Q6 – Q11 Swing interview questions and answers

Posted on March 7, 2015 by Arulkumaran Kumaraswamipillai

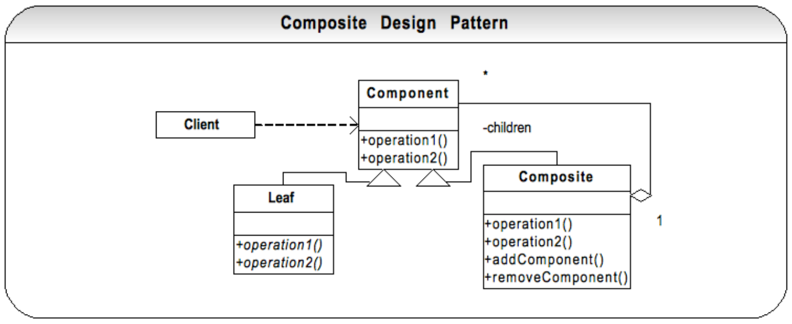
0 Like Share Tweet 0 G+1 Share

Extends Q1 to Q5 Swing interview questions and answers.

Q6. If you add a component to the CENTER of a border layout, which directions will the component stretch?
A6. The component will stretch both horizontally and vertically. It will occupy the whole space in the middle.

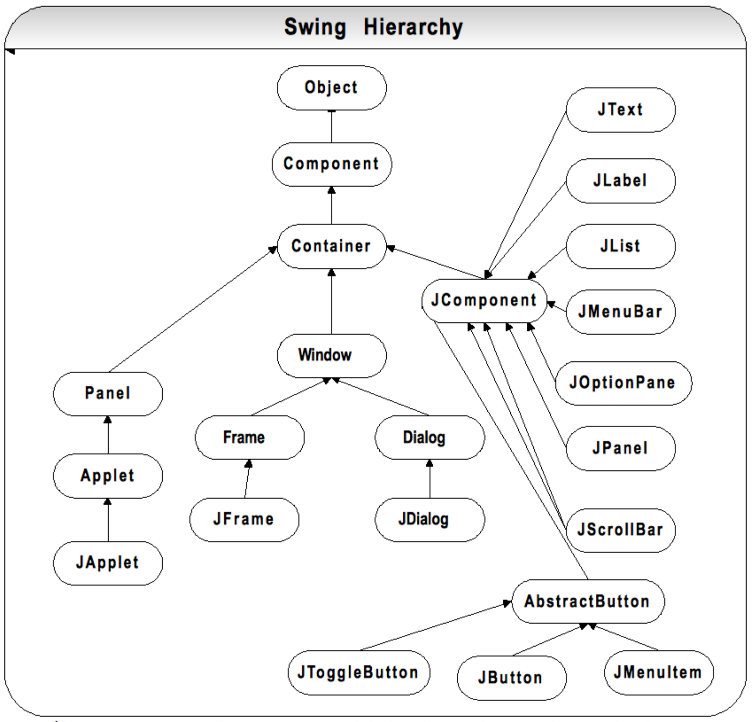
Q7. What is the base class for all Swing components? What design pattern does this follow?
A7. Design pattern: As you can see from the diagram below, containers collect components. Sometimes you want to add a container to another container. So, a container should be a component. For example container.getPreferredSize() invokes getPreferredSize() of all contained components.

Composite design pattern is used in GUI components to achieve this. A composite object is an object, which contains other objects. Composite design pattern manipulates composite objects just like you manipulate individual components.



Composite design pattern

All the Swing components start with ‘J’. The hierarchy diagram is shown below. **JComponent** is the base class.



(**Diagram source**: http://www.particle.kth.se/~fmi/kurs/PhysicsSimulation/Lectures/07A/swingDesign.html)

Swing components

16 Technical Key Areas

Q8. Explain the Swing event dispatcher mechanism?
A8. Swing components can be accessed by the Swing **event dispatching thread**. A few operations are guaranteed to be thread-safe but most are not. Generally the Swing components should be accessed through this event-dispatching thread. The event-dispatching thread is a thread that executes drawing of components and event-handling code. For example the paint() and actionPerformed() methods are automatically executed in the event-dispatching thread. Another way to execute code in the event-dispatching thread from outside event-handling or drawing code, is using SwingUtilities **invokeLater()** or **invokeAndWait()** method. **Swing lengthy initialization tasks (e.g. I/O bound and computationally expensive tasks), should not occur in the event-dispatching thread because this will hold up the dispatcher thread**. If you need to create a new thread for example, to handle a job that's computationally expensive or I/O bound then you can use the thread utility classes such as SwingWorker or Timer without locking up the event-dispatching thread.

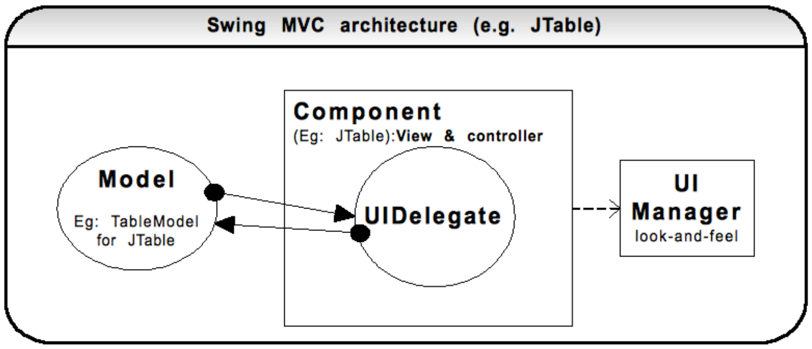
- **SwingWorker** – creates a background thread to execute time consuming operations.
- **Timer** – creates a thread that executes at certain intervals.

However after the lengthy initialization the GUI update should occur in the event dispatching thread, for thread safety reasons. We can use invokeLater() to execute the GUI update in the event-dispatching thread. The other scenario where invokeLater() will be useful is that the GUI must be updated as a result of non-AWT event.

Q9. What do you understand by MVC as used in a JTable?
A9. MVC stands for **M**odel **V**iew **C**ontroller architecture. Swing "J" components (e.g. JTable, JList, JTree etc) use a modified version of MVC. MVC separates a model (or data source) from a presentation and the logic that manages it.

80+ step by step Java Tutorials

100+ Java pre-interview coding tests


Swing MVC architecture (e.g. JTable)
Component
(Eg: JTable):View & controller
Model
Eg: TableModel for JTable
UIDelegate
UI Manager
look-and-feel

How good are your .....?

open all | close all

⊞ Career Making Know-

⊞ Job Hunting & Resum

- **Component (**e.g. JTable, JTree, and JList): coordinates actions of model and the UI delegate. Each generic component class handles its own individual **view-and-controller** responsibilities.
- **Model** (e.g. TableModel): charged with storing the data.
- **UIDelegate**: responsible for getting the data from model and rendering it to screen. It delegates any look-and-feel aspect of the component to the **UI Manager**.

Q10. Explain layout managers? What design pattern does it use?
A10. Layout managers are used for arranging GUI components in windows. The standard layout managers are:

- **FlowLayout**: Default layout for Applet and Panel. Lays out components from left to right, starting new rows if necessary.
- **BorderLayout**: Default layout for Frame and Dialog. Lays out components in north, south, east, west and center. All extra space is placed on the center.
- **CardLayout**: stack of same size components arranged inside each other. Only one is visible at any time. Used in TABs.
- **GridLayout**: Makes a bunch of components equal in size and displays them in the requested number of rows and columns.
- **GridBagLayout**: Most complicated but the most flexible. It aligns components by placing them within a grid of cells, allowing some components to span more than one cell. The rows in the grid aren't necessarily

all the same height, similarly, grid columns can have different widths as well.
- **BoxLayout**: is a full-featured version of FlowLayout. It stacks the components on top of each other or places them in a row.

Complex layouts can be simplified by using nested containers for example having panels within panels and each panel can use its own LayoutManager. It is also possible to write your own layout manager or use manual positioning of the GUI components. Note: Further reading on each LayoutManagers is recommended for Swing developers.

Design pattern: The AWT containers like panels, dialog boxes, windows etc do not perform the actual laying out of the components. They delegate the layout functionality to layout managers. The layout managers make use of the **strategy design pattern**, which encapsulates family of algorithms for laying out components in the containers. If a particular layout algorithm is required other than the default algorithm, an appropriate layout manager can be instantiated and plugged into the container. For example, panels by default use the FlowLayout but it can be changed by executing:

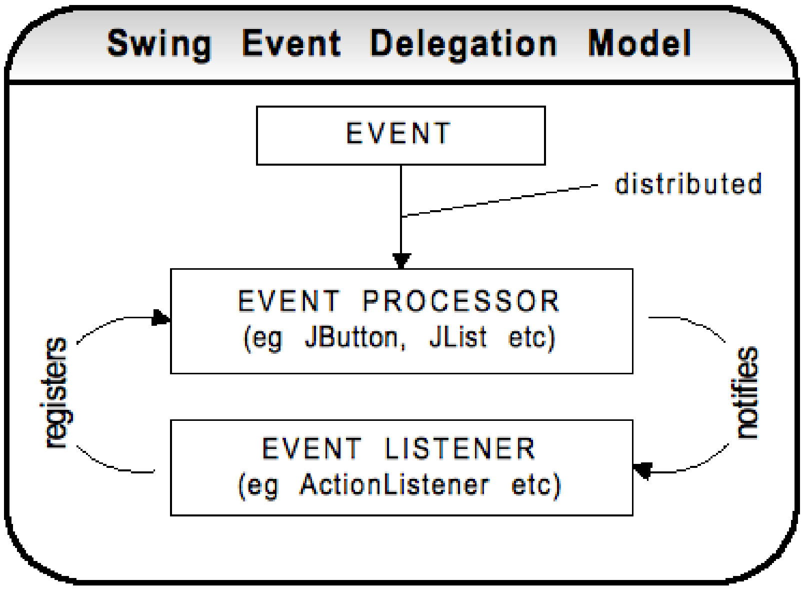
```
panel.setLayout(new GridLayout(4,5));
```

This enables the layout algorithms to vary independently from the containers that use them. This is one of the key benefits of the strategy pattern.

Q11. Explain the Swing delegation event model?
A11. In this model, the objects that receive user events notify the registered listeners of the user activity. In most cases the event receiver is a component.

- **Event Types**: ActionEvent, KeyEvent, MouseEvent, WindowEvent etc.
- **Event Processors**: JButton, JList etc.

- **EventListeners**: ActionListener, ComponentListener, KeyListener etc.


Swing Event Delegation Model
EVENT
distributed
EVENT PROCESSOR (eg JButton, JList etc)
EVENT LISTENER (eg ActionListener etc)
registers
notifies

Popular Posts

♦ 11 Spring boot interview questions & answers

825 views

♦ Q11-Q23: Top 50+ Core on Java OOP Interview Questions & Answers

766 views

18 Java scenarios based interview Questions and Answers

400 views

001A: ♦ 7+ Java integration styles & patterns interview questions & answers

388 views

01b: ♦ 13 Spring basics Q8 – Q13 interview questions & answers

295 views

♦ 7 Java debugging interview questions & answers

293 views

01: ♦ 15 Ice breaker questions asked 90% of the time in Java job interviews with hints

285 views

♦ 10 ERD (Entity-Relationship Diagrams) Interview Questions and Answers

279 views

♦ Q24-Q36: Top 50+ Core on Java classes, interfaces and generics interview questions & answers

239 views

001B: ♦ Java architecture & design concepts interview questions & answers

201 views

Bio | Latest Posts

Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site.**945+** paid members. join my LinkedIn Group. **Reviews**

About Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site.**945+** paid members. join my LinkedIn Group. **Reviews**

‹ 5 Swing & AWT interview questions and answers

Explain abstraction, encapsulation, Inheritance, and polymorphism with the given code? ›

Posted in Swing & AWT

Empowers you to open more doors, and fast-track

Technical Know Hows

☀ Java generics in no time ☀ Top 6 tips to transforming your thinking from OOP to FP ☀ How does a HashMap internally work? What is a hashing function?
☀ 10+ Java String class interview Q&As ☀ Java auto un/boxing benefits & caveats ☀ Top 11 slacknesses that can come back and bite you as an experienced Java developer or architect

Non-Technical Know Hows

☀ 6 Aspects that can motivate you to fast-track your career & go places ☀ Are you reinventing yourself as a Java developer? ☀ 8 tips to safeguard your Java career against offshoring ☀ My top 5 career mistakes

Prepare to succeed

☀ Turn readers of your Java CV go from "Blah blah" to "Wow"? ☀ How to prepare for Java job interviews? ☀ 16 Technical Key Areas ☀ How to choose from multiple Java job offers?

Select Category ▼

© Disclaimer

The contents in this Java-Success are copy righted. The author has the right to correct or enhance the current content without any prior notice.

These are general advice only, and one needs to take his/her own circumstances into consideration. The author will not be held liable for any damages caused or alleged to be caused either directly or indirectly by these materials and resources. Any trademarked names or labels used in this blog remain the property of their respective trademark owners. No guarantees are made regarding the accuracy or usefulness of content, though I do make an effort to be accurate. Links to external sites do not imply endorsement of the linked-to sites.

© 2016 Java-Success.com ↑ Responsive Theme powered by WordPress