INFO1113 Citadels Game Report

Overview

This project implements a single hunman-player version of the Citadels card game using object-oriented principles in Java. The core design aims to promote encapsulation, modularity, and extensibility, ensuring the code is maintainable and adaptable to future features.

Implement of Extension

Compared to traditional card games, the online version allows game progress to be saved and loaded at any time. Firstly, all standard input and output operations are integrated into the `Super` class for easier and centralized control. Then, this project adds caching and logging function, which can record the operation from human player. When loading data, it can exactly come back to saving position by re-running game. For future features, it will not affect the implementation of game saving and loading, and this also reduces the risk of potential crashes. In addition, the functions of `getInput` and `getInput_withoutcommand` in the `App` class, integrate all player input, as well as the strict player command format control is enforced through function parameters.

Class Hierarchy

The main interactive `App` class and the `Player` class, which handles user information, both inherit from the `Super` class that contains all static I/O as well as save and load functionalities. Encapsulating standard input/output and modularized user command prompts reduces code duplication and enforces consistent input standards.

Main Implement of Character Cards and Purple District Cards

The project uses `enum` to encapsulate character cards and district cards into two separate enums, with each card represented as an enum constant.

Character Cards: The `GameCharacter` enum models the characters, each with distinct abilities and gameplay mechanics. This class effectively encapsulates character-specific logic, including unique resource behaviors and actions during a player's turn.

Each **GameCharacter** instance contains overridden methods to define special behavior (`CharacterAbility`, `ExtraResource`). Enum constants override the same

method signatures to provide character-specific behaviors. This allows dynamic dispatch based on the character instance. This also avoids an excessive class hierarchy and keeps all characters in a single, cohesive structure.

Purple District Cards: Skill is enum that encapsulates the logic of the special abilities of the "purple cards". Each Skill instance corresponds to a special area building (e.g. Laboratory, Armory, Museum, etc.) and defines its behavior such as building, destroying, and triggering skills.

Each skill implements its own behavior by overriding the `Special`, `OnBuilding`, `destroyed`, `extraScore`, and `extraResource` methods.

OOP Principle Application Analysis

For each enum constant, different methods can be overridden within an anonymous class to implement their respective unique logic. All abilities share interface methods under a unified enum structure, leveraging the object-oriented features built into enums to avoid creating numerous subclasses—effectively implementing a class hierarchy similar to a `SkillBase` or `GameCharacterBase`. For example, when we need to add a Queen, we simply add a new enumeration constant and override `ExtraResource` (similar to the merchant implementation) without modifying the existing class structure at all.

Conclusion

This design allows to add new character or district cards through enum constants and overriding methods, thereby supporting future extensibility.