1. What Are the Java Foundation Classes (JFC)?

Java Foundation Classes are a set of Graphical User Interface(GUI) support packages, including:

- Abstract Window Toolkit (AWT)
- The Swing component set
- 2D graphics, using the Java
- 2D API to perform advanced drawing.
- Pluggable look-and-feel, this feature provides Swing components a choice of look-and-feel and the same program can be rendered in Microsoft Windows and other platforms.
- Accessibility, these features used programs to be accessible to those with disabilities.
- Drag-and-drop, it includes cut-and-paste and drag-and-drop.
- Internationalization, it supports different characters sets such as Japanese, Chinese and Korean.

1.1.Graphical User Interface (GUI) using Swing API

Swing is an enhanced component set that provides replacement components for those in the original AWT and a number of more advanced components.
These components enable user to create user interface with the type of functionality that has become expected in modern application, such as **trees, tables, advanced text editors, and tear-off toolbars.**

Swing also has special features. For example, using Swing, you can write a program that adopts either the ***look-and-feel*** of the host platform or that use a common look-and-feel written especially for the Java programming language.

1.2.Pluggable feel-and-feel

Pluggable feel-and-look enables developers to build application that execute on any platform as if they were developed for that specific platform. In other words, program executed in the Microsoft Windows environment appears as if it was developed for this environment; and the same program executed on the UNIX platform appears as if it was developed for the UNIX environment.

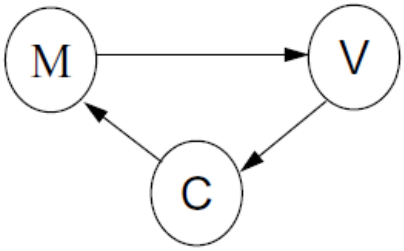
1.3.Swing Architecture

Swing components are designed based in the **Model-View-Controller** (MVC) architecture, The Swing component architecture is not strictly based on the MVC controller but its root in the MVC.

Model-View-Controller architecture

According to the MVC architecture , a component can be modeled as there separate parts. Figure in below shows that:

- Model: the model stores the data used to define the component.
- View: the view represents the visual display of the components.
- Controller- the controller deals with the behavior of the components when a user interacts with it



Model-View-Controller Architecture

Swing Packages API has a rich and convenient set of packages that makes it powerful and flexible . Table (1) list each package name and the purpose of each package

Package Name	Purpose
javax.swing	Provides a set of *light-weight* components such as, JButton, JFrame, JCheckBox and much more
javax.swing.border	Provides classes and interfaces for drawing specialized borders such as , bevel, etched, line , matte and more
javax.swing.event	support for events fired by Swing componants
javax.swing.undo	Allows developers to provide support for undo/redo in applications such as a text editors
javax.swing.colorchooser	Contains classes and interfaces used by JColorChooser components
javax.swing.filechooser	Contains classes and intefaces used by the JFilechooser component
javax.swing.table	Provides classes and interfaces for handling JTable
javax.swing.tree	Provides classes and interfaces for handling JTree
javax.swing.plaf	Provides one interface and many abstract classes that Swing uses to provide its pluggable look-and-feel capabilities
java.swing.plaf.basic	Provide user interface objects built according to the basic look-and-feel
javax.swing.plaf.metal	Provides user interface objects built according to the |Java look-and-feel
javax.swing.plaf.multi	Provides user interface objects that combine two or more look-and-feels
javax.swing.plaf.synth	Provides user interface objects for a skinnable look-and-feel in which all painting us delegated
javax.swing.text	Provides classes and interfaces that deal with editable and non-editable text components
javax.swing.text.html	Provides the class HTMLEditorKit and supporting classes for creating HTML editors

Examining the Composition of a Java Technology GUI

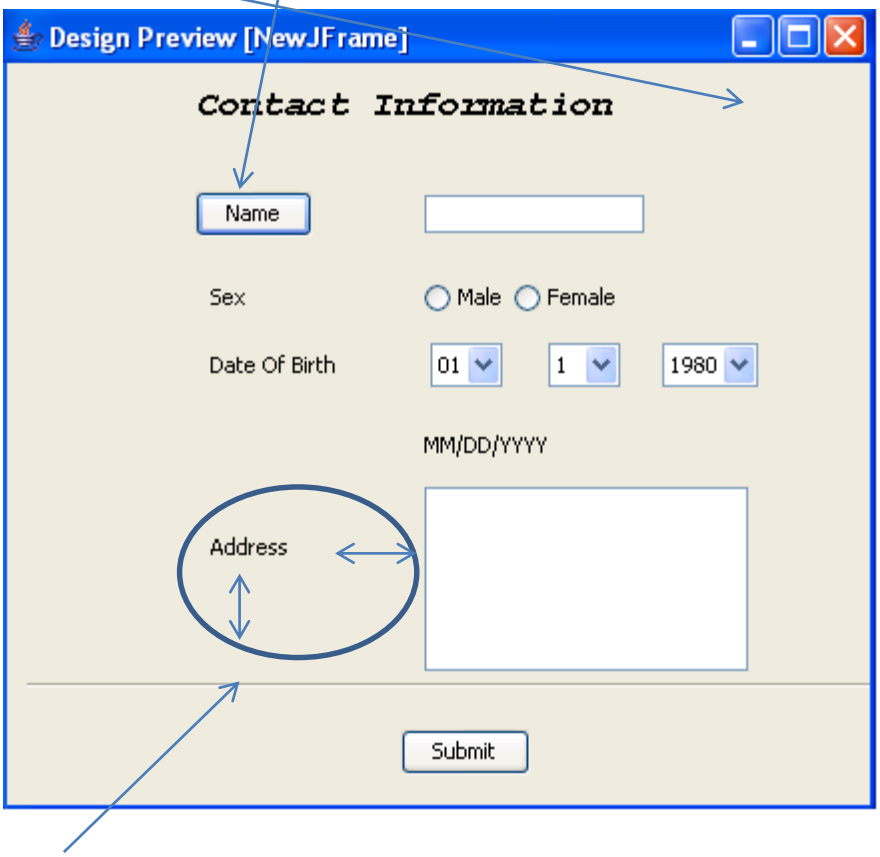
A Swing API-based GUI is composed of the following elements:

- Containers – Are on top of the GUI containment hierarchy.
- Components – Contain all the GUI components that are derived from the JComponent class.
- Layout Managers – Are responsible for laying out components in a container.

Example:

Name button is **component**

Container



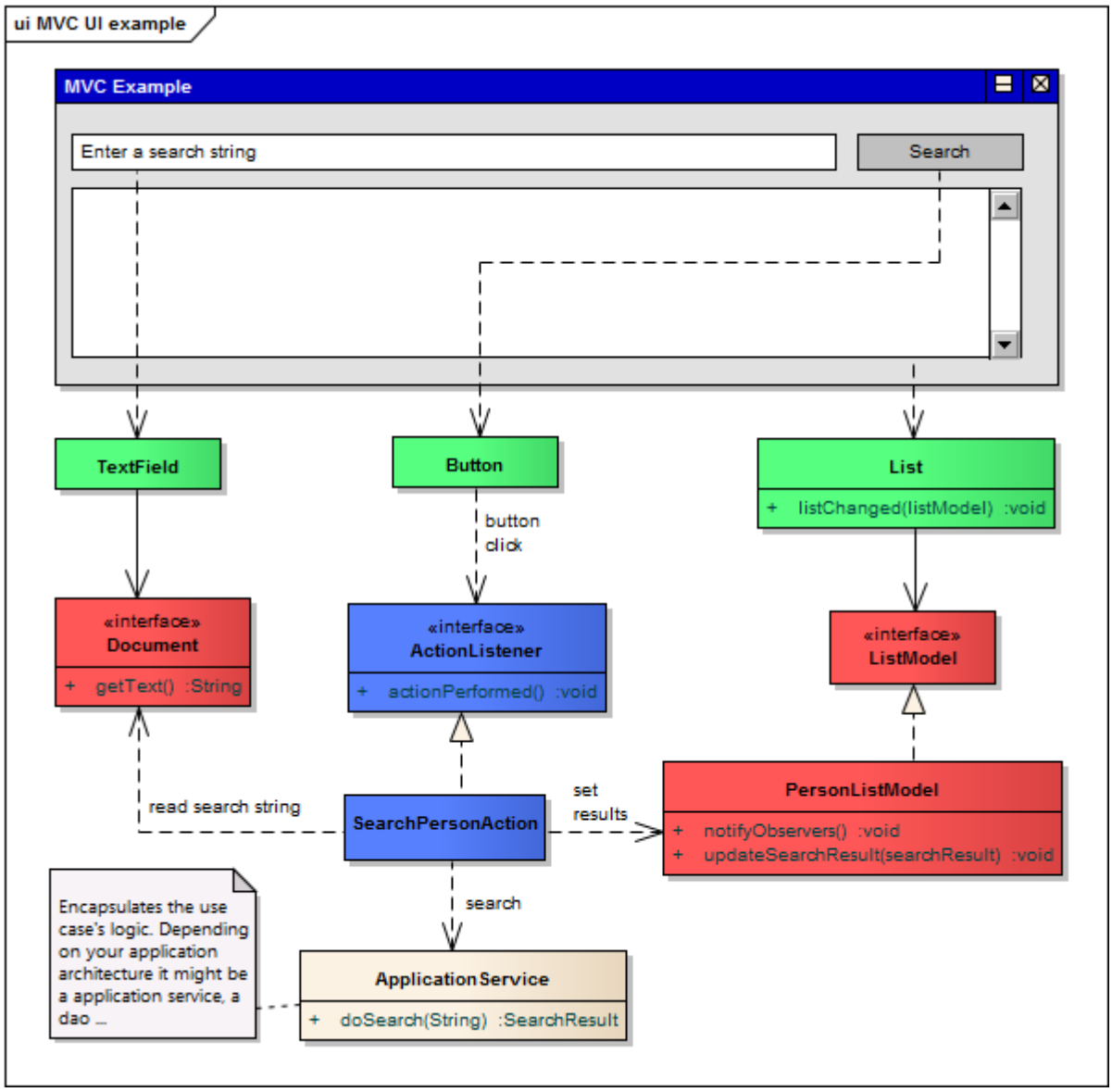
Layout manager

Example 2:

Suppose an application that let you search for persons. The UI must have a text field where the user can enter a search string and it might have a button to start the search. Finally it must have an area where the search results are displayed. In our case this are is implemented as a list component.

The normal use case is:

- The user enters a search string in the text field
- The user clicks the search button.
- The search result is displayed in the result list.


ui MVC UI example
MVC Example
Enter a search string
Search
TextField
Button
List
+ listChanged(listModel) :void
button
click
«interface»
Document
+ getText() :String
«interface»
ActionListener
+ actionPerformed() :void
«interface»
ListModel
read search string
SearchPersonAction
set
results
PersonListModel
+ notifyObservers() :void
+ updateSearchResult(searchResult) :void
search
Encapsulates the use case's logic. Depending on your application architecture it might be a application service, a dao ...
ApplicationService
+ doSearch(String) :SearchResult