game.py

```python
from abc import ABC,abstractmethod
from dataclasses import dataclass

class Move:
    def __init__(self, value: int, name: str):
        self.value = value
        self.name = name

@dataclass
class State:
    other_previous_move: Move
    own_blocks: int
    other_blocks: int
    own_has_attack: bool
    other_has_attack: bool
    rounds_left: int

MoveSelection = {
    "a": Move(1, "Attack"),
    "b": Move(2, "Block"),
    "g": Move(4, "Grab"),
    "dp": Move(8, "Dragon Punch")
}

class Player(ABC):
    @property
    @abstractmethod
    def name(self):
        pass

    @abstractmethod
    def act(self, game_state: State) -> Move:
        pass

class Footsies:
    def __init__(self, player1: Player, player2: Player, rounds: int = 1, blocks: int = 3, attackstowin: int = 2, timeout: int = 0):
        self.p1 = player1
        self.p2 = player2
        self.rounds = rounds
        self.attacks = attackstowin

        self.p1_blocks = blocks
        self.p2_blocks = blocks
        self.p1_has_attack = False
        self.p2_has_attack = False
        self.p1_lose = False
        self.p2_lose = False
        self.p1_previous: Move = None
        self.p2_previous: Move = None

        self.timeout = False
        self.timeout_rounds = 0
        self.current_round = 1
        if timeout > 0:
            self.timeout = True
            self.timeout_rounds = timeout
            self.current_round = 0

    def start(self) -> int:
        '''Starts the game loop until a player wins or there's a timeout. Returns the number of the player that won. '''

        def no_timeout():
            return True

        def timeout():
            self.current_round += 1
            print(f"Round {self.current_round}/{self.timeout_rounds}")
            return self.current_round <= self.timeout_rounds

        condition = None
        if self.timeout:
            condition = timeout
        else:
            condition = no_timeout

        while condition():
            rounds_left = self.timeout_rounds - self.current_round
            p1_state = State(self.p2_previous, self.p1_blocks, self.p2_blocks, self.p1_has_attack, self.p2_has_attack, rounds_left)
            p2_state = State(self.p1_previous, self.p2_blocks, self.p1_blocks, self.p2_has_attack, self.p1_has_attack, rounds_left)
            move1 = self.p1.act(p1_state)
            move2 = self.p2.act(p2_state)
            self.p1_previous = move1
            self.p2_previous = move2

            print(f"{self.p1.name} chose {move1.name}. {self.p2.name} chose {move2.name}.")

            p1_hit_attack = False
            p2_hit_attack = False

            match (move1.value - move2.value):
                case 0:
                    print("Same option chosen!")
                case 1:
                    print("Player 1 blocks a hit!")
                    self.p1_blocks -= 1
                case -1:
                    print("Player 2 blocks a hit!")
                    self.p2_blocks -= 1
                case 2:
                    print("Player 2 gets thrown!")
                    self.p2_lose = True
                case -2:
                    print("Player 1 gets thrown!")
                    self.p1_lose = True
                case 3:
                    print("Player 2 lands a hit!")
                    if self.p2_has_attack:
                        self.p1_lose = True
                    p2_hit_attack = True
                case -3:
                    print("Player 1 lands a hit!")
                    if self.p1_has_attack:
                        self.p2_lose = True
                    p1_hit_attack = True
                case 6:
                    print("Player 2 blocks the Dragon Punch and counters!")
                    self.p1_lose = True
                case -6:
                    print("Player 1 blocks the Dragon Punch and counters!")
                    self.p2_lose = True
                case _:
                    if move1.value > move2.value:
                        print("Player 1 lands a Dragon Punch!")
                        self.p2_lose = True
                    else:
                        print("Player 2 lands a Dragon Punch!")
                        self.p1_lose = True

            self.p1_has_attack = p1_hit_attack
            self.p2_has_attack = p2_hit_attack

            if self.p1_lose:
                print("Player 2 wins")
                return 2

            if self.p2_lose:
                print("Player 1 wins")
                return 1

        return 0
```