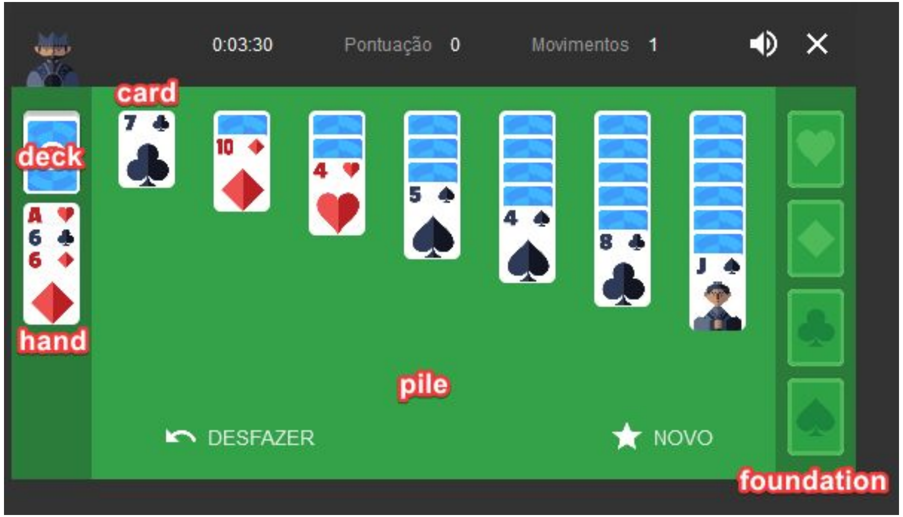
# Implementação do jogo Paciência em Python

João Gabriel Corrêa Krüger
Paulo José Machado Filho
Dupla Dinâmica

# Nomenclaturas e Implementações

- Pilha
    - Deck, Hand e Foundation
- Lista
    - Pile



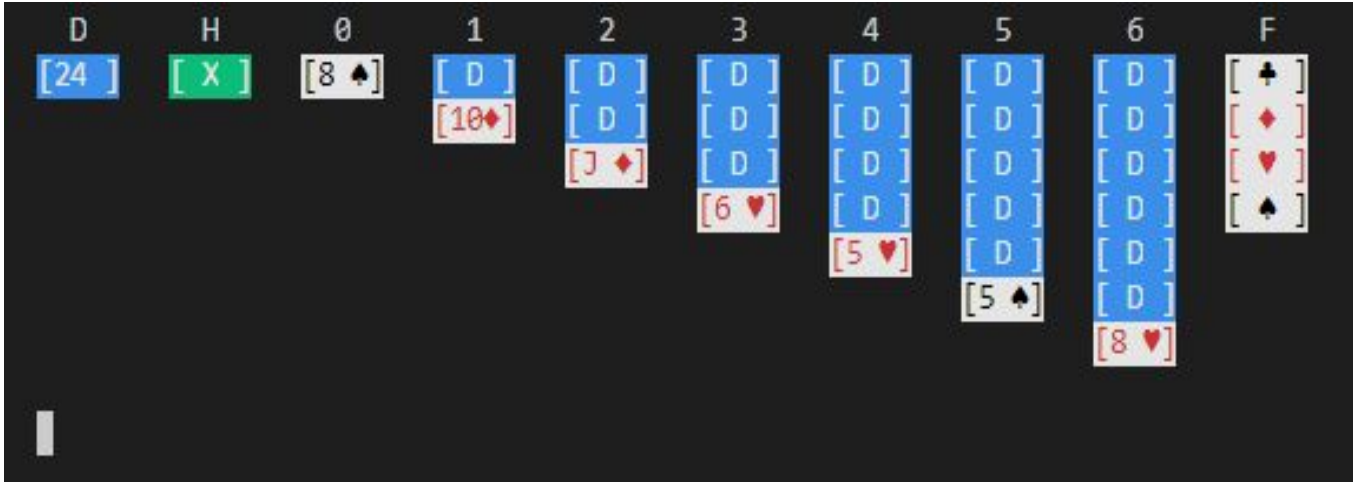
0:03:30

Pontuação 0

Movimentos 1

# Jogo

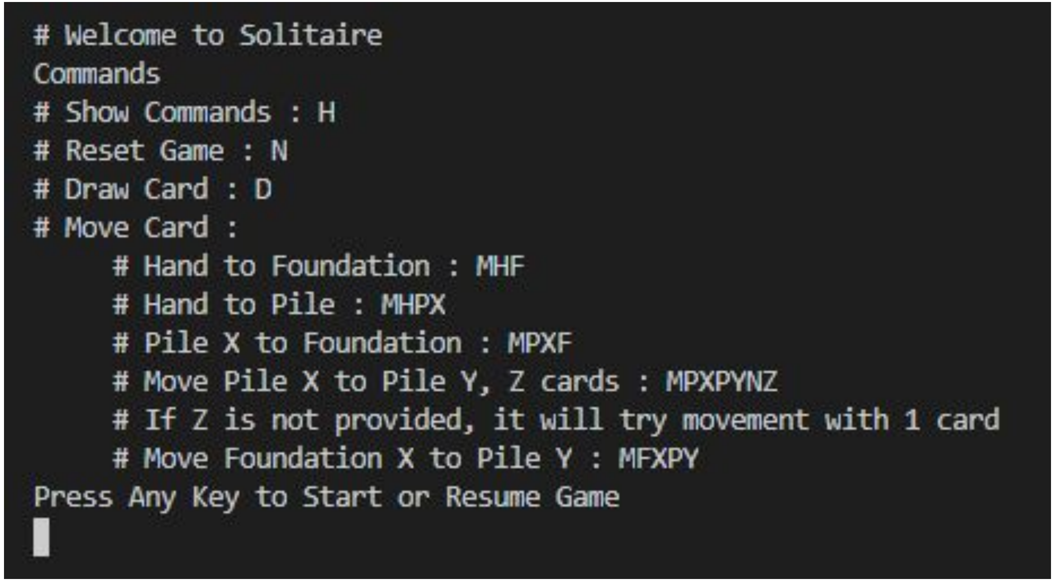
- Tela principal do jogo



```
  D      H      0      1      2      3      4      5      6      F
[24 ]  [ X ]  [8 ♠]  [ D ]  [ D ]  [ D ]  [ D ]  [ D ]  [ D ]  [ ♣ ]
                     [10♦]  [ D ]  [ D ]  [ D ]  [ D ]  [ D ]  [ ♦ ]
                            [J ♦]  [ D ]  [ D ]  [ D ]  [ D ]  [ ♥ ]
                                   [6 ♥]  [ D ]  [ D ]  [ D ]  [ ♠ ]
                                          [5 ♥]  [ D ]  [ D ]
                                                 [5 ♠]  [ D ]
                                                        [8 ♥]
```

# Manual

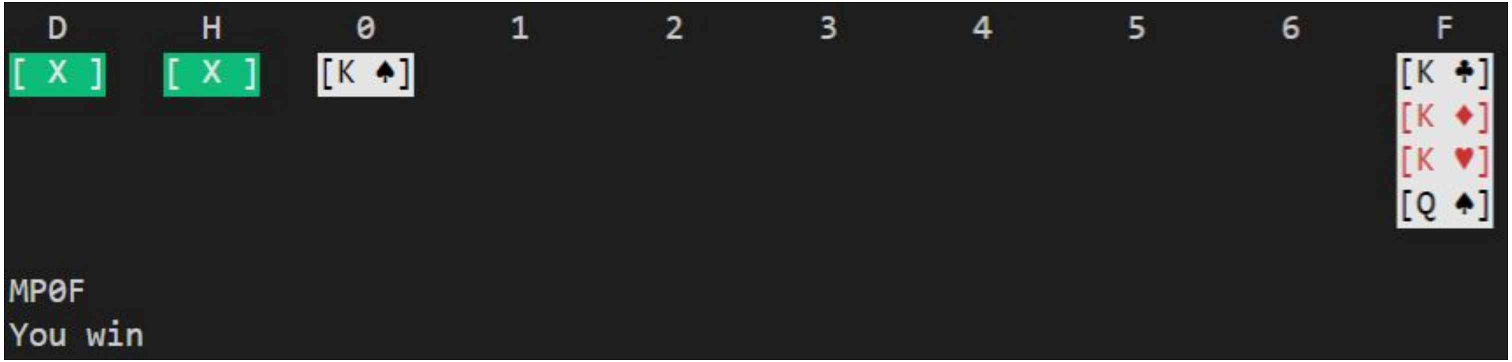
- Manual de comandos
  - Ao iniciar
  - Ao usar o comando ‘H’



```
# Welcome to Solitaire
Commands
# Show Commands : H
# Reset Game : N
# Draw Card : D
# Move Card :
        # Hand to Foundation : MHF
        # Hand to Pile : MHPX
        # Pile X to Foundation : MPXF
        # Move Pile X to Pile Y, Z cards : MPXPYNZ
        # If Z is not provided, it will try movement with 1 card
        # Move Foundation X to Pile Y : MFXPY
Press Any Key to Start or Resume Game
█
```

# Fim de jogo

- Funcional
  - Difícil
  - Sem a dança das cartas

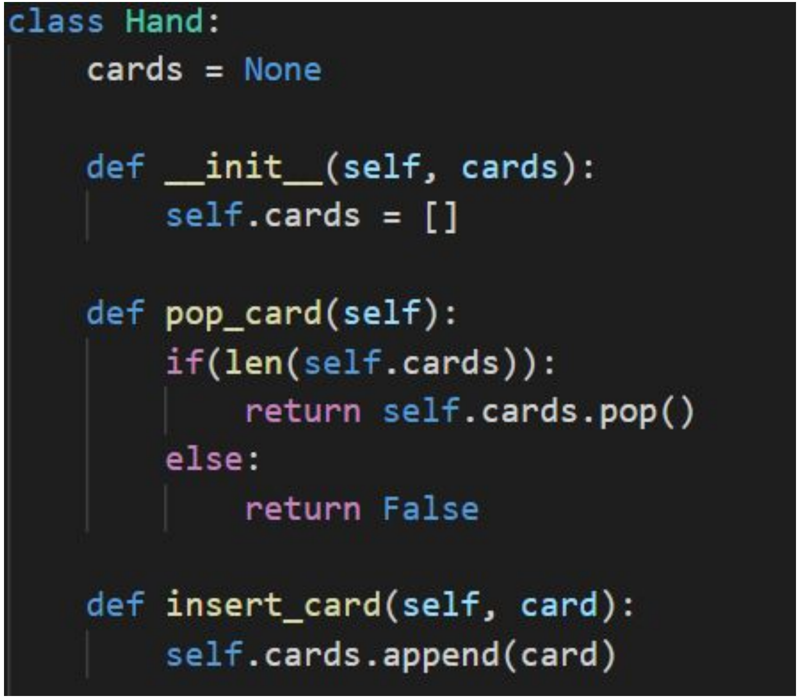


```
  D       H       0       1       2       3       4       5       6       F
[ X ]   [ X ]   [K ♠]                                                   [K ♣]
                                                                        [K ♦]
                                                                        [K ♥]
                                                                        [Q ♠]

MP0F
You win
```

# Implementações

# Hand - Stack



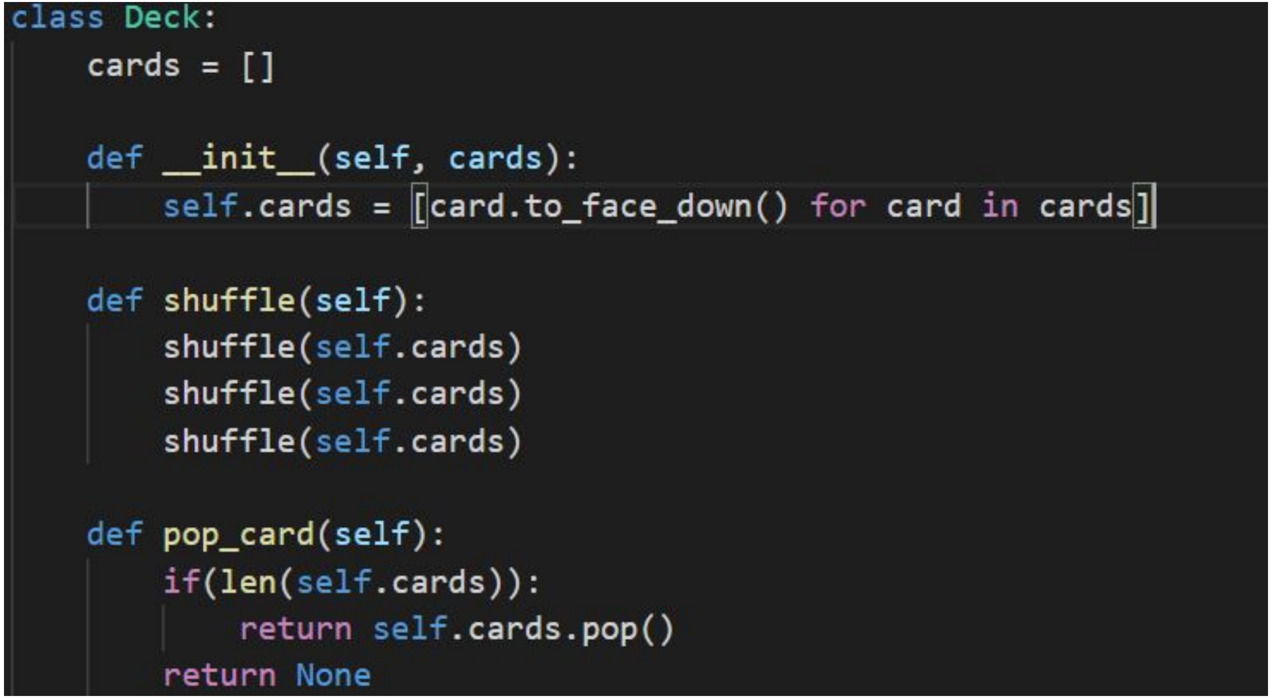
```python
class Hand:
    cards = None

    def __init__(self, cards):
        self.cards = []

    def pop_card(self):
        if(len(self.cards)):
            return self.cards.pop()
        else:
            return False

    def insert_card(self, card):
        self.cards.append(card)
```

# Deck - Stack



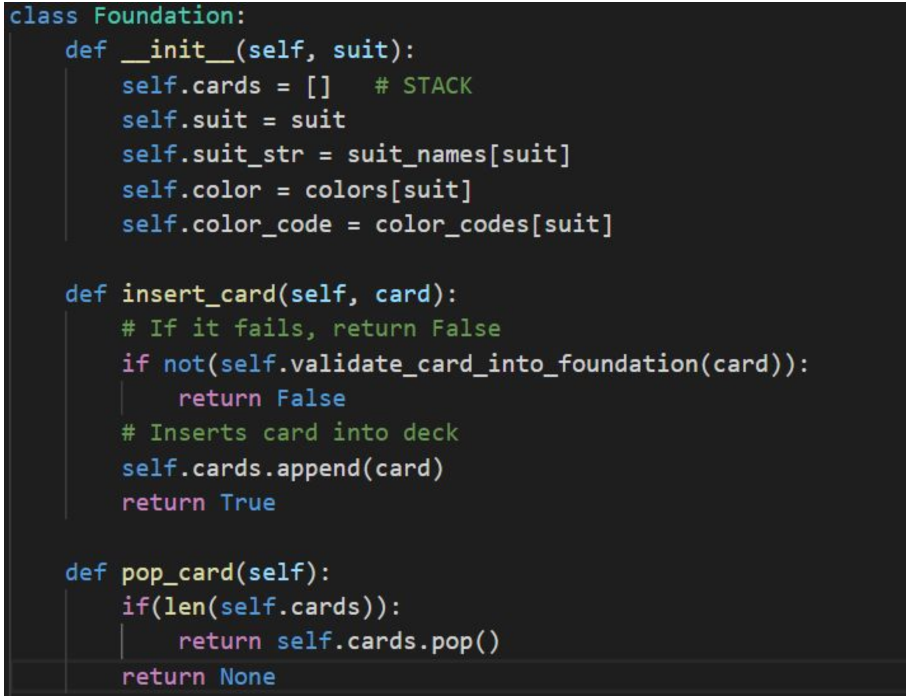
```python
class Deck:
    cards = []

    def __init__(self, cards):
        self.cards = [card.to_face_down() for card in cards]

    def shuffle(self):
        shuffle(self.cards)
        shuffle(self.cards)
        shuffle(self.cards)

    def pop_card(self):
        if(len(self.cards)):
            return self.cards.pop()
        return None
```

# Foundation - Stack

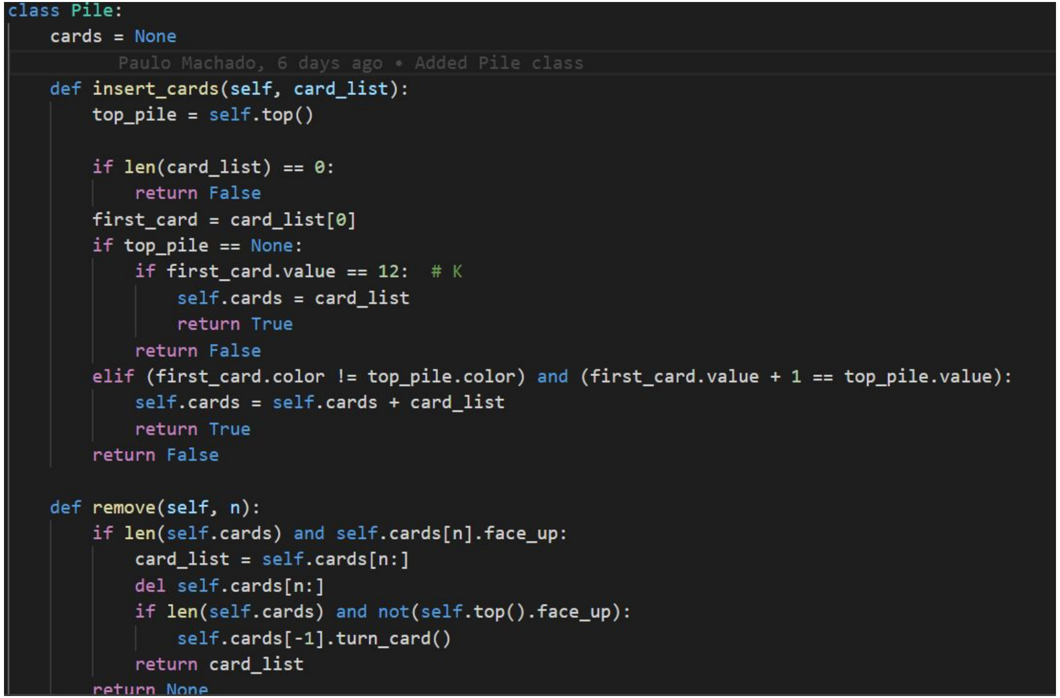


```python
class Foundation:
    def __init__(self, suit):
        self.cards = []    # STACK
        self.suit = suit
        self.suit_str = suit_names[suit]
        self.color = colors[suit]
        self.color_code = color_codes[suit]

    def insert_card(self, card):
        # If it fails, return False
        if not(self.validate_card_into_foundation(card)):
            return False
        # Inserts card into deck
        self.cards.append(card)
        return True

    def pop_card(self):
        if(len(self.cards)):
            return self.cards.pop()
        return None
```

# Pile - List



```
class Pile:
    cards = None
          Paulo Machado, 6 days ago • Added Pile class

    def insert_cards(self, card_list):
        top_pile = self.top()

        if len(card_list) == 0:
            return False
        first_card = card_list[0]
        if top_pile == None:
            if first_card.value == 12:  # K
                self.cards = card_list
                return True
            return False
        elif (first_card.color != top_pile.color) and (first_card.value + 1 == top_pile.value):
            self.cards = self.cards + card_list
            return True
        return False

    def remove(self, n):
        if len(self.cards) and self.cards[n].face_up:
            card_list = self.cards[n:]
            del self.cards[n:]
            if len(self.cards) and not(self.top().face_up):
                self.cards[-1].turn_card()
            return card_list
        return None
```