Hearthstone Lite

Program Classes

* Player
* User
* AI
* Field
* Deck
* Active Cards
* Hand Card
* Player
  + The Player Class is a Super class of User and AI. This class creates an inheritance relationship with both of those classes. The class is used to generalize both the User and AI classes into a similar framework to be used throughout the program. The Player Class takes the given information of the User and AI classes, and stores it within similar variables to be passed into the gameplay phase of the program based within the Field class.
  + Attributes
    - ID
      * Datatype: Integer (int)
      * Private
      * Specifies the unique identifier of each player that is given to them.
    - Name
      * Datatype: String
      * Private
      * The designated name or title of the player.
    - Decks
      * Datatype: list(Deck)
      * Private
      * Contains the list of custom decks stored for a player.
    - HP
      * Datatype: Integer (int)
      * Private
      * The Health Points of the player.
    - Mana
      * Datatype: Integer (int)
      * Private
      * The Magic Points of a player.
    - chosenDeck
      * Datatype: Deck
      * Private
      * Stores the Deck chosen to be passed into the gameplay phase.
  + Methods
    - newPlayer(int)
      * Protected
      * Creates a new player object using a provided ID from either the User or AI object.
    - selectedDeck()
      * Public
      * Returns the Deck selected by the player in the pre-game screen. It is used to pass the chosen Deck variable to the Field class.
* User
  + The subclass of Player, it inherits Player. This class creates a User object to be passed into the program.
  + Attributes
    - ID
      * Datatype: Integer (int)
      * Private
      * Specifies the unique identifier of each user.
    - Name
      * Datatype: String
      * Private
      * The chosen name of the user.
    - Decks
      * Datatype: list(Deck)
      * Stores the custom decks of the user.
  + Methods
    - newUser()
      * Private
      * Generates a unique integer to pass into the ID variable.
    - viewDeck(int)
      * Private
      * Returns a Deck at the index from the user’s list of decks.
* AI
  + The subclass of Player, inherits Player. This class creates an AI object to be passed into the program.
  + Attributes
    - ID
      * Datatype: Integer (int)
      * Private
      * Specifies the unique identifier of the AI.
    - Name
      * Datatype: String
      * Private
      * The default name of the AI
    - Decks
      * Datatype: list(Deck)
      * Stores the default decks of the AI.
  + Methods
    - newAI()
      * Private
      * Generates a unique integer to pass into the ID variable.
    - viewDeck(int)
      * Private
      * Returns a Deck at the index from the AI’s list of decks.
* Field
  + The Field class provides a platform for the User and AI players to play against each other. This class represents the interface of the playing field. The Field takes player information as input variables and communicates with the Hand and ActiveCards classes.
  + Attributes
    - Sides
      * Datatype: list(Player)
      * Private
      * Contains a list of the two opposing Player objects
    - PlayerSide
      * Datatype: ActiveCards
      * Private
      * The container for the ActiveCards object of the Player
    - AISide
      * Datatype: ActiveCards
      * Private
      * The container for the ActiveCards object of the AI
  + Methods
    - DefeatPlayer(Player)
      * Private
      * Applies the Defeat variable to the passed player object by choice of a Boolean variable.
    - WinPlayer(Player)
      * Private
      * Applies the Win variable to the passed player object by choice of a Boolean variable.
* Deck
  + The Deck class contains Card objects stored by default or chosen by the player.
  + Attributes
    - ID
      * Datatype: Integer (int)
      * Private
      * Unique identifier for the Deck, this is primarily used for utilitarian purposes.
    - Name
      * Datatype: String (int)
      * Private
      * The chosen name of the Deck, either by default or created by the player.
    - Size
      * Datatype: Integer (int)
      * Private
      * A stored variable of list.size() used in method calls.
    - listCards
      * Datatype: list(Card)
      * Private
      * The container for the Deck that stores the selected Cards
  + Methods
    - createDeck(int,String)
      * Protected
      * This method is called to create a new Deck object
    - editDeck(int)
      * Private
      * This method is called to edit an existing Deck object
    - removeDeck(int)
      * Private
      * Removes the Deck object at the selected index
    - returnSize()
      * Public
      * Returns the size of the Deck
    - returnName()
      * Public
      * Returns the name of the Deck
    - viewDeck()
      * Public
      * Returns a list(Card) object copied from the Deck
    - addCard(Card)
      * Protected
      * Adds the passed Card object to the list
    - removeCard(int)
      * Protected
      * Removes the Card object at the index and returns it.
* ActiveCards
  + The ActiveCards class contains the current Cards contained ‘on the field’ for either player.
  + Attributes
    - listCards
      * Datatype: list(Card)
      * Private
      * Contains the list of active cards for that player
    - Owner
      * Datatype: Integer (int)
      * Private
      * Stores the passed value of the player ID
    - Size
      * Datatype: Integer (int)
      * Private
      * A stored variable of list.size()
  + Methods
    - viewActive()
      * Protected
      * Returns a list of Cards in the object
    - addCard(Card)
      * Protected
      * Adds a Card object to the list
    - removeCard(int)
      * Protected
      * Removes a Card object from the passed index value
* Hand
  + The Hand class contains a list of Cards that only the player can see in detail. The player interacts with this class to activate Cards that are passed onto the ActiveCards class.
  + Attributes
    - listCards
      * Datatype: list(Card)
      * Private
      * Contains the list of Cards stored in the Hand
    - Owner
      * Datatype: Integer (int)
      * Private
      * Stores the passed value of the player ID
    - Size
      * Datatype: Integer (int)
      * Private
      * A stored variable of list.size()
  + Methods
    - viewHand()
      * Protected
      * Returns a list(Card) object copied from the list
    - addCard(Card)
      * Protected
      * Adds a Card object from the Deck into the list
    - removeCard(int)
      * Protected
      * Removes a Card from the list at the given index value and returns a Card object
* Card
  + This class exists to portray the actual Cards played within the game. It is normally used to store its objects within Decks.
  + Attributes
    - ID
      * Datatype: Integer (int)
      * Private
      * The unique identifier of the Card
    - Name
      * Datatype: String
      * Private
      * The name of the Card
    - HP
      * Datatype: Integer (int)
      * Private
      * The Health Points of the Card
    - Attack
      * Datatype: Integer (int)
      * Private
      * The attack power of the Card
    - Defense
      * Datatype: Integer (int)
      * Private
      * The defense power of the Card
  + Methods
    - returnID()
      * Public
      * Returns the ID of the Card
    - returnName()
      * Public
      * Returns the name of the Card
    - Health()
      * Public
      * Returns the HP value of the Card
    - Attack()
      * Public
      * Returns the attack value of the Card
    - Defense()
      * Public
      * Returns the defense value of the Card

**RELATIONSHIPS BETWEEN CLASSES**:

* **Inheritance (Generalization)** relationship is present between Player, User, AI and Field classes.
  + **Super Class:** Player
  + **Sub Classes:** User, AI and Field
* **Inheritance (Generalization)** relationship is present between Field, Deck and Hand classes.
  + **Super Class:** Field and Deck
  + **Sub Classes:** Hand
* **Inheritance (Generalization)** relationship is present between Field and Active Deck classes.
  + **Super Class:** Active Cards
  + **Sub Classes:** Field
* **Association** relationship is present between Hand and Active Cards classes.
* **Associate Composition** relationship is present between Deck, Active Cards, Hands and Card classes.
  + **Dependent classes:** Deck, Active Cards and Hand