

**SCHOOL OF
COMPUTING
CHENNAI**

Mercedes F1 Car Application

A CASE STUDY ON SWINGS

Submitted by

CH.EN.U4CYS21025

Jawakar Sri

Submitted to

Mr. Durga Rao

for the course

20CYS283 JAVA PROGRAMMING LAB

IV Semester (2022-2023)

B. Tech in Computer Science and Engineering (Cyber Security)

ACKNOWLEDGEMENT

I Jawakar Sri of Cyber Security bearing roll number CH.EN.U4CYS21025, would like to thank Mr. Durga Rao for providing me with this opportunity to a case study cum have fun with GUI in JAVA using SWINGS.

I am extremely grateful to him for his constant motivation and helping in all aspects while I made this Fitness Application, a dummy project.

I am also thankful to director Shri. I.B. Manikantan and Principal Dr. V. Jayakumar for always guiding us in the right direction.

Last but not the least, humble pranams to All Mighty and Amma, for their constant blessings upon all of us.

INDEX

Sr. No.	Topic	Page No.
01.	Abstract	03
02.	What is Swing?	03
03.	Architecture of Swings	04
04.	Relationship with AWT	06
05.	Components of Swing's Class	06
06.	Code	09
07.	Output Screenshots	09,10

ABSTRACT

This is a case study where in I will be using Swings in Java to create a simple and functional GUI for a F1 car application. The case study consists of a code and its output screenshots and sample demonstration under a video.

WHAT IS SWING?

Swing is a GUI widget toolkit for Java. It is part of Oracle's Java Foundation Classes (JFC) – an API for providing a graphical user interface (GUI) for Java programs.

Swing was developed to provide a more sophisticated set of GUI components than the earlier Abstract Window Toolkit (AWT). Swing provides a look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform. It has more powerful and flexible components than AWT. In addition to familiar components such as buttons, check boxes and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables, and lists.

Swing is a highly modular-based architecture, which allows for the "plugging" of various custom implementations of specified framework interfaces: Users can provide their own custom implementation(s) of these components to override the default implementations using Java's inheritance mechanism via LookAndFeel.

ARCHITECTURE

Swing is a component-based framework, whose components are all ultimately derived from the JComponent class. Swing objects asynchronously fire events, have bound properties, and respond to a documented set of methods specific to the component. Swing components are JavaBeans components, compliant with the JavaBeans specification.

Configurable

Swing's heavy reliance on runtime mechanisms and indirect composition patterns allows it to respond at run time to fundamental changes in its settings. For example, a Swing-based application is capable of hot swapping its user-interface during runtime.

Furthermore, users can provide their own look and feel implementation, which allows for uniform changes in the look and feel of existing Swing applications without any programmatic change to the application code.

Lightweight UI

Swing's high level of flexibility is reflected in its inherent ability to override the native host operating system (OS)'s GUI controls for displaying itself. Swing "paints" its controls using the Java 2D APIs, rather than calling a native user interface toolkit. Thus, a Swing component does not have a corresponding native OS GUI component, and is free to render itself in any way that is possible with the underlying graphics GUIs.

However, at its core, every Swing component relies on an AWT container, since (Swing's) JComponent extends (AWT's) Container. This allows Swing to plug into the host OS's GUI management framework, including the crucial device/screen mappings and user interactions, such as key presses or mouse movements. Swing simply "transposes" its own (OS-agnostic) semantics over the underlying (OS-specific) components. So, for example, every Swing component paints its rendition on the graphic device in response to a call to `component.paint()`, which is defined in (AWT) Container. But unlike AWT components, which delegated the painting to their OS-native "heavyweight" widget, Swing components are responsible for their own rendering.

This transposition and decoupling is not merely visual, and extends to Swing's management and application of its own OS-independent semantics for events fired within its component containment hierarchies. Generally speaking, the Swing architecture delegates the task of mapping the various flavors of OS GUI semantics onto a simple, but generalized, pattern to the AWT container. Building on that generalized platform, it establishes its own rich and complex GUI semantics in the form of the JComponent model.

RELATIONSHIP OF SWING'S WITH AWT



COMPONENETS OF SWING CLASS

Class	Description
Component	A Component is the Abstract base class for about the non menu user-interface controls of SWING. Components are represents an object with graphical representation
Container	A Container is a component that can container SWING Components
JComponent	A JComponent is a base class for all swing UI Components In order to use a swing component that inherits from JComponent, component must be in a containment hierarchy whose root is a top-level Swing container.

JLabel	A JLabel is an object component for placing text in a container.
JColorChooser	A JColorChooser provides a pane of controls designed to allow the user to manipulate and select a color.
JCheckBox	A JCheckBox is a graphical(GUI) component that can be in either an on-(true) or off-(false) state
JRadioButton	The JRadioButton class is a graphical(GUI) component that can be in either an on-(true) or off-(false) state. in the group
JList	A JList component represents the user with the scrolling list of text items.
JComboBox	A JComboBox component is Presents the User with a show up Menu of choices.
JTextField	A JTextField object is a text component that will allow for the editing of a single line of text.
JPasswordField	A JPasswordField object it is a text component specialized for password entry
JTextArea	A JTextArea object s a text component that

	allows for the editing of multiple lines of text
ImageIcon	A ImageIcon control is an implementation of the Icon interface that paints Icons from Images.
JButton	This class creates a labeled button
JScrollbar	A JScrollbar control represents a scroll bar component in order to enable users to Select from range values
JOptionPane	JOptionPane provides set of standard dialog boxes that prompt users for a value or Something
JFileChooser	A JFileChooser it Controls represents a dialog window from which the user can select a file.
JProgressBar	As the task progresses towards completion, the progress bar displays the tasks percentage on its completion
JSlider	A JSlider this class is lets the user graphically(GUI) select by using a value by sliding a knob within a bounded interval.
JSpinner	A JSpinner has a single child component that's responsible for displaying and potentially changing the current element.

CODE

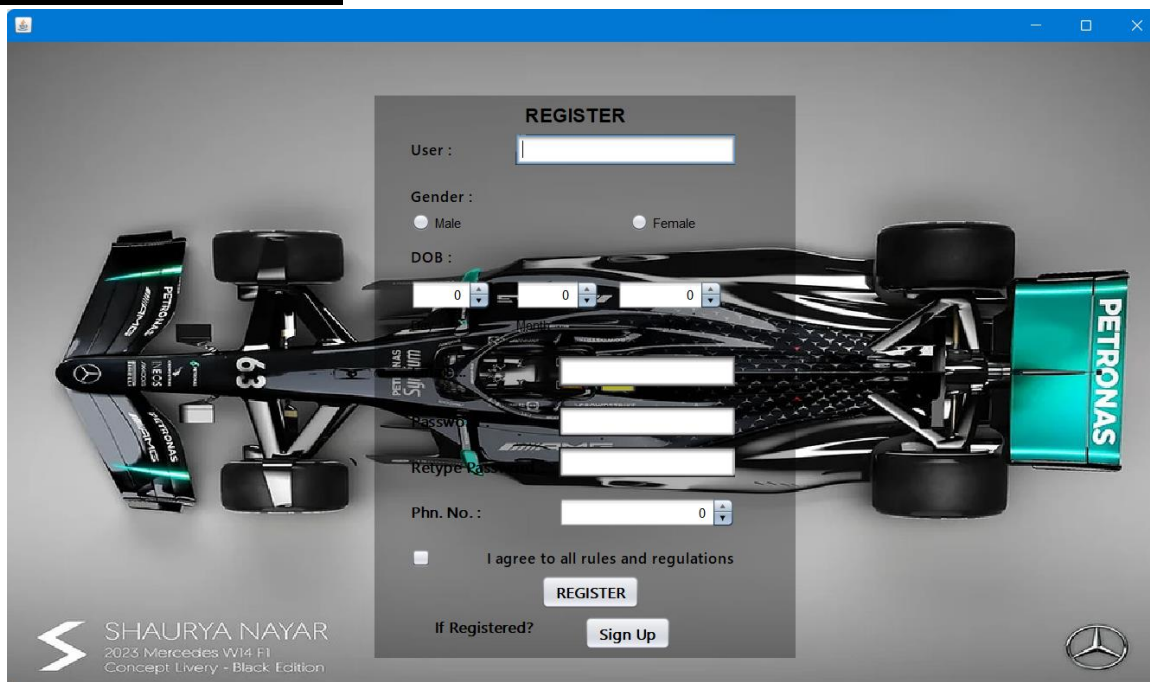
<https://github.com/jawakarsri/F1-Car-Application-on-java-swings>

OUTPUT SCREENSHOTS

LOGIN PAGE:



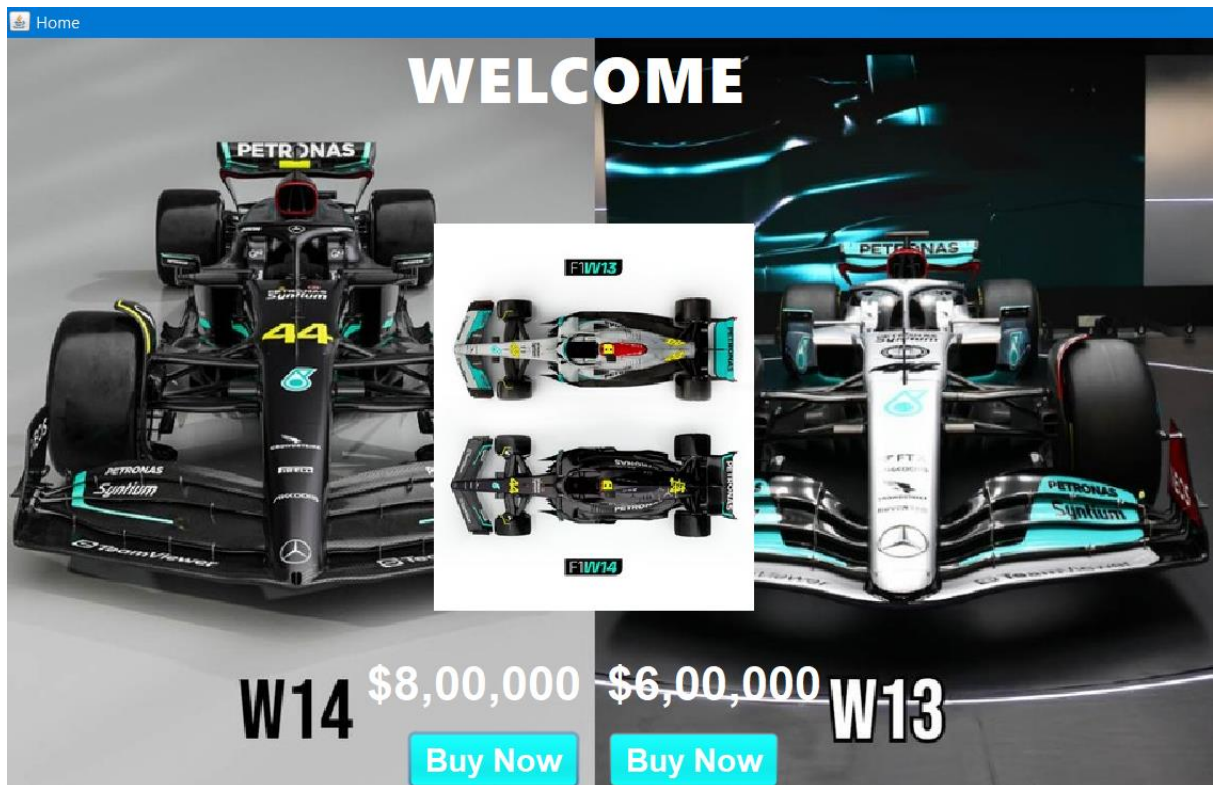
REGISTER PAGE:



HOME PAGE :

Home

WELCOME



The image displays two Mercedes-AMG Petronas Formula 1 cars. On the left is the W14, a black and teal car with the number 44. On the right is the W13, a white and teal car. A central inset shows top-down views of both cars, labeled FW13 and FW14.

W14 \$8,00,000 **W13** \$6,00,000

[Buy Now](#) [Buy Now](#)