creating a graphical system's or Web site's screens and pages are:**

**Step 1: Know Your User or Client. To begin, an understanding of the most important system or Web site component, the user or client, must be obtained. Understanding people and what they do is a critical and often difficult and undervalued process. The first step in the design process involves identifying people's innate and learned characteristics, and understanding how they affect design.**

**Step 2: Understand the Business Function. A system or Web site must achieve the business objectives for which it is designed. To do so requires an understanding of the goals of the system and the functions and tasks performed. Determining basic business functions, describing user activities through task analysis, understanding the user's mental model, and developing a conceptual model of the system accomplish this. The system's conceptual model must fit the user's view of the tasks to be performed. Step 2 also addresses the establishment of design standards or style guides, and the definition of training and documentation needs.**

**Step 3: Understand the Principles of Good Screen Design. A well-designed screen must reflect the needs and capabilities of its users, be developed within the physical constraints imposed by the hardware on which it is displayed, and effectively utilize the capabilities of its controlling software. Step 3 involves understanding the capabilities of, and limitations imposed by, people, hardware, and software in designing screens and Web pages. It presents an enormous number of general design principles for organizing and presenting information to people.**

**Step 4: Develop System Menus and Navigation Schemes. Graphical systems and Web sites are heavily menu-oriented. Menus are used to designate commands, properties that apply to an object, documents, and windows. To accomplish these goals, a variety of menu styles are available to choose from. Step 4 involves understanding how menus are used, and selecting the proper kinds for specific tasks. The principles of menu design are described, and the purpose and proper usage of various menu types are detailed. In this step Web site navigation schemes are also discussed.**

**Step 5: Select the Proper Kinds of Windows. Graphical screen design will consist of a series of windows. Step 5 involves understanding how windows are used and selecting the proper kinds for the tasks. The elements of windows are described, and the purpose and proper usage of various types of windows are detailed.**

**Step 6: Select the Proper Device-Based Controls. In addition to the keyboard, a system or Web site might offer the user a mouse, trackball, joystick, graphic tablet, touch screen, light pen, or some other similar device. Step 6 consists of identifying the characteristics and capabilities of these various control mechanisms and providing the proper ones for users and their tasks.**

**Step 7: Choose the Proper Screen-Based Controls. The designer is presented an array of screen-based controls to choose from. Selecting the right one for the user and the task is often difficult. But, as with device-based controls, making the right choice is critical to system success. A proper fit between user and control will lead to fast, accurate performance. A poor fit will result in lower productivity, more errors, and often user dissatisfaction. Step 7 consists of identifying the characteris tics and capabilities of these various screen-based controls and guidelines for providing the proper ones for users and their tasks.**

**Step 8: Write Clear Text and Messages. Creating text and messages in a form the user wants and understands is absolutely necessary for system acceptance and success. Rules for writing text and messages for systems and Web sites are presented.**

**Step 9: Provide Effective Feedback and Guidance and Assistance. Effective feedback and guidance and assistance are also necessary elements of good design. This step presents the guidelines for presenting to the user feedback concerning the system and its processing status. It also describes the system response times necessary to meet user needs. Step 9 also describes the kinds of guidance and assistance that should be included in a system, and presents important design guidelines for the various kinds.**

**Step 10: Provide Effective Internationalization and Accessibility. People from different cultures, and people who speak different languages may use graphical systems and Web sites. Guidelines for accommodating different cultures and languages in a design are presented. People with disabilities may also be users. Design considerations for these kinds of users are also described.**

**Step 11: Create Meaningful Graphics, Icons, and Images. Graphics, including icons and images, are an integral part of design. Design guidelines for various types of graphics are presented. Icons are also described, including a discussion of what kinds of icons exist, what influences their usability, and how they should be designed so they are meaningful and recognizable.**

**Step 12: Choose the Proper Colors. Color, if used properly, can emphasize the logical organization of a screen, facilitate the discrimination of screen components, accentuate differences, and make displays more interesting. If used improperly, color can be distracting and cause visual fatigue, impairing a system's usability. Step 12 involves understanding color and how to use it effectively on textual and statistical graphics screens, and in Web sites.**

**Step 13: Organize and Layout Windows and Pages. After determining all the components of a screen or page, the screen or page must be organized and its elements presented clearly and meaningfully. Proper presentation and organization will encourage the quick and accurate comprehension of information and the fastest possible execution of user tasks. Step 13 addresses the rules for laying out all screen elements and controls in the most effective manner possible.**

**Step 14: Test, Test, and Retest. A host of factors must be considered in design and numerous trade-offs will have been made. Indeed, the design of some parts of the system may be based on skimpy data and simply reflect the most educated guess possible. Also, the implications for some design decisions may not be fully appreciated until the results can be seen. Waiting until after a system has been implemented to uncover any deficiencies and make any design changes can be aggravating, costly, and time-consuming. To minimize these kinds of problems, interfaces and screens must be continually tested and refined as development proceeds. Step 14 reviews the kinds of tests that can be performed, and discusses creating, evaluating, and modifying prototypes in an iterative manner. It also reviews final system testing and ongoing evaluations of working systems.**

**The actual design process does not always fall into such neat categories — one step finishing and only then the next step starting. In reality, some steps will run concurrently or overlap, and design iterations will cause occasional movements backward as well as forward. If any of these steps are omitted, or carelessly performed, a product's foundation will be flawed. A flawed foundation is difficult to correct afterwards.**

# Guidelines for Visually Pleasing Composition

- **Balance - equal weight of screen elements, left and right, top and bottom.**
- **Symmetry - replicate elements left and right of the screen centerline.**
- **Regularity - similar element sizes, shapes, colors, and spacing.**
- **Predictability - be consistent and follow conventional orders or arrangements.**
- **Sequentiality – arrange elements to guide the eye through the screen.**
- **Economy - use as few styles, display techniques, and colors as possible.**
- **Unity - use similar sizes, shapes, or colors for related information.**
- **Proportion - data or text with aesthetically pleasing proportions.**
- **Simplicity - optimize the number of elements on a screen, within limits of clarity.**
- **Groupings - provide functional groupings of associated elements.**

**Eyeball fixation studies also indicate that during the initial scanning of a display in a clockwise direction, people are influenced by the symmetrical balance and weight of the titles, graphics, and text of the display. The human perceptual mechanism seeks order and meaning, trying to impose structure when confronted with uncertainty. Whether a screen has meaningful and evident form or is cluttered and unclear is immediately discerned. A cluttered or unclear screen requires that some effort be expended in learning and understanding what is presented. The screen user who must deal with the display is forced to spend time to learn and understand. The user who has an option concerning whether the screen will or will not be used may reject it at this point if the perceived effort in understanding the screen is greater than the perceived gain in using it.**

# Introduction to Swing

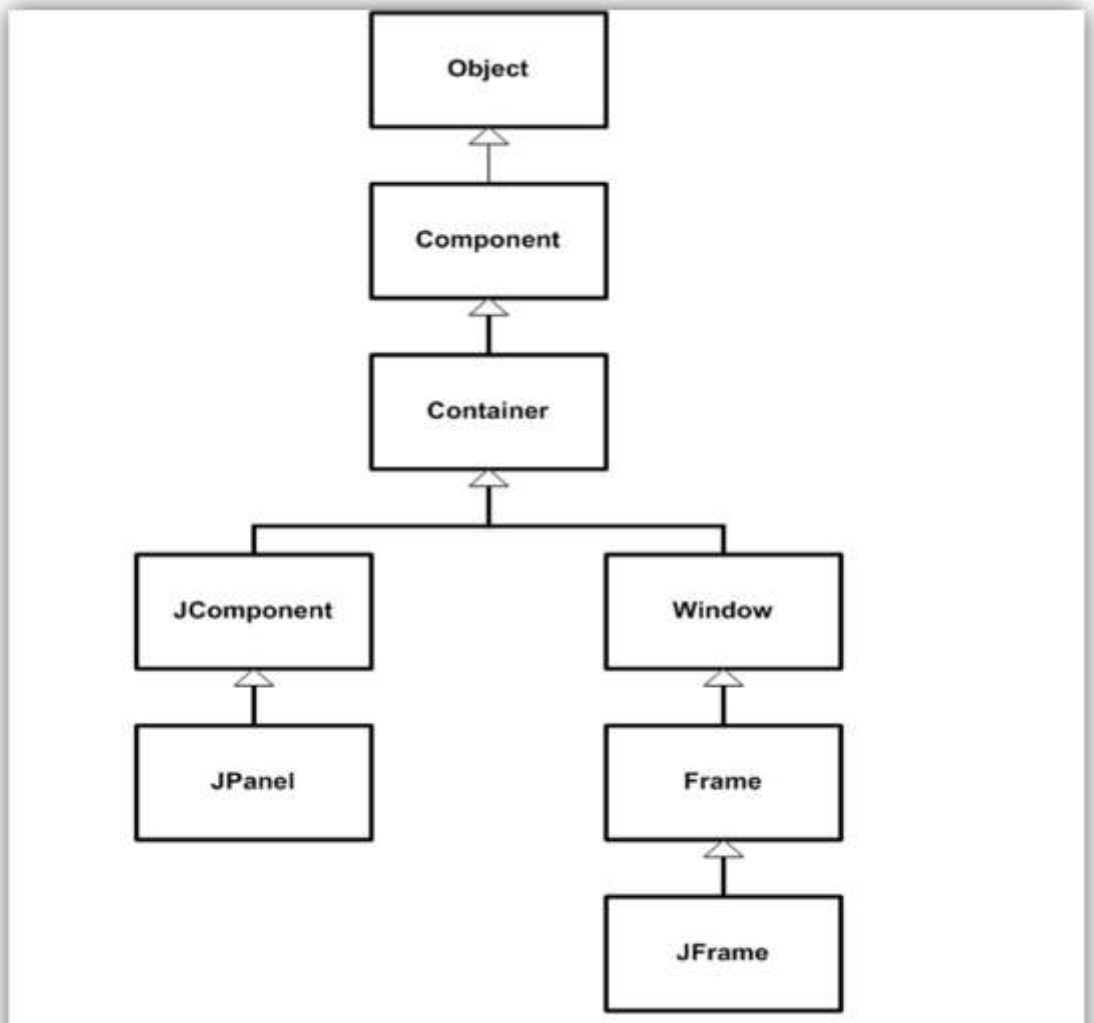
**When Java was introduced for the first time, it contained a class library called Abstract Windowing Toolkit (AWT) which provided the programmer with the basic components for GUI programming. The basic AWT library deals with user interface elements by delegating their creation and behavior to the native windowing system (Windows, Solaris, Linux, Mac, and so on). Each component in AWT has a "native" peer responsible for handling the user actions. For that reason are called "heavyweight components." The peer based approach is very effective but has two major problems. First, it depends on the platform, which means that the GUI will look and behave differently on each platform, and second, if the platform changes or adds some components new peers must be provided in the library. The solution to this problem is to create a GUI library, which is not heavily dependent on the underlying platform. Sun created such library for Java 1.2 under the code name Swing. Swing is now part of so called Java Foundation Classes (JFC). The Swing components such as buttons, menus and so on are painted onto blank windows. For that reason they are called "lightweight components." The only functionality required from the underlying windowing system is to put up a window and to pain on the window. Swing is not a complete replacement for the AWT – it is build on the top of AWT architecture and contains some heavyweight components. It uses the foundation of AWT for event handling. Of course, Swing based user interface will be always somewhat slower to appear on the user's screen than the peer-based component used by AWT, but this is not a major concern for application running on a modern computers. On the other hand the advantages of Swing are numerous: Swing provides a very reach set of user interface elements; Swing has a very few dependencies on the underlying platform, which means that it is less prone platform specific bugs; Swing provides consistent user experience across platforms and allows for adjustment to the native platform "Look and Feel."**

# Components and Containers

**To build a GUI the programmer needs to perform a few simple tasks:**

- **Choose a primary container (window).**
- **Choose GUI components and containers which will be part of the interface. Arrange the components inside the containers.**
- **Arrange the components and the containers inside the primary container.**
- **Handle the events generated by the user actions such as mouse click or a keystroke.**

**A primary container or a top-level window (that is, a window that is not contained inside another window) is called *frame* in Java. The AWT library has a class, called *Frame* for this primary container. The Swing version of this class is called *JFrame* and it extends the *Frame* class. The JFrame is one of the few Swing components which is not painted. Thus all its decorations are drawn by the user's windowing system.**

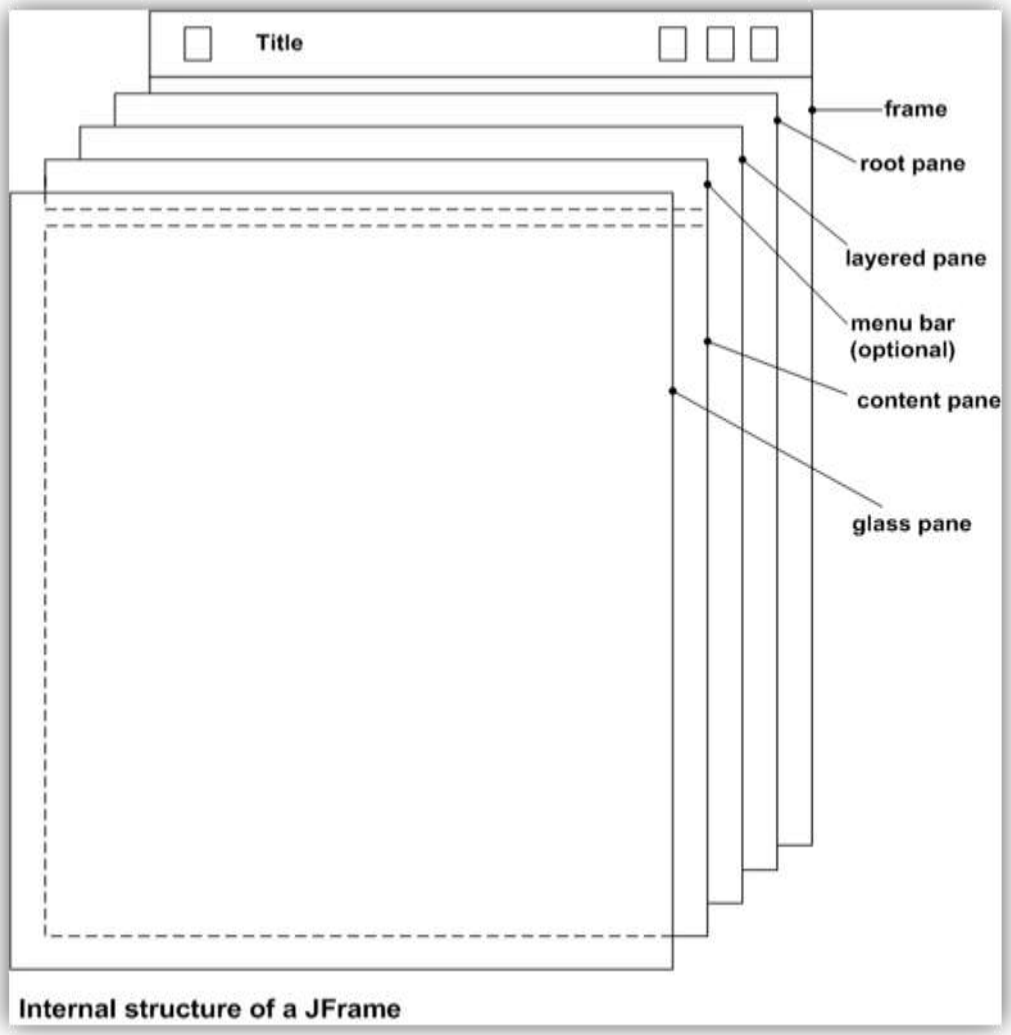


Inheritance hierarchy for frame and component classes in AWT and Swing

**This is how a simple frame looks like.**


A simple frame

**Internally the frame has a very complex structure. It contains a root pane which has four parts.**



Internal structure of a JFrame

## Root pane

**The root pane is created automatically when you instantiate JInternalFrame or one of the top-level Swing containers, such as JFrame , JApplet, JDialog, JWindow, and JInternalFrame is instantiated**.

## Glass pane

**Hidden, by default. If you make the glass pane visible, then it's like a sheet of glass over all the other parts of the root pane. It's completely transparent unless you implement the glass pane's paintComponent method so that it does something, and it can intercept input events for the root pane. The glass pane is useful when you want to be able to catch events or paint over an area that already contains one or more components. For example, you can deactivate mouse events for a multi-component region by having the glass pane intercept the events. Or you can display an image over multiple components using the glass pane.**

## Layered pane

**A layered pane is a container with depth such that overlapping components can appear one on top of the other. It serves to position its contents, which consist of the content pane and the optional menu bar. The layers of the layered pane are: Default, Palette, Modal, Popup, and Drag.**

## Content pane

**The container of the root pane's visible components, excluding the menu bar.**

## Optional menu bar

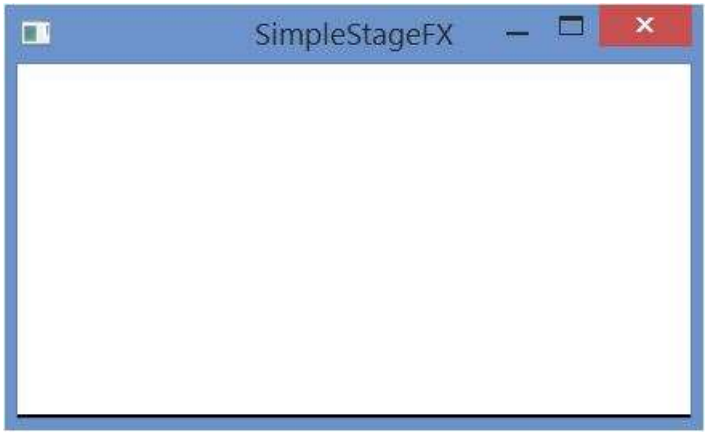
**The home for the root pane's container's menus. If the container has a menu bar, you generally use the container's setJMenuBar method to put the menu bar in the appropriate place.**

# Introduction to Java FX

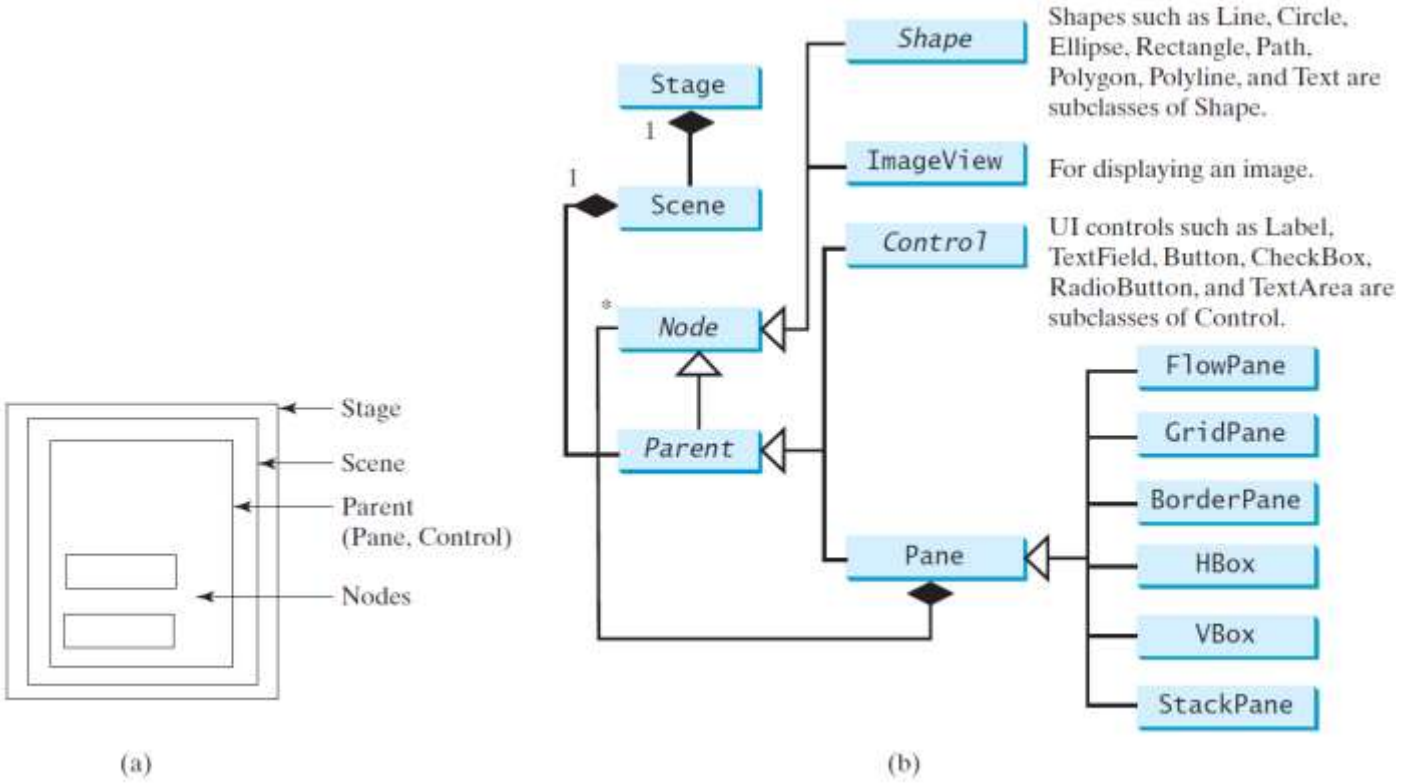
**Swing has been designed to for developing desktop GUI applications. It is now gradually replaced by completely new GUI framework known as JavaFX. JavaFX is fully integrated in Java 8. JavaFX incorporate modern GUI technologies and has a built-in support for modern GPUs. It enables the development of rich Internet applications. A rich Internet application (RIA) is a web application designed to deliver the same functions normally associated with desktop applications. A JavaFX application can run on a desktop and/or within a web browser. Additionally, Java FX provides a multi-touch support for touch-screen devices. JavaFX has a built-in 2D, 3D, animation support, video and audio playback.**

**JavaFX and Swing share many similarities but JavaFX introduces completely new architecture, terminology and API. In Swing the main container is called Frame; in JavaFX the main container is called Stage. JavaFX can embed Swing GUIs.**

**The figure below shows a simple empty stage**.




**The JavaFX API provides many different components for building a GUI. The relationship among these different component is shown in the figure below**.




**The link below gives details of the JavaFX architecture.**

http://docs.oracle.com/javafx/2/architecture/jfxpub-architecture.htm