Explore the history, benefits, and drawbacks of AWT, Swing, and JavaFX GUI platforms, and discuss their compatibility with different Java versions. Compare similar applications built using Swing and JavaFX and document the code differences to illustrate their unique features. Engage in a concise discussion on the suitability of each platform for various development scenarios, considering their performance, flexibility, and future prospects.

The three Java UI frameworks are AWT, Swing, and JavaFX. AWT is the original GUI and remains a reliable framework. JavaFX is the replacement for Swing, and Swing was the replacement for AWT (Liang, 2019). Swing was a much “more robust, versatile, and flexible library” compared to AWT, but the newer replacement of JavaFX, which allows for richer GUI environments and options for “touch-enabled devices (Liang, 2019). With advancements in the platforms also comes increasing complexity in learning.

According to GeeksforGeeks (2024), AWT creates interfaces that are familiar to users and developers regardless of the operating system used. This means AWT is still valuable even though it has been replaced. AWT's features are components like “buttons, text fields, and checkboxes,” which help create the application and allow for user interaction (GeeksforGeeks, 2024). Other visual components like “menu, dialogs, and windows” are also available through the Abstract Window Toolkit; however, a drawback of AWT is that “components are based on the native platform” (GeeksforGeeks, 2024). The best time to utilize AWT is for simple applications, and it is compatible with modern Java standards (GeeksforGeeks, 2024).

Here is an example of coding with Java AWT (GeeksforGeeks, 2024):

**Java AWT**

**import** **java.awt.\***;

**public** **class** **HelloWorldAWT** **extends** Frame {

**public** **static** void main(String[] args) {

Frame frame = **new** Frame("Hello, World (AWT)");

Label label = **new** Label("Hello, World!");

frame.add(label);

frame.setSize(300, 100);

frame.setVisible(**true**);

}

}

As mentioned, Java Swing replaced AWT, but according to GeeksforGeeks (2024), it began as an “extension of AWT.” The Swing extension brought a new set of capabilities, including implementing Java and cross-platform properties (GeeksforGeeks, 2024). Swing has a richer selection of components, expanding upon the components of AWT and with new components like lists and tables (GeeksforGeeks, 2024). Swing is more customizable and consistent across platforms, with features like drag, drop, undo, and redo (GeeksforGeeks, 2024). Swing can be integrated with AWT and uses event-driven programming (GeeksforGeeks, 2024). Swing is better suited for more complex applications than AWT while still delivering performance and is compatible with modern Java standards (GeeksforGeeks, 2024).

Here is an example of coding with Java Swing (GeeksforGeeks, 2024):

**Java Swing**

**import** **javax.swing.\***;

**public** **class** **HelloWorldSwing** {

**public** **static** void main(String[] args) {

JFrame frame = **new** JFrame("Hello, World (Swing)");

JLabel label = **new** JLabel("Hello, World!");

frame.add(label);

frame.setSize(300, 100);

frame.setVisible(**true**);

}

}

The latest UI toolkit, JavaFX, is built in Java and offers an appealing user interface (GeeksforGeeks, 2024). JavaFX can support “multimedia, 2D and 3D graphics, and animation” (GeeksforGeeks, 2024). Like Swing, JavaFX offers the same components, richer components, and even more. JavaFX can be used on various platforms and supports CSS styling and FXML (GeeksforGeeks, 2024). JavaFX offers the best and richest experience for a user interface. Since JavaFX is the newest, it is the recommended choice and is up to date with the Java standards (GeeksforGeeks, 2024).

Here is an example of coding with JavaFX (GeeksforGeeks, 2024):

**JavaFX**

**import** **javafx.application.Application**;

**import** **javafx.scene.Scene**;

**import** **javafx.scene.control.Label**;

**import** **javafx.stage.Stage**;

**public** **class** **HelloWorldJavaFX** **extends** Application {

@Override

**public** void start(Stage stage) {

Label label = **new** Label("Hello, World!");

Scene scene = **new** Scene(label, 300, 100);

stage.setTitle("Hello, World (JavaFX)");

stage.setScene(scene);

stage.show();

}

**public** **static** void main(String[] args) {

launch(args);

}

}

**References**

GeeksforGeeks. (2024, February 1). *Java AWT vs Java Swing vs Java FX*. GeeksforGeeks. https://www.geeksforgeeks.org/java-awt-vs-java-swing-vs-java-fx/

Liang, Y. D. (2019). *Introduction to Java programming and data structures: comprehensive version*. Pearson. https://plus.pearson.com/home?utm\_source=ereader

**Assignment Requirements and Grading:**

* An initial post of approximately 250 words is due by **Thursday, 11:59 p.m. CST**.
* Submit your post by clicking on the assignment link above, then Create Thread. You must create a thread in order to view your peers' posts. Tip: Create your post in a Word document and then copy and paste your work into the thread.
* A minimum of three (3) responses, to the original threads of other students, of 100-200 words each are due by **Sunday, 11:59 p.m., CST**.
* To view the rubric grading criteria, click on the following link: [Discussion Board Grading Rubric](https://content.bellevue.edu/cst/csd/rubricdbv3.pdf).

Nardos, I really enjoyed reading your discussion post for this week. You are spot-on when you say that the Java GUI platforms have really gone through an evolution over the years. Even so, all platforms are still functional and can be utilized to this day, and better yet, the newest replacement of JavaFX can use these features in the previous platforms. Overall, I think that JavaFX is the safest bet to use when wanting to evolve the state of an application or program. But AWT and Swing could still have moments where they can be helpful since they are lighter weight.

Brett, I think you did a great job on your discussion board this week and identifying and explaining the different GUI Java frameworks. You are correct that AWT does have the benefits of being fast and lightweight but does not have all the capabilities that JavaFX has now. The touch support of JavaFX is necessary for many programs in the current technology environment. A huge positive of JavaFX is that it can also use the features of AWT and Swing, so if someone is used to the older frameworks, they still can integrate those prior functions if needed or wanted.

Arely, you did an excellent job of explaining the history of the different GUI platforms and diving deeper into it all. It is interesting how there was not much of a gap between AWT and Swing, but there was a much longer break between Swing and JavaFX. You are correct that all the frameworks come with positives and negatives. The examples you included for Swing and JavaFX accurately depict how to use each framework. Even though Swing is still relevant, I am sure it will become more obsolete as time passes. Do you foresee JavaFX getting replaced in the future?