Title: Blackjack Game Simulation: An Object-Oriented Approach

Ron Cox

Johns Hopkins University

Java 605.201.83

Assignment 9 Mini Project #2

Instructor: Dr. Sidney Rubey

Date: 4/1/2024

Abstract

This document outlines the design and functionality of a simple Blackjack game simulation developed in Java. The program utilizes core principles of object-oriented programming to model the game elements and interactions. A brief overview of the classes, methods, and game flow is provided, followed by a conceptual UML diagram to illustrate the relationships between the objects.

Introduction

Blackjack, also known as 21, is a popular card game where the goal is to have a hand value closer to 21 than the dealer's hand without exceeding 21. This simulation provides a basic interactive command-line version of the game, where a single player competes against a digital dealer. The Java programming language was chosen for its robust handling of object-oriented concepts, such as encapsulation, inheritance, and polymorphism, though the latter two are minimally employed in this particular application for simplicity.

Program Design

The program is structured into several classes, each responsible for a distinct aspect of the game:

- **Deck**: Represents a deck of cards. Responsible for shuffling and dealing cards.
- **Card**: Represents a single playing card, holding a suit and a value.
- **Hand**: Represents a hand of cards for either the player or the dealer. Capable of adding cards, calculating the total hand value, and determining if the hand is a blackjack or bust.
- **Player**: Represents the game's player. Manages the player's balance and betting.
- **BlackjackGame**: The main game controller. Handles the gameplay logic, user input, and maintains the game state.

Methods and Game Flow

The game starts by initializing the player's balance and dealing two cards each to the player and the dealer. The player then decides whether to "hit" (take another card) or "stand" (hold their current hand), aiming to achieve a hand value as close to 21 as possible without busting (exceeding 21). The dealer follows a simple set of rules, hitting on hands worth less than 17 and standing otherwise.

The winner is determined after both the player and the dealer have finished playing their hands. The game updates the player's balance based on the outcome of the round and asks whether the player wishes to continue. The game ends when the player chooses to stop or when the player's balance is depleted.

UML Diagram

The UML diagram below conceptualizes the relationships between the classes:

```
+----------------+     +--------------+     +-----------+
| BlackjackGame  |---->|    Player    |     |    Deck   |
+----------------+     +--------------+     +-----------+
       |                      ^                    |
       |                      |                    |
       V                      |                    V
+------------+    uses    +--------+    contains +--------+
|    Hand    |<-----------|  Card  |<-----------|  Card   |
+------------+            +--------+            +--------+
```

Conclusion

This Blackjack game simulation demonstrates a practical application of object-oriented programming principles in Java to create a simple yet interactive game. Through encapsulation, the program efficiently separates concerns, allowing for easier maintenance and potential scalability. Future enhancements could introduce more complex features such as multiple players, betting strategies, and a graphical user interface.

References

1. Deitel, P. J., & Deitel, H. M. (2017). *Java: How to Program, Early Objects* (11th ed.). Pearson Education.
2. Sierra, K., & Bates, B. (2005). *Head First Java* (2nd ed.). O'Reilly Media.