```python
"""
22/11/2021 - Time:
Gabriel S.J. Spadoni
spadoni.gabriel-TASK39-DLV01-INC01-SPOP-PROG-v0.2
This product is the second version of the python program of the SPOP (Small Proprietary Original Project).
The SPOP is a simple one player game with fantasy elements. In this game, the user plays with a random
generated deck of 20 offensive and defensive cards. The player starts at the first floor of a dungeon and
will try to climb it. Each floor is constituted of one fight against an enemy. The enemy will attack and
defend itself against the player using cards. If the player is able to win each single fight, he wins a game.
The program contains the different classes (Card, Deck, CardUser, Player, Enemy) as well as the main function
that is responsible of the CLI used to interact with the software.
"""

import random as rd

# Class for cards, a card as a name, a type(offensive or defensive), health_effect, status_effect and the text which the decription of the card.
class ctCard:

    def __init__(self, name, type, health_effect, status_effect, text):
        self.name = name
        self.type = type
        self.health_effect = health_effect
        self.status_effect = status_effect
        self.text = text


    def oeDescribeCard(self):
        print(f' --- The card {self.name} is a {self.type}. --- \n --- {self.text} ---')

# Class for a deck, a deck is composed of offensive_cards, defensive_cards and has a max card.
# It also contains the deck which is a list and isShuffled.
# There are the functions createDeck, shuffleDeck and printDeck
class ctDeck:

    def __init__(self, offensive_cards, defensive_cards, max_cards=20):
        self.offensive_cards = offensive_cards
        self.defensive_cards = defensive_cards
        self.deck = []
        self.total_deck = []
        self.max_cards = max_cards
        self.isShuffled = False

    def oeCreateDeck(self):

        self.isShuffled = False

        n_offensive_cards = rd.randint(8, 12)
        n_defensive_cards = self.max_cards - n_offensive_cards

        for _ in range(n_offensive_cards):
            card = self.offensive_cards[rd.randint(0, len(self.offensive_cards) - 1)]
            self.deck.append(card)
        for _ in range(n_defensive_cards):
            card = self.defensive_cards[rd.randint(0, len(self.defensive_cards) - 1)]
            self.deck.append(card)

        self.total_deck = self.deck

    def oeShuffleDeck(self):
        self.isShuffled = True
        rd.shuffle(self.deck)


    def oePrintDeck(self):
        n = 1
        for card in self.deck:
            print(n, card.name)
            n += 1

    def oeShowDeck(self):
        for card in self.deck:
            print(card.name)


# Class CardUser is the class for player and enemies. A card user has a hand, a deck, health_points and a status.
# There are the functions showStatus which shows the health and the status and useCard that has
class ctCardUser:

    def __init__(self, health_points):
        self.hand = []
        self.deck = []
        self.health_points = health_points
        self.status_effect = None
        self.isDead = False

    def oeShowStatus(self):
        pass

    def ugUseCard(self, card, target):

        if card not in self.hand:
            return "--- Not in hand ---"

        if self.status_effect == 'freeze':
            return "-- You are frozen ---"

        if target is None:
            return "--- No target Selected ---"

        self.oeApplyCardEffect(card, target)
        self.oeRemoveCardFromHand(card)

        return f"--- Card Effect applied --- \n --- {card.text} ---"

    def oeApplyCardEffect(self, card, target):
        health_effect = card.health_effect
        status_effect = card.status_effect

        target.health_points += health_effect
        target.status_effect = status_effect

    def oeRemoveCardFromHand(self, card):
        self.hand.remove(card)

    def oeDrawCard(self, number):

        for _ in range(number):
            self.hand.append(self.deck.deck[0])
            self.deck.deck = self.deck.deck[1:]

    def oeShowHand(self):

        for card in self.hand:
            print(f"--- {card.name} ---")

    def oeShowDeck(self):

        self.deck.showDeck()

# Class Player inherits from CardUser, however here with have a set health_points value
class ctPlayer(ctCardUser):

    def __init__(self, tries):
        super().__init__(health_points= 20)
        self.isFightingEnemy = False
        self.tries = tries

    def oeGiveDeck(self, deck):
        self.deck = deck




# Class Enemy inherist from CardUser
class ctEnemy(ctCardUser):

    def __init__(self, health_points):
        super().__init__(health_points)
        self.isFightingPlayer = False



class ctAdmin:
    pass


# Class Game will take care of the interactions of the user.
class Game:

    def __init__(self):

        self.player = None
        self.enemies = []
        self.current_floor = 0
        self.turn_counter = 0

        self.quit = False
        self.isDeckCreated = False
        self.isDeckShuffle = False
        self.isGameStarted = False
        self.isFightStarted = False
        self.isPlayerCreated = False
        self.isTypeUser = False
        #self.commands = dict()

        self.offensive_cards_dic = {
            0: ctCard("FireBall", "offensive", -5, "burn", "Inflict 5 damages to the targeted entity"),
            1: ctCard("Ice Spike", "offensive", -3, "freeze", "Inflict 3 damage to targeted entity"),
            2: ctCard("Burn", "offensive", 0, "burn", "Inflict burning effect to the target")
        }

        self.defensive_cards_dic = {
            0: ctCard("Heal", "defensive", +5, None, "Heal 5 health points to target"),
            1: ctCard("Block", "defensive", 6, "prevent", "Prevent next turn damage"),
            2: ctCard("Blessing", "defensive", 0, "bless", "Next healing will be doubled")
        }


    # Add a command manager with commands dic. To have a single function that manages
    # Main process of the CLI.
    def main(self):


        print("--- Welcome in the SPOP of Gabriel S.J. Spadoni ---")
        print("--- Type help to display the available commands and their description ---")
        print("--- Enter your user type ---")

        while not self.isTypeUser:
            command = input("--- Type your command --- ")

            if command == 'player':
                player = ctPlayer(tries= 3)
                self.isPlayerCreated = True
                self.isTypeUser = True
            elif command == 'admin':
                print("---  admin not supported --- ")
            else:
                print('--- Enter your user type: player or admin ---')

        while not self.quit:

            command = input("--- Type your command --- ")

            self.commandHandler(command, player)

    #Handles the different command that the user can enter in the terminal
    def commandHandler(self, command, player):
        command = command.split(' ')

        if command[0] == 'quit':
            self.quit = True
        elif command[0] == 'help':
            print("--- Commands are: ---")
            print("--- createDeck: creates a deck of random cards taken from the card pool ---")
            print("--- showDeck: shows the cards in the deck --- ")
            print("--- shuffleDeck: shuffles the deck --- ")
            print("--- startGame: starts the game ---")
            print("--- startFight: starts a fight agains the enemy of the floor --- ")
            print("--- showHand: shows the cards in hand ---")
            print("--- describeCard cardname: describes the effect of the card ---")
            print("--- useCard cardname target: applies the card effect to the targeted card user ---")
            print("--- quit: quit the software --- ")
        elif command[0] == 'createDeck':
            deck = ctDeck(self.offensive_cards_dic, self.defensive_cards_dic)
            deck.oeCreateDeck()
            player.oeGiveDeck(deck)
            print("--- Deck Created! --- ")
            self.isDeckCreated = True
        elif command[0] == 'showDeck':
            if self.isDeckCreated:
                print("--- Deck contains --- ")
                player.deck.oePrintDeck()
            else:
                print('--- Deck not created! --- ')
        elif command[0] == 'shuffleDeck':
            if self.isDeckCreated:
                player.deck.oeShuffleDeck()
                self.isDeckShuffled = True
                print('--- Deck succesfuly shuffled! --- ')
            else:
                print('--- Deck not created! --- ')
        elif command[0] == 'startGame':
            if self.isDeckCreated and self.isDeckShuffled and self.isPlayerCreated and not self.isGameStarted:
                self.createEnemies()

                print('--- The game has started! --- ')
                self.isGameStarted = True
            else:
                print("--- Game can't be started ---")
        elif command[0] == 'describeCard':
            try:
                card_name = command[1]

                for card in player.deck.total_deck:
                    if card.name == card_name:
                        card_used = card
                    else:
                        card_used = None


                if card_used is not None:

                    card.oeDescribeCard()
                else:
                    print("--- Card not in deck ---")


            except:
                print("--- no card given ---")
        elif command[0] == 'startFight':
            if self.isDeckCreated and self.isDeckShuffled and self.isPlayerCreated and self.isGameStarted and not self.isFightStarted:

                self.isFightStarted = True
                print("--- Fight Started ! ---")
                self.startFight(player, self.enemies[self.current_floor])


        elif command[0] == 'useCard':
            if self.isDeckCreated and self.isDeckShuffled and self.isPlayerCreated and self.isGameStarted and self.isFightStarted:
                card_name = command[1]
                target_name = command[2]

                for card in player.deck.total_deck:
                    if card.name == card_name:
                        card_used = card
                    else:
                        card_used = None

                if target_name == 'enemy':
                    target = self.current_enemy
                elif target_name == 'player':
                    target = player
                else:
                    target = None

                message = player.ugUseCard(card_used, target)
                print(message)
        elif command[0] == 'showHand':
            if self.isDeckCreated and self.isDeckShuffled and self.isPlayerCreated and self.isGameStarted and self.isFightStarted:
                print("--- The cards in your hand are: ---")
                player.oeShowHand()
        else:
            print("--- Unknown Command. Try again! --- ")


    def createEnemies(self):

        enemy0 = ctEnemy(10)
        self.enemies.append(enemy0)

    #Handles a fight against an enemy
    def startFight(self, player, enemy):
        player.isFightingEnemy = True
        enemy.isFightingPlayer = True

        player.oeDrawCard(5)
        self.current_enemy = enemy

        while player.isFightingEnemy and enemy.isFightingPlayer and not self.quit:

            command = input("--- Type your command --- ")
            self.commandHandler(command, player)

            if player.health_points <= 0 or enemy.health_points <= 0:
                player.isFightingEnemy = False
                enemy.isFightingPlayer = False



game = Game()

game.main()
```