Software Design Fundamentals - Term Project

Project Designation: .magic (dot magic)

Project Variables:

Card() - Details regular cards and their interactions with the point system
CardSpecial() - Details the special cards and their unique interactions with both themselves, other special cards and normal cards/gameplay
CardGroups() - Each color represented in the game.
CardHand() - The "hand" distributed to each player at the start of the game
Player() - The player object (instantiated multiple times for 2-4 players)

*** I think the trickiest thing about this project is ensuring the stack properly vindicates priority.***

Language Chosen: Java
Base Code: Some methods/functions utilized from START_CODE, given via SLATE

Game Rules:
Be the first player to 20 points! Use the specialized cards to steal points and other special cards from other players to ensure your victory, or, utilize special cards to drown your opponent

Each card played is worth 1 points with special cards worth their face value. (game style can be a reflection of uno's typical game play -> i.e. matching faces to colors)

Cards and card pile can be shuffled to and from the game field as per special card rules.

Some special cards are "fast" while others are "slow" this means that only fast cards gain priority on the "stack" while "slow" special cards can only be played on the player's turn.

If a fast card is "negated" the effect does nothing, if a negated special is then negated the special effect takes places.

I.e. Your turn begins -> Draw card -> You play "Steal Special" -> Opponent plays "Refuse" targeting your "Steal Special" -> You play "Refuse" targeting the opponent's "Refuse" -> Initial "Steal Special" triggers -> Steal a card -> End of Turn

Special Cards:

(fast) Steal Special - Similar to Thought Erasure from MTG, this card allows a player to select an opponent and steal any card they want from their hand, whether it is a special card or a normal card.

(fast) Draw (face value) Special - Similar to Uno’s draw system, this card when played, forces target player of choice to draw that many cards. Can be seriously over-powered if face value is 5+.

(fast) Redirect Special - Special card that “deflects” another special cards abilities to another opponent at random -> if you were targeted with a draw card, this card would redirect the same effect to the opponent to your left. If 2 players are playing, it redirects the ability back to the “caster” of the special ability.

I.e. Opponent turn begins -> Opponent uses “Draw (2) on you” -> You play Redirect card played -> Opponent draw 2 cards -> Your turn begins

(slow) Refuse Special - Special card works as a “negate” function to cancel the effects of any special card used against the player. Slow speed ONLY.

(slow) Redraw Special - Special cards as an extra draw card, can be useful if a player does not have a desired card and wish to take a chance at drawing a particular card they may need i.e. perhaps a Draw(face) card lies at the top of your card library….

TYLER W. BELAIR
STUDENT ID# - 991561958
GROUP MEMBERS - SELF

SOFTWARE DESIGN FUNDAMENTALS - PROJECT 1

PROJECT DESCRIPTION

.magic (dot magic) is modeled after the exceptionally famous existing trading card game Magic: The Gathering. However this game will not entail the same level of mechanics and card interplay. This game will be a “short tutorial” version implemented through the use of Intellij with a few special cards to enact some interaction between player(s) and/or computer simulated player as well as provide acceptable wincon/losecon’s to make it interesting to play.

PROJECT SCOPE

The project scope will allow for sustainability, creative gameplay interpretations as well as a strict guidelines towards standardised coding practices. To accomplish these goals the group member has chosen Java to utilize its vast libraries and applets accessible to everyone with an appropriate IDE. The project itself will be complete once the system enables a player to achieve a win/lose condition against another player and/or another computer.

GITHUB REPOSITORY URL:

https://github.com/Yldraziw/Sheridan_Code.git (please request service should the link not work!)

CODING STANDARDS

This project will utilize the IDE IntelliJ to operate, play, debug errors and create enticing interplay between player and game system. IntelliJ is chosen for its in-depth libraries, modules and accessible utilities as well as conforming to a strict encapsulated class-based program.

BASECODE WRITE-UP

Encapsulation - Each module of this game will include as many mutators and accessors as required to provide the most strict Data encapsulation. I would not want my player (or computer) to suddenly gain a wincon advantage due to faulty coding logic!

Delegation - Each module of this game will also house 1-2 specific rule sets and no other module will be given a “say” in what each other module does. SImilarly the total function of each module will be sent to the “game stack” which will be separately controlled by a “stack engine” to determine the best course of action regarding game play, card play and rule modifications.

POTENTIAL PROBLEMS/TECHNICAL DIFFICULTIES

To my knowledge the entirety of this “game” will take place in a module called “STACK_ENGINE” this module will comprise the “turn base” that will be implemented in this game. The unfortunate part about this is that I am relying on a particular “clock” value during interplay and there may arise difficulties in executing this “clock” without errors.