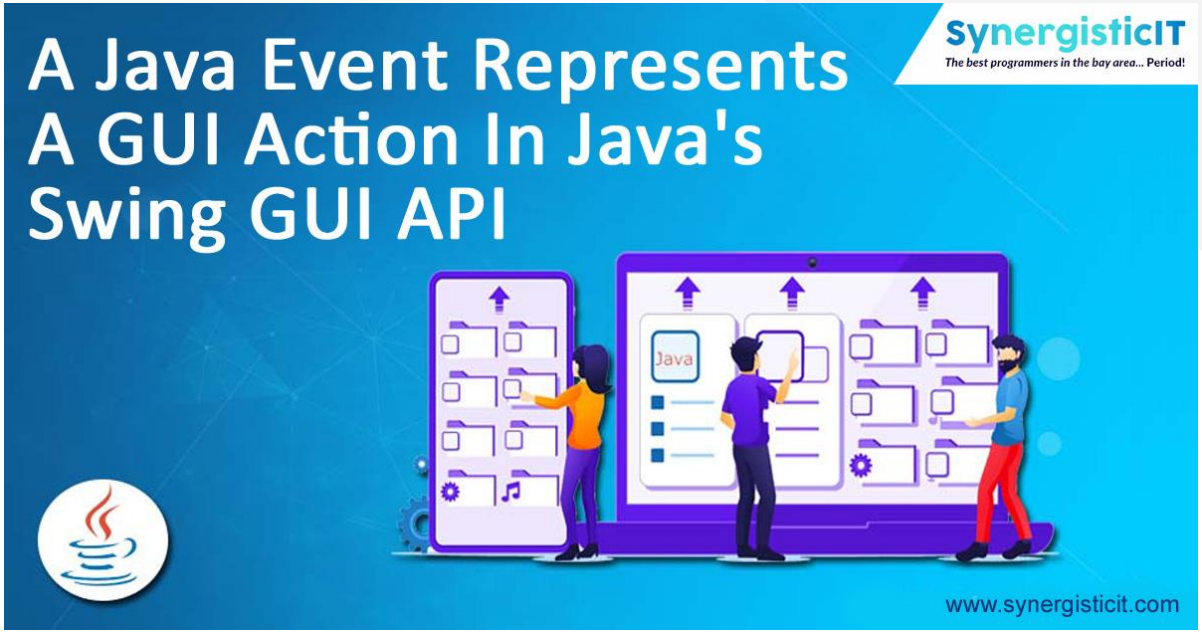
SynergisticIT

The best programmers in bay area... Period!

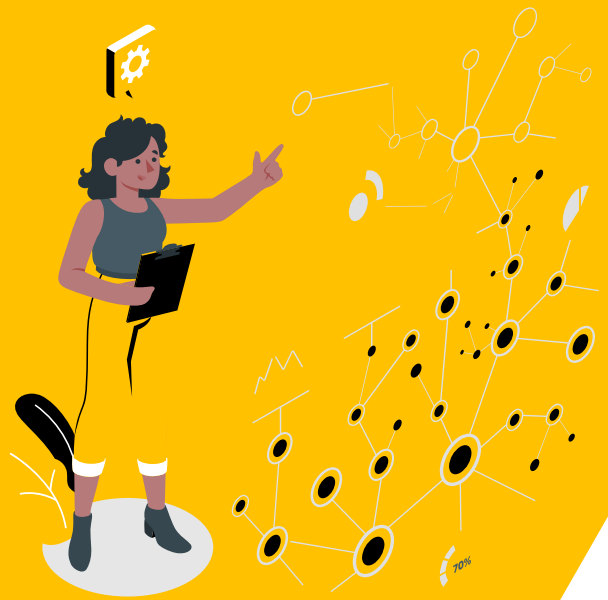
Java Event: Types and How They Work?



Java programming is the backbone of several websites, mobile applications, servers, and desktop GUI. Due to its platform independence, versatility, robustness, security features, it has been a core language for multiple startups and top tech organizations.

Many beginners like to start their coding career with Java as **Java coding** is simple. The simple English-like syntax makes it easy for the programmers to learn it with training and practice. Considering this, many institutions offer short-term training programs for Java.

Most **Java training in San Francisco** is 3-6 months long and can teach the learners basic and advanced programming skills from scratch. The training is aimed to help tech enthusiasts build a solid coding foundation.



You'll learn several concepts as part of your Java training. One such popular concept is an event related to Java's event handling mechanism in the Swing GUI library.

Java Swing is a lightweight GUI (graphical user interface) widget toolkit, offering a rich set of widgets and several packages useful for developing desktop Java applications.

Java Swing Event Handling

Swing GUI applications are event-driven, meaning they react to numerous events generated throughout their lives. These events result from user interaction with the application or other means.

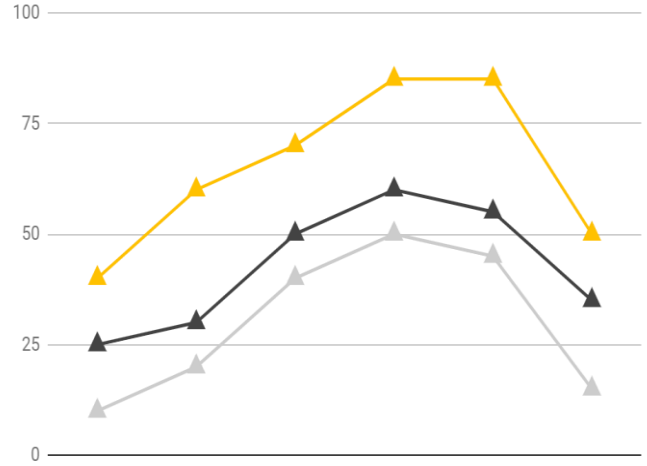
This article will help the learners understand what an event is, its types, and how event handling works.

What is an Event?

An event is a change in the state of an Object. It is triggered by any change in the Graphical User Interface (GUI. Whenever a user clicks on a button, moves the cursor, clicks on the combo box, types characters in the field, scrolls page, closes the window, etc., a relevant object event is created.

In simple words, an event is a signal that the operating system gives to the program whenever a user does something, or a change occurs in an instant in time. It indicates the user's interaction with the GUI components.

Let's understand this phenomenon with an example. Let's say you have a JButton, a push-button; so, whenever a user clicks on this button, an event will be triggered and created.



The created event will then be sent to the relevant event listener. For an event like this, ActionListener is the relevant listener, which has the implemented code defining the action to be taken if such an event occurs.

The ActionListener interface is a part of the Java.awt.event package and has only one method: actionPerformed (). This method is automatically invoked when you click on the JButton.

It is imperative to note that the component should be paired with the event listener. Otherwise, no action will initiate from the trigger caused by the change in the GUI.

An event is generally classified into two categories

- **Foreground Events –** These events occur when the user directly interacts with the GUI components by clicking on a button, moving the cursor, typing a character through a keyboard, scrolling the page, clicking on an item from a list, etc. They are generated by direct user interaction and triggered when any related change happens.

- **Background Events –** Unlike the foreground events, these events don't require the user's interaction. They generally happen due to interruption in the operating system, hardware/software failure, timer expiration, completion of the operation, etc. All these events have less to do with user interaction and happen in the background.

Event Handling

Event handling is an instrument by which the events can be controlled. It determines what action to take after an event takes place. **Java coding** follows the Delegation Event model to handle the events.

The delegation Event Model comprises two key elements: source and listener. Event Handling requires the source to be registered with the listener.

- **Source:** Events occur when you make changes in the GUI, and sources are how you make those changes. Examples of sources are the push button, checkbox, menu items, scrollbar, drop-down list, text components, etc.

- **Listener:** An event listener is responsible for handling the event. Each source has a specific event listener that manages the events triggered by their use.

Types of Events

There are various types of events and listeners in Java. Each event is associated with a matching listener. Let's look at some common events types in Java and their corresponding listeners.

- **ActionEvent:** It occurs when a graphical element is clicked, like a button or an element in a list. ActionListener is the event listener for the ActionEvent.

- **KeyEvent:** This Event is triggered when the user presses, types, or releases a key. The event listener related to it is, KeyListener.

- **MouseEvent:** This Event occurs after clicking or pressing the mouse. MouseListener is the related listener for it.

- **WindowEvent:** It is an event relating to changes happening in the window. For example, when a window closes, it is activated or deactivated. WindowListener is the Related listener for this event.

Remember that multiple event sources and listeners can interact with one another at a single point in time. It happens because a single listener can register multiple events of the same type.

One event listener can handle a set of components performing a similar action. Similarly, a single event can be related to multiple listeners. However, it's a rare occurrence.

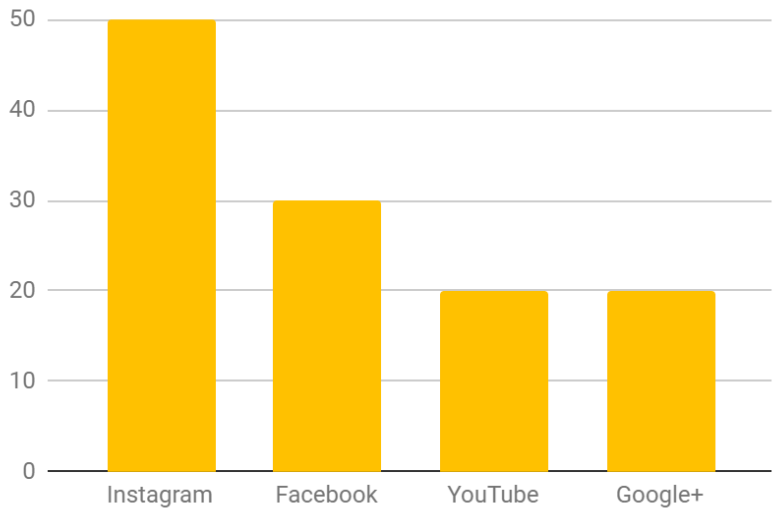
How Do Events Work?

Let's understand how an event works through a common type of Event, ActionEvent. ActionEvents are generated when a user clicks an item in the list or a button.

As a result of the user interaction, an ActionEvent object matching the corresponding event is created. The object includes the event source information and the action performed by the user. This information is passed to the related ActionListener object's method:

void actionPerformed(ActionEvent e)

After the method is executed, an appropriate GUI response is returned, which could be to open or close a dialogue box, give a digital signature, download a file, or any action available to the users in the interface.



Conclusion

This article discussed the event, its types, and event handling in detail. Similar to this concept, there are several concepts in Java. If you want to familiarize yourself with the advanced Java concepts through hands-on training, a **Java coding bootcamp** like **SynergisticIT** can help.

Experienced mentors will help you learn the OOP concepts, objects, and classes in Java, collections, interfaces, multithreading, and more through interactive lectures and projects.

Source: https://javacodingbootcamp.blogspot.com/2022/03/java-event-types-and-how-they-work.html

Thanks

Do you have
any questions?

Visit website:
www.synergisticit.com
admin@synergisticit.com
+510-550-7200