COMP 132: Advanced Programming

Spring 2024

Programming Project

Due: May 15, 2024, 11:59 pm (Late submissions will not be graded.)

Uno Card Game: Simulation Design and Development

This is an **individual programming project**. All work you submit must belong to you. For any questions regarding this programming project, use the corresponding Blackboard discussion forum and post your questions there. Besides, the project Info Session will be conducted by the TAs. The goal of this project is to practice OOP in-depth and learn by yourself how to use and deal with GUIs in Java.

You must include and sign (by writing your name and student ID number) the following Pledge of Honor statement at the beginning of your main method source code file. Otherwise, your project will not be graded.

```
/************** Pledge of Honor ****************************************

I hereby certify that I have completed this programming project on my own
without any help from anyone else. The effort in the project thus belongs
completely to me. I did not search for a solution, or I did not consult any
program written by others or did not copy any program from other sources. I
read and followed the guidelines provided in the project description.

READ AND SIGN BY WRITING YOUR NAME SURNAME AND STUDENT ID

SIGNATURE: <Name Surname, Student id>

*********************************************************************/
```

Blackboard submission: You should submit one ***.zip** file named your <name-surname> (for example, Ayse-Yilmaz.zip) that contains the whole Java project folder and a report file as a PDF. The report must be written in two parts. The first part should be a guideline describing the execution steps of your application. In the second part, you should describe your project design, implemented types and their hierarchies, graphical user interfaces, and references to the resources you have used.

Please note that the report grade is part of your project grade. You should prepare it as clearly as possible. The report has no page limit, and a report template is provided.

Bonus: The top projects with the best GUI design, report, and demonstration will get bonus points.

The students who get Bonus will have a chance to play UNO with TAs.

Project Overview

In this project, you'll design and develop a software application that simulates **the Uno Card Game**. The application will allow users to play Uno against computer-controlled opponents via a graphical user interface (GUI). The game will include card drawing, card playing, special card effects, and determining the winner and score. The rules and details of the operations not mentioned are left to your design. Make sure to state your assumptions in this direction clearly. Do not try to extend the project too much by adding any other unrequired large-scale features. The features of the Uno card game simulation are listed below.

Login Page

- Users can create accounts with unique usernames and passwords.
- Existing users can log in with their username and password to access the game.

Main Menu

- After logging in through the login page, users must see a **leaderboard** that shows all the existing users' usernames and the total score they have accumulated.
    - By clicking on a specific user on the leaderboard, the user must see all the tracked statistics on that specific user (Details are below under the User Statistics section).
- Users can continue an existing game saved beforehand with a specified session name (Details are below under the Game Save/Load section) or create a new game on the same page.
- A new game can be created by selecting the number of players between 2 and 10 and a session name. One is the user, and the others will be computer bots.

Game Session

In the game session, a user must be able to:

- See the session name
- See and interact with their cards,
- See the other player's names with their card count on their hands,
- Interact with the draw pile and see how many cards are left within the draw pile,
- See the last card that was put on top of the discard pile,
- See the direction of the play (clockwise/counter-clockwise)
- Interact to declare "Uno" if the player will have one card in their hand after their play.
- Interact to point out if a player did not say "Uno" before the next player's turn to penalize the player.

Cards

The players within a game interact with the cards to play the Uno game. The interactions and the effects change based on the cards' type and color. A Uno deck consists of 108 cards, of which there are 76 Number cards, 24 Action cards, and 8 Wild cards. The Uno cards have four color "suits", which are red, yellow, blue, and green.

The specifications of the types of cards are as follows:

- **Number card**:
    - These cards have only a color and a number between 0 and 9.
    - The scoring value for each of these cards is based on what number is shown.
    - Within a deck, each color contains 19 cards, one number 0 card, and two sets of cards numbered 1-9.
- **Action card**:
    - Within a deck, there are two of each action card in each color, which are:
        - **Draw two**: the next player must draw 2 cards and miss their turn
        - **Reverse**: reverses the direction of play
        - **Skip**: the next player forcefully skips their turn.
    - The score value for these cards is 20 points each.

- **Wild card**:
    - Wild cards allow the player to choose the next color to be played by the other players. There are two types of wild cards, four of each type within a deck. These types are:
        - Wild: only changes the color that continues play.
        - Wild draw four: changes the color that continues play and forces the next player to draw 2 cards and miss their turn.
    - The score value for these cards is 50 points each.

Computer Bots

To simulate a Uno game without other users, there must be “bots” considered a computer-based player capable of playing Uno like a human player. The Bots must be able to make decisions and play their turn based on the current (or even past) game state. In this project, we are not asking for an advanced AI for the “bot”; instead, we are asking for a simple implementation with some randomness for a game session to function and continue normally.

Gameplay Rules

The game's objective:
Be the first player to eliminate all the cards in your hand and keep as many cards as possible in the opponents' hands to gather as many points as possible based on the remaining opponents' card values.

Game Initialization:
Shuffle the card deck and deal seven cards for each player. Put the remaining cards upside down to form a draw pile. Turn over the top card of the draw pile to start the discard pile and Start the game with the user's turn. The game starts taking turns in clockwise order.

Main game loop:
On a player's turn, they must play one card from their hand that matches the top card's color, number, or symbol on the discard pile. For example, if the top card of the discard pile has a green card with a 5, you have to place either a green card or a card with a 5. You can also play a Wild card (which can alter the current color in play).

If a player cannot play a card based on the criteria mentioned previously or does not want to play even though he can, the player must draw a card from the draw pile until a drawn card meets the criteria. If a drawn card meets the criteria, the player can play it or skip his turn.

If a player is going to have one card left after their turn, during their turn, the player must say "Uno". Failure to do so may result in a penalty if another player points it out to the player before the next player's turn. The penalty is taking two new cards from the Draw Pile.

If the Draw Pile is empty, the cards inside the Discard pile, apart from the last one, are shuffled and put into the draw pile.

Game end and Scoring
The game continues until one player has no cards left in their hands. That player wins the game and earns points based on the cards remaining in other players' hands. The value for each card is described earlier in the Cards section.

User Statistics

The application must keep track of game statistics for each user, which are:

- number of wins/losses
- number of games played
- The total score they accumulated over all games played.

With these statistics, the game application can also display derived statistics such as:

- Average score per game played
- The win/loss ratio

Game Logs

Log every event or action every player (bots included) within the game makes for review and analysis for each game a user plays. Card plays, special card effects, penalties, and game outcomes are examples of game events or actions.

Important: The logging output for each game must be a text file, and you are not allowed to serialize the Java objects.

Game Save/Load

In the game session, the application user must have the option to save the game whenever they want to or want to quit that game such that they can load that game through the Main Menu to resume it.
There must be a list of saved games to load per user to be shown on the Main Menu, and once the game is finished, you have to update the user's overall game statistics.

Implementation Guidelines

- **OOP Principles:** Utilize object-oriented programming principles such as inheritance, polymorphism, and encapsulation for clean and maintainable code.

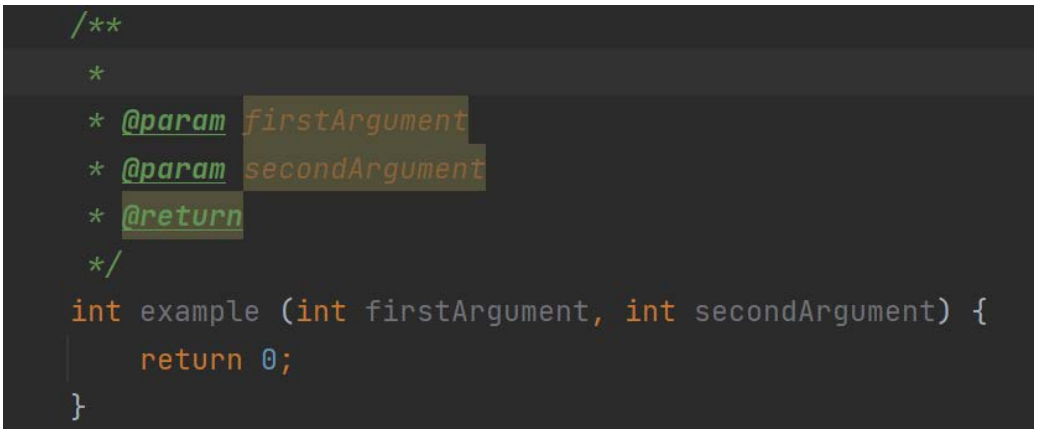
  Note: You have to decide on each attribute you have, whether static or final, and its access modifier, whether it will be private, protected, or public. Regarding getters and setters, add only what is needed. Do not add all getters and setters for each class; it is not a good thing to do.

- **Exception Handling:** Include robust mechanisms to gracefully handle unexpected user inputs and edge cases.
- **File Handling:** Implement file handling to store game data and logs in text files for persistence. So, a game should be stored and loaded from where it is paused. There should be pause, save, and load buttons.
- **Documentation:** Provide comprehensive documentation for classes, methods, and functionalities using Javadoc or similar tools.
- **External Libraries:** Any usage of non-standard libraries is **PROHIBITED**.
- **GUI Designers:** You can use GUI designers, not to have to write the GUI code just by yourself.
- **Testing and Debugging:**
  - Thoroughly test the application to ensure it functions as expected in various scenarios.
  - Debug any issues or errors encountered during testing to deliver a stable and reliable product.

In your project design and implementation, you should:

- Use inheritance and type hierarchies.
- Apply polymorphism: abstract classes, interfaces.
- Use Java Collections Framework types.
- Use Java Swing GUI components.
- Use Exception Handling and File Processing (player stats and logging)
- Apply code documentation
  - Explain what each method does, the method's parameters, and the return type.
  - Your documentation for a method should be explanatory to a first-time reader yet concise enough that a reader should understand it within a few lines. You can get inspiration from Java Language's official documentation.
  - You may use Javadoc for documentation. quick introduction or https://www.oracle.com/java/technologies/javase/javadoc-tool.html. Then, you can generate the Javadoc website for your project by using the tools your ide provides and show the "index.html" file.

Hint: your IDE can generate a documentation string (docstring for short) template by typing /** above a method declaration and hitting enter.



Note: Not providing sufficient documentation would lead to grade reduction, even if your code is perfect.

Design and implement a GUI using the **Java Swing Framework**. You can design your GUI using components such as JTextField, JButton, JComboBox, and JOptionPane. You can use any layout, but your **GUI** should be easy to understand and use.

To meet the project demo requirements, your application must *have at least the following*:

- At least three created users with different user information (such as unique usernames, can initialize users with a predefined total score, win/loss count, etc.)
- At least a game session is saved beforehand per user. (Not finished, saved game sessions)

During the **demonstration** of your project, you should show the execution of the following operations via your application's **Graphical User Interface**:

- User registration and login process.
- Joining existing and creating new game sessions.
- Gameplay features include card drawing, playing, and special card effects.
- Interaction with AI opponents.
- Tracking game statistics and leaderboard.
- Viewing game logs and history.

Good Luck