Nama: Moh.Rangga Wijaya Fadilah
NPM: 227006416144
Mata Kuliah:

Janken Game (Suit Jepang / Rock-Paper-Scissors)

Project Overview

This Python OOP project simulates a two-player console game of Janken (the Japanese version of Rock-Paper-Scissors) using object-oriented programming. The game involves a **human player** and a **computer player**, both starting with a **counter value of 10**. The goal is to reduce a player's counter to 0 before the opponent does. The player who first reaches **counter = 0** wins.

General Requirements

- The game consists of **two players**:
    - One **human** (user input)
    - One **computer** (random move generation)
- **Each player starts with a counter of 10**.
- **Each round**:
    - Both players choose a move: rock, paper, or scissors.
    - Moves are compared using the traditional rules:
        - Rock beats Scissors
        - Scissors beats Paper
        - Paper beats Rock
    - The **winner's counter is reduced by 1**.
    - The **loser's counter is increased by 1**.
    - If the round is a **tie**, no counter changes.
- The game **continues** until **one player's counter reaches 0**.
- The **first player to reach counter = 0 is declared the winner**.

Specific Requirements

1. Interactivity

- The human player must **press Enter** before each round starts.
- The human is **prompted to enter** one of the following moves:
    - "rock"
    - "paper"
    - "scissors"
- Input must be **validated** to prevent invalid choices.

2. Play

- The **computer player randomly** selects its move.
- The game **compares both moves** and determines the winner using conditional logic.
- The **counters are updated** based on the round result.

3. Messaging

- The game prints:
    - A **welcome message** when the game starts.
    - The **round number**.
    - The **moves** selected by both players.
    - The **result of the round** (win, lose, or tie).
    - The **updated counters** after each round.
    - A **game over message** declaring the winner.

Class Descriptions

Class: Move

Encapsulates the logic of selecting and holding a move.

- **Attributes**:
    - choice: the current selected move (rock, paper, or scissors).
- **Methods**:
    - set_choice(choice): Sets the move if it's valid.

    - random_choice(): Assigns a random valid move (for computer).
    - get_choice(): Returns the current move.

Class: Player

Represents a player, either human or computer.

- **Attributes**:
    - is_human: Boolean indicating whether the player is human.
    - move: An instance of the Move class.
    - counter: Integer initialized at 10 to track score.
- **Methods**:
    - make_move(): Gets user input or generates random move.
    - increment_counter(): Increases counter by 1.
    - decrement_counter(): Decreases counter by 1.
    - get_move(): Returns the current move of the player.

Class: JankenGame

Controls the flow of the game and holds references to both players.

- **Attributes**:
    - human: A Player instance for the human.
    - computer: A Player instance for the computer.
    - round: Tracks the number of rounds played.
- **Methods**:
    - play(): The main game loop.
    - play_round(): Handles one round of the game: move selection, comparison, result, and counter update.
    - determine_winner(human_choice, computer_choice): Returns the winner of a round.

Game Flow

1. The game displays a welcome message.
2. Each round starts when the player presses Enter.
3. The human selects a move.
4. The computer selects a random move.
5. Both moves are compared and a winner is decided.
6. Counters are updated:
    - Winner: counter -1
    - Loser: counter +1
    - Tie: counters unchanged
7. The game continues until a player's counter reaches 0.
8. A final message is shown to declare the winner.