KOÇ ÜNİVERSİTESİ
COMPUTER ENGINEERING

# COMP132: Advanced Programming

## Programming Project Report

# Uno Card Game Simulation

*Enes Goktas, 75721*

*Spring 2024*

## General Demo Information:

**List of Users**

The application allows for multiple user accounts, each identified by a unique nickname and a password.

**Information about Each User**

Profiles track individual scores, games played, wins, and losses, contributing to a competitive leaderboard.

**List of Existing Game Sessions**

Game sessions are tracked and can be saved or resumed, supporting multiple ongoing games across different user accounts.

## Application Usage Information

**Sign up/Login Guide**

Users create an account or log in using a secure system that manages authentication and session handling.

**User's Guide**

After login, the user interface allows joining or creating game sessions, interacting through a GUI for game management.

**Game Session Gameplay**

Adheres to Uno rules where players match cards by color or number, utilize wild cards, and follow turn-based gameplay.

**Saving/Loading Game Sessions**

Functionality to save the game state to a file and load it later, ensuring continuity.

**Leaderboard**

Ranking based on scores collected from game wins.

**How to Check Log Information**

Logs of user activities are accessible for review, ensuring transparency and helping resolve disputes.

## Project Design Description

**Class Relations and Package Details**

The project is structured into several packages, each containing classes that handle specific aspects of the application:

- **core Package:**
    - **Player (Abstract Class):** Base class for **HumanPlayer** and **SimpleBot**, handling common player functionality.
    - **Card and Deck Classes:** Manage the cards in play and the deck operations like shuffling and drawing cards.
    - **GameManager:** Coordinates the game's rules, player turns, and the game state transitions.

- **bots Package:**
  - **SimpleBot:** Extends **Player**, providing AI behavior for automated game playing against human players.
- **gui Package:**
  - **GameFrame and LoginFrame:** Provide the graphical user interface for gameplay and user authentication.
  - **LeaderboardFrame:** Displays user rankings and statistics.
- **users Package:**
  - **User and UserManager:** Manage user data, including registrations, logins, and scoring.
- **utils Package:**
  - **FileHandler and FileLogger:** Handle saving game states and logging application events, respectively.

**Inheritances, Type Hierarchies, and Interfaces**

Using Java's object-oriented features, the application leverages inheritance, abstract classes, and interfaces to promote code reusability and encapsulation:

- **Player:** Serves as an abstract base for **HumanPlayer** and **SimpleBot**, defining shared methods that are then customized in subclasses.
- **GameManager and GUI Components:** Interface with both the core logic and presentation layers, managing the flow of the game and user interactions.

**GUI Components**

Built using Java Swing, the GUI is crucial for user interaction, providing buttons, menus, and dialogs that facilitate gameplay and navigation.

### .txt File Processing Details

User and game session data are managed through text file processing, allowing for persistent storage of game states and user profiles.

### Game Session Loop Implementation

The loop within **GameManager** checks game state, enforces rules, and manages the transitions between different players' turns.

### Computer Bot Implementation

**SimpleBot** uses strategic logic to play cards effectively against human opponents, simulating a challenging game environment.

### References


- Java Swing Documentation
- Official Uno Game Rules