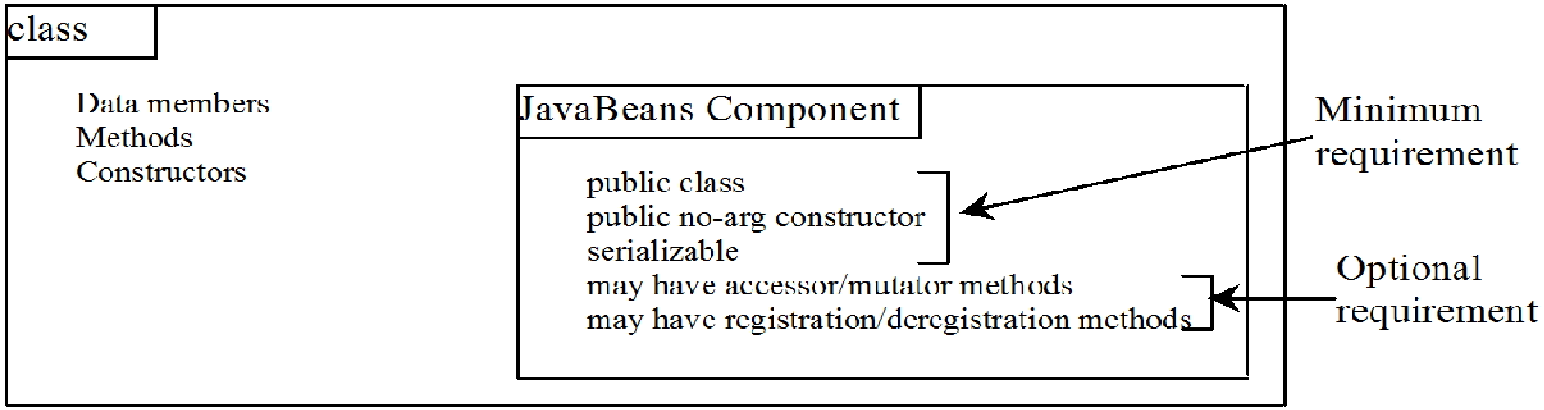
Course Name:
Advanced Java

Lecture 22
Topics to be covered

- JAVABEANS COMPONENTS
- Beans
- the Bean-Writing Process

What is a Java Bean?

- "A Java Bean is a reusable software component that can be manipulated visually in a builder tool"



JavaBeans Objectives

- Provide a platform neutral component architecture
- Simple to develop
- Leverage existing Java technologies
- Provide design-time support for visual builder tools

JavaBeans Characteristics

- Properties
- Event
- Persistence
- Introspection
- Customization

Component Framework

- Component
  - A reusable software module that is supplied in a binary form and can be used to compose applications
- Container
  - Provides the context that components can be assembled and interact with one another
- Component Model
  - Defines the architecture of how components can interact in a dynamic environment

The Component Model

- JavaBeans is Java's component model
- The model is made up of an architecture and an API
- The API makes it possible to write component software in Java
- The architecture provides the framework (services and rules) that allows components to participate properly

The Component Model

- Discovery and Registration
- Raising and Handling of Events
- Persistence
- Visual Presentation
- Support for Visual Programming

The Component Model

- Discovery and Registration
  - Locate a component at run-time and determine its supported interfaces
  - Registration process for a component to make itself and its interfaces known

This mechanism allows components and applications to be developed independently

The Component Model

- Raising and Handling of Events
  - Beans (or JavaBeans components) use events to communicate with other Beans
  - A Bean that wants to receive events (a *listener* Bean) registers its interest with the Bean that triggers the event (a *source* Bean)

The Component Model

- Persistence
  - Persistence enables Beans to save and restore their state
  - JavaBeans uses Java Object Serialization to support persistence

The Component Model

- Visual Presentation
  - The Bean is free to choose its own visual presentation (fonts, colours, shape, etc)
  - Many of these characteristics will be properties of the Bean (some might be persistent too)

The Component Model

- Support of Visual Programming
  - User can select a component from the toolbox and place it into a container
  - Properties of the component can then be edited to create the desired behaviour

Bean Methods

- A Bean may be implemented by a Java Class
- That Class contains a number of methods that may be used to access and control the Bean
- These are generally all the public methods of the Class that implements the Bean

The Bean-Writing Process

- The simplest kind of bean is really nothing more than a java class that follows Some strict naming conventions for its methods. All accessor methods begin With get & all mutator methods begin with set. Builder tools use this standard naming conventions for its methods. Properties are conceptually at a higher level than instance fields – they are features of the interface, whereas instance fields belong to the implementation of the class.

- Real - world Beans are much more elaborate & tedious to code, because of two reasons:

- Beans must be usable by less-than-expert programmers. U need to expose lots of properties so that your users can access most of the functionality of your bean with a design tool & without programming.
- The same bean must be usable in a wide variety of contexts. Both the behavior & appearance of your bean is customizable.
- A good example of a bean with rich behavior is CalendarBean by Kai Todter.
- This bean gives user a convenient way of entering dates, simply by locating them in a calendar display.
- By using a bean such as this one, u can take advantage of the work of others, simply by