CS2 Skin Rating GUI

line 1: 2nd Given Name Surname  
line 2: *dept. name of organization   
(of Affiliation)*  
line 3: *name of organization   
(of Affiliation)*line 4: City, Country  
line 5: email address or ORCID

Abstract—This report presents the design, implementation, and analysis of a Java-based application for rating weapon skins from the Counter-Strike 2 (CS2) game. The graphical user interface (GUI) application leverages Java Swing to create an intuitive dark-themed user interface that allows users to browse, view, and rate various in-game weapon skins. This paper explores the system architecture, object-oriented programming principles employed in the development process, and implementation details. The application demonstrates practical application of core object-oriented programming concepts including inheritance, encapsulation, abstraction, polymorphism, and composition through a modular design that facilitates maintenance and future expansion.

Keywords—object-oriented programming, Java Swing, GUI development, component-based design, inheritance.

# Pendahuluan

Aplikasi CS2 Skin Rating dikembangkan menggunakan bahasa pemrograman **Java** dengan library GUI **Java Swing**. Tujuan utama dari proyek ini adalah membangun antarmuka pengguna interaktif yang menampilkan item grafis berbentuk skin senjata, serta memungkinkan pengguna memberikan penilaian terhadap item yang ditampilkan.

Pengembangan aplikasi ini menekankan pada penerapan **konsep Object-Oriented Programming (OOP)**, serta pemisahan komponen berdasarkan tanggung jawabnya masing-masing. Dengan pendekatan ini, aplikasi menjadi lebih terstruktur, mudah dipelihara, dan dapat dikembangkan lebih lanjut.

Proyek ini juga memanfaatkan **Apache NetBeans** untuk menyusun layout awal dengan drag-and-drop, yang kemudian disempurnakan di **Visual Studio Code** dengan penambahan fitur, styling, dan optimisasi struktur kode. Kombinasi dua alat ini mempercepat workflow pengembangan dan memungkinkan fokus pada kualitas logika program.

Aplikasi ini dikembangkan dengan menggunakan Java Swing, sebuah toolkit GUI untuk Java. Tujuan utama dari aplikasi ini adalah:

1. Create an intuitive user interface with a dark theme consistent with gaming aesthetics.
2. Implement a modular, maintainable codebase following object-oriented principles
3. Allow users to browse skin collections by weapon category
4. Display detailed information about each skin and enable user rating

This report examines the technical implementation of these objectives through the application's architecture, design patterns, and core OOP principles..

## System Architecture

The CS2 Skin Rating application follows a component-based architecture typical of Java Swing applications. The system is composed of multiple classes, each responsible for specific UI components and functionality:

## Core Components

1. **CS2SkinRating**: The main application window class that inherits from JFrame and serves as the container for all other components.
2. **TopBarPanel**: Manages the title bar and utility buttons (Discord, Ko-Fi).
3. **SidebarPanel**: Handles navigation between different weapon categories.
4. **CardDisplayPanel**: Displays skin images in a grid layout.
5. **SkinCard**: Represents individual skin items with image and information.
6. **BottomBarPanel**: Contains action buttons for rating and saving.
7. **Button**: A utility class for creating consistent buttons throughout the application.

.

## Component Relationships

The components are arranged hierarchically, with the main CS2SkinRating class containing all others. The application uses BorderLayout as its primary layout manager, with components positioned as follows:

* TopBarPanel (NORTH)
* SidebarPanel (WEST)
* CardDisplayPanel in a JScrollPane (CENTER)
* BottomBarPanel (SOUTH)

The SideBarPanel communicates with the CardDisplayPanel to update the displayed content when a category is selected.

# Implementation Details

## User Interface Design

The application features the dark-themed UI with consistent styling across components :

* Background colors ranging from dark to medium gray (RGB: 45,45,45 to 70,70,70)
* White or light gray text for contrast
* Consistent button styling through the Button utility class
* Grid layout for skin cards with appropriate spacing and borders
* Scrollable main content area to accommodate variable content size

## Component Implementation

The application is divided into seven distinct Java classes:

|  |  |
| --- | --- |
| CS2SkinRating.java | Main application window |
| TopBarPanel.java | Title and utility buttons |
| SidebarPanel.java | Navigation panel |
| Button.java | Button styling utility |
| CardDisplayPanel.java | Content display area |
| SkinCard.java | Individual skin component |
| BottomBarPanel.java | Action buttons |

Each class handles Specific UI elements and behaviors:

## **CS2SkinRating**: Initializes the main window, sets up layout and contains the main() method.

## **SkinCard**: Loads and displays skin images with dynamic scaling, provides interactive behavior.

## **CardDisplayPanel**: Manages the collection of SkinCard instances in a grid layout.

## **SidebarPanel**: Contains category buttons with event handling for content switching.

## Event Handling

The application implements event handling primarily through ActionListener for buttons and MouseListener for skin cards:

* SidebarPanel buttons trigger content updates in CardDisplayPanel.
* SkinCard components show detailed information in a dialog when clicked.
* Rating and Save buttons in BottomBarPanel are prepared for future implementation.

# Object Oriented Programming Principles

The application demonstrates several key OOP principles that enhance its structure and maintainability:

## Classes and Objects

The codebase is organized into classes that serve as blueprints for objects. For example:

* SkinCard card = new SkinCard(...) creates instances of skin display components.
* Each panel class (TopBarPanel, SidebarPanel, etc.) represents a distinct UI component.

## Inheritance

Inheritance is used extensively to extend Java Swing components:

* CS2SkinRating extends JFrame - The main window inherits from Swing's JFrame.
* TopBarPanel extends JPanel - Custom panels extend Swing's JPanel.
* SidebarPanel extends Button - Navigation panel extends the custom Button class.
* Button extends JPanel - Custom button implementation extends JPanel.

## Encapsulation

#### Component data and behaviors are encapsulated within their respective classes:

#### **SkinCard** encapsulates image loading, scaling, and display logic.

#### **TopBarPanel** encapsulates title and button creation/styling.

#### **SidebarPanel** encapsulates category navigation logic.

## Abstraction

#### The application employs abstraction to hide implementation complexity:

* The **Button.createButton()** factory method abstracts button creation and styling.
* **CardDisplayPanel.updateDisplay()** abstracts the process of loading and displaying skins.
* JSwing components themselves provide abstraction over low-level GUI operations.

## Polymorphism

#### Polymorphism is demonstrated in several ways:

* Method overriding: The anonymous Runnable implementation overrides the run() method.
* The add() method accepts various Component subtypes (JPanel, JLabel, JButton).
* Method overloading: Button.createButton() has two variants with different parameters.

## Composition

#### The application uses composition to build complex object from simpler ones:

* CS2SkinRating contains TopBarPanel, SidebarPanel, CardDisplayPanel, and BottomBarPanel
* CardDisplayPanel contains multiple SkinCard instances
* TopBarPanel contains JLabel and JPanel components

# Result and Discussion

The CS2 Skin Rating application successfully implements a functional GUI for browsing and viewing CS2 weapon skins. The application features:

* A responsive dark-themed user interface
* Navigation between weapon categories (AWP, Knives)
* Grid display of skin images with names
* Detailed information dialogs for individual skins

## Strengths

1. **Modularity**: The component-based architecture allows for independent development and testing of UI elements.
2. **Maintainability**: Clear separation of concerns makes the code easier to maintain and extend.
3. **Consistency**: The Button utility class ensures consistent styling across the application.
4. **Scalability**: New skin categories can be easily added through the existing framework.

## Limitations And Future Work

1. **Persistence**: The current implementation lacks data persistence for user ratings.
2. **Limited Categories**: Only AWP and Knife categories are implemented.
3. **Rating Functionality**: The rating feature is present in the UI but not fully implemented.

Future enhancements could include:

1. Database integration for persistent storage of skins and ratings
2. User authentication system
3. Expanded skin categories
4. Filtering and sorting capabilities
5. Statistical analysis of community ratings

# Conclusion

This paper presented the design and implementation of a Java Swing application for browsing and rating CS2 weapon skins. The application demonstrates practical application of object-oriented programming principles through a modular, component-based architecture.

The CS2 Skin Rating application successfully meets its design objectives by providing an intuitive user interface for skin browsing and viewing. The adherence to OOP principles has resulted in a maintainable and extensible codebase that can be further enhanced with additional features in future iterations.

The implementation showcases how Java Swing can be used to create modern, themed applications with complex UI components through proper application of object-oriented design principles.

# References

1. Oracle, "Java Swing (GUI) Programming: Examples," Oracle Documentation, 2023. [Online]. Available: [https://docs.oracle.com/javase/tutorial/uiswing/](vscode-file://vscode-app/c:/Users/rifki/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html).
2. E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.
3. B. Eckel, *Thinking in Java*, 4th ed. Prentice Hall, 2006.
4. J. Bloch, *Effective Java*, 3rd ed. Addison-Wesley Professional, 2018.
5. A. Potdar, S. Haramritpal, and W. Chen, "Object-Oriented Design in GUI Applications," *Journal of Computer Science Applications*, vol. 12, no. 3, pp. 234-249, 2022.
6. Valve Corporation, "Counter-Strike 2," 2023. [Online]. Available: [https://www.counter-strike.net/](vscode-file://vscode-app/c:/Users/rifki/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html).
7. D. Alexander, "The Economics of Virtual Items in Video Games," *International Journal of Game Studies*, vol. 7, no. 2, pp. 55-67, 2021.
8. M. Robinson, *Java for GUI Applications*. Springer, 2020.
9. G. Booch, *Object-Oriented Analysis and Design with Applications*, 3rd ed. Addison-Wesley, 2007.
10. S. P. Reiss, "WYSIWYG Web and GUI Design," *IEEE Software*, vol. 27, no. 5, pp. 71-74, 2010..