Prepared By: Ildar Mamin
BIT student

Ohjelmoinnin perustaito R0027-3030

Technical Report Step by Step

Main tools used in this project:

15.12.2024

APPLICATION: Blackjack Game

The IDEA:

After experimenting with various projects, I finally decided to create a game based on a popular card game. I was inspired by a project I came across from someone else and decided to create a similar version for this course using Python.

APPLICATION: Blackjack Game

Description

This project is a Python-based console application simulating the classic card game Blackjack, where players compete against a virtual dealer to achieve a hand value close to 21 without exceeding it.

The game uses object-oriented programming principles to model cards, decks, and player hands, ensuring a realistic representation of Blackjack mechanics. It features betting, dynamic Ace handling, dealer behavior, and chip management.

The project is currently designed to run exclusively in an IDE or terminal, as it lacks a graphical user interface (GUI). Despite this, it provides an engaging and interactive experience while demonstrating fundamental programming techniques, including randomization, loops, and conditionals.

Note1

Spent over 25h for completing this assignment

Tried to create 7 projects before, and even created my API in Openai API, fore making a chatbot

1. Setup: Classes and Initialization

Card Class:

- Represents a playing card with a suit (e.g., Hearts) and rank (e.g., Ace, 2, Jack).
- __str__ method provides a string representation like "Ace of Hearts".

Hand Class:

- Represents a player's or dealer's collection of cards.
- Tracks the total value of cards (total_value) and whether the hand contains an Ace (contains_ace), as Aces can have two values (1 or 11).
- add_card() adds a card to the hand and updates the total value.
- calculate_value() computes the best value of the hand, considering the special rule for Aces.

Deck Class:

- Represents the deck of cards.
- Initializes a standard 52-card deck.
- shuffle() randomizes the card order.
- deal_one() removes and returns the top card from the deck.

2. Game Flow: Core Functions

play_game() (Main function):

- Handles the overall flow of the game.
- Starts with a certain number of chips (starting_chips).
- Loops through rounds of the game until the player runs out of chips or decides to quit.

place_bet(chips):

- Asks the player how much they want to bet for the round.
- Ensures the bet is valid and within the player's chip balance.

deal_initial_cards(deck, player_hand, dealer_hand):

- Deals two cards to both the player and dealer at the start of the round.

display_hands(player_hand, dealer_hand, show_dealer):

- Displays the player's and dealer's hands. If show_dealer is False, the dealer's first card is hidden.

hit_or_stand(deck, player_hand):

- Asks the player whether they want to "hit" (take another card) or "stand" (end their turn).
- If the player chooses to hit and their total exceeds 21, they "bust," ending the round.

2. Game Flow: Core Functions

dealer_turn(deck, dealer_hand):

- The dealer draws cards until their hand value is at least 17.
- This follows the standard Blackjack rule that the dealer must stand on 17 or higher.

check_winner(player_hand, dealer_hand, chips):

- Determines the winner by comparing the player's and dealer's hand values:
  - Player busts → Dealer wins.
  - Dealer busts → Player wins.
  - Player's value higher than dealer's → Player wins.
  - Equal values → Tie (push).
- Adjusts the player's chip balance accordingly.

game_active:

- A global flag that tracks whether the round is active or over.

3. Core Blackjack Rules Implemented

Player Actions:

- The player can "hit" to take more cards or "stand" to end their turn.
- The goal is to have a hand value close to 21 without exceeding it.

Dealer Rules:

- The dealer always plays after the player finishes their turn.
- The dealer must draw cards until their hand value is at least 17.
- If the dealer's value exceeds 21, they bust, and the player automatically wins.

Ace Handling:

- Aces are treated as either 1 or 11, depending on what benefits the hand most:
  - If adding 10 (making Ace = 11) keeps the hand value $\leq$ 21, the Ace counts as 11.
  - Otherwise, the Ace counts as 1.

4. Game Flow: Round Logic

Start a New Round:

- The deck is shuffled, and both the player and dealer receive two cards.

Player's Turn:

- The player decides whether to "hit" or "stand."
- If the player busts (hand value > 21), the round ends immediately.

Dealer's Turn:

- The dealer automatically draws cards until their hand value is at least 17.

Compare Hands:

- If neither the player nor dealer busts, their hand values are compared to determine the winner.

Adjust Chips:

- The player's chip balance is updated based on whether they win, lose, or tie the round.

Play Again:

- The player can choose to play another round if they still have chips.

5. Terminating the Game

The game ends if:

- The player runs out of chips.
- The player chooses not to play another round.

6. Key Features

Randomness:

- The deck is shuffled randomly before each round.

Modular Code:

- Classes and functions are modular, making the code easy to extend or modify.

User Interaction:

- Input prompts guide the player through betting, choosing actions, and playing additional rounds.

Realistic Rules:

- Follows standard Blackjack rules for Aces, dealer behavior, and hand comparisons.