Durak Card Game - Project Documentation

Table of Contents

Contents

1. Introduction

The Durak Card Game project is a final group assignment for the Object-Oriented Programming 3 (OOP-3) course. The goal is to develop a 2-player version of the traditional Russian card game Durak using C#. The game features a graphical user interface and basic computer AI, demonstrating key object-oriented programming principles.

2. Project Setup

Prerequisites

- Visual Studio or any C# IDE
- .NET Framework installed

Installation

1. Clone the repository from GitHub:
2. Open the solution file `DurakCardGame.sln` in Visual Studio.
3. Build the project to restore the dependencies.

3. Usage

Starting the Game

1. Run the project from Visual Studio.
2. The game window will open, displaying the main menu.
3. Click "Start Game" to begin playing.

Game Controls

Mouse Click: Select and play cards.
Menu Navigation: Use mouse to navigate through options.

4. Game Rules

Durak is played with a deck of 36 cards (6 to Ace in each suit). The objective is to avoid being the last player with cards in hand. The game includes attacking and defending phases, with players taking turns to play cards.

5. Code Structure

Main Components

- `Program.cs`: Entry point of the application.
- `Game.cs`: Core game logic and rules.
- `Player.cs`: Represents a player in the game.
- `Card.cs`: Defines properties and behaviors of a card.
- `Deck.cs`: Manages the deck of cards.
- `AIPlayer.cs`: Implements the AI logic for the computer opponent.
- `MainForm.cs`: Handles the graphical user interface.

Class Descriptions

- **Program:** Initializes the game and starts the main application loop.
- **Game:** Manages the game state, turn order, and win conditions.
- **Player:** Contains player-specific information and actions.
- **Card:** Represents individual cards with suit and rank.
- **Deck:** Handles shuffling and dealing of cards.
- **AI Player:** Extends `Player` with AI decision-making capabilities.
- **Main Form:** Implements the GUI, including rendering and user interactions.

6. Object-Oriented Principles

Encapsulation

Encapsulation is achieved by keeping the data (properties) private and providing public methods for access and modification. For example, the `Card` class encapsulates card properties and provides methods to access them.

Abstraction

Abstraction is used to hide complex implementation details from the user. The `Deck` class abstracts the details of shuffling and dealing cards.

Polymorphism

Polymorphism allows the program to process objects differently based on their class. The `Player` class is extended by `AIPlayer` to provide different behaviors for human and AI players.

Inheritance

Inheritance is utilized by creating a base `Player` class and extending it with `AIPlayer` to add specific functionalities for the AI opponent.

7. AI Implementation

The AI logic is implemented in the `AIPlayer` class. It makes decisions based on the current game state, including which card to play during the attack and defense phases.

8. Graphical User Interface (GUI)

Design

The GUI is designed to be simple and user-friendly, with a focus on accessibility and usability. It includes:

- Main menu
- Game board
- Score display
- Player and AI hands

Accessibility

The interface adheres to accessibility standards, including:

- High contrast for readability
- Tooltips for buttons and interactive elements
- Keyboard navigation support

9. UML Diagram