```python
"""
8/11/2021 - Time:
Gabriel S.J. Spadoni
spadoni.gabriel-TASK31-DLV01-INC01-SPOP-PROG-v0.1
This product is the first 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 Card:

    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 describeCard(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 Deck:

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

    def createDeck(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)]
```

```
51              self.deck.append(card)
52          for card in range(n_defensive_cards):
53              card = self.defensive_cards[rd.randint(0, len(self.defensive_cards) -
    1)]
54              self.deck.append(card)
55
56      def shuffleDeck(self):
57          self.isShuffled = True
58          rd.shuffle(self.deck)
59
60
61      def printDeck(self):
62          n = 1
63          for card in self.deck:
64              print(n, card.name)
65              n += 1
66
67      def showDeck(self):
68          pass
69
70  # Class CardUser is the class for player and enemies. A card user has a hand, a
    deck, health_points and a status.
71  # There are the functions showStatus which shows the health and the status and
    useCard that has
72  class CardUser:
73
74      def __init__(self):
75          self.hand = []
76          self.deck = []
77          self.health_points = 20
78
79      def showStatus(self):
80          pass
81
82      def useCard(self, card, target):
83          pass
84
85      def applyCardEffect(self, card, target):
86          pass
87
88      def removeCardFromHand(self, card):
89          pass
90
91
92  # Class Player inherits from CardUser, however here with have a set health_points
    value
93  class Player(CardUser):
94
95      def __init__(self, health_points = 20):
96          super().__init__()
97          self.health_points = health_points
98
99      def giveDeck(self, deck):
100         self.deck = deck
101
102     pass
103
104
105 # Class Enemy inherist from CardUser
106 class Enemy(CardUser):
```

```
107     pass
108 
109 
110 class Admin:
111     pass
112 
113 
114 # Class Game will take care of the interactions of the user.
115 class Game:
116 
117     def __init__(self):
118 
119         self.player = None
120         self.enemies = []
121         self.current_floor = 0
122         self.isGameStarted = False
123 
124         self.quit = False
125         self.isDeckCreated = False
126         self.isDeckShuffle = False
127         self.isGameStarted = False
128         self.isPlayerCreated = False
129         self.isTypeUser = False
130         self.commands = dict()
131 
132         self.offensive_cards_dic = {
133             0: Card("FireBall", "offensive", -5, "burn", "Inflict 5 damages to the
targeted entity"),
134             1: Card("Ice Spike", "offensive", -3, "freeze", "Inflict 3 damage to
targeted entity"),
135             2: Card("Burn", "offensive", 0, "burn", "Inflict burning effect to the
target")
136         }
137 
138         self.defensive_cards_dic = {
139             0: Card("Heal", "defensive", +5, None, "Heal 5 health points to
target"),
140             1: Card("Block", "defensive", 6, "prevent", "Prevent next turn damage")
141         }
142 
143 
144     # Add a command manager with commands dic. To have a single function that
manages
145     # Main process of the CLI.
146     def main(self):
147 
148 
149         print("--- Welcome in the SPOP of Gabriel S.J. Spadoni ---")
150         print("--- Type help to display the available commands and their description
---")
151         print("--- Enter your user type ---")
152 
153         while not self.isTypeUser:
154             command = input("--- Type your command --- ")
155 
156             if command == 'player':
157                 player = Player()
158                 self.isPlayerCreated = True
159                 self.isTypeUser = True
160             elif command == 'admin':
```

```
161                 print("---  admin not supported --- ")
162             else:
163                 print('--- Enter your user type: player or admin ---')
164 
165         while not self.quit:
166 
167             command = input("--- Type your command --- ")
168 
169             command = command.split(' ')
170 
171 
172             if command[0] == 'quit':
173                 self.quit = True
174             elif command[0] == 'help':
175                 print("--- Commands are: ---")
176                 print("--- createDeck: creates a deck of random cards taken from the
    card pool ---")
177                 print("--- showDeck: shows the cards in the deck --- ")
178                 print("--- shuffleDeck: shuffles the deck --- ")
179                 print("--- quit: quit the software --- ")
180             elif command[0] == 'createDeck':
181                 deck = Deck(self.offensive_cards_dic, self.defensive_cards_dic)
182                 deck.createDeck()
183                 print("--- Deck Created! --- ")
184                 self.isDeckCreated = True
185             elif command[0] == 'showDeck':
186                 if self.isDeckCreated:
187                     print("--- Deck contains --- ")
188                     deck.printDeck()
189                 else:
190                     print('--- Deck not created! --- ')
191             elif command[0] == 'shuffleDeck':
192                 if self.isDeckCreated:
193                     deck.shuffleDeck()
194                     self.isDeckShuffled = True
195                     print('--- Deck succesfuly shuffled! --- ')
196                 else:
197                     print('--- Deck not created! --- ')
198             elif command[0] == 'startGame':
199                 if self.isDeckCreated and self.isDeckShuffled and
    self.isPlayerCreated:
200                     player.giveDeck(deck)
201                     print('--- The game has started! --- ')
202                     self.isGameStarted = True
203                 else:
204                     print("--- Game can't be started ---")
205             else:
206                 print("--- Unknown Command. Try again! --- ")
207 
208 
209 
210 
211     def startFight(self, player, enemy):
212         pass
213 
214 
215 game = Game()
216 
217 game.main()
```