```
from random import randint
import random

class Die():
    def __init__(self):
        self.roll1= random.randint(1,6)

    def __str__(self):
        return str(self.roll1)

    def roll(self):
        self.roll1=random.randint(1,6)

    def get_value(self):
       return self.roll1

class DiceCup():

    def __init__(self):
        self._dice=[Die() for _ in range(5)]
        self.banked={}
        self.die_index=0

    def value(self,index):
        return self._dice[index].get_value()

    def bank(self,index):
        if index in self.banked:
            pass
        else:
            self.banked.update({self._dice[index]:self._dice[index].get_va
lue()})

    def is_banked(self,index):

        if self._dice[index] in self.banked.keys():
            return True
        else:
            return False

    def release(self,index):
        self.banked.pop(self._dice[index])

    def release_all(self):
        self.banked.clear()

    def roll(self):
        for die in self._dice:
            if not die in self.banked:
                die.roll()

class ShipOfFoolsGame():

    def __init__(self):
```

```
        self._cup = DiceCup()
        self.winning_score = 50

    def has_want(self,val):
        found=False
        for i in range(5):
            if not self._cup.is_banked(i) and self._cup.value(i) == val:
                found = True
        return found

    def has_index(self,val):
        for i in range(5):
            if not self._cup.is_banked(i) and self._cup.value(i) == val:
                return i

    def turn(self):
        has_ship = False #6
        has_captain = False #5
        has_mate = False #4
        self.crew= 0
        self.crew_0=0
        for _ in range(3):
            self._cup.roll()
            if not has_ship and self.has_want(6):
                self._cup.bank(self.has_index(6))
                has_ship = True
            if has_ship and not has_captain and self.has_want(5):
                self._cup.bank(self.has_index(5))
                has_captain = True
            if has_captain and not has_mate and self.has_want(4):
                self._cup.bank(self.has_index(4))
                has_mate = True
            if has_ship and has_captain and has_mate:
                for i in range(5):
                    if self._cup._dice[i] in self._cup.banked.keys():
                        pass
                    else:
                        if self._cup._dice[i].get_value()>3:
                            self._cup.banked.update({self._cup._dice[i]:se
lf._cup.value(i)})

        if has_ship and has_captain and has_mate:
            for i in range(5):
                self.crew = self._cup.value(i) + self.crew
            self.crew_0 = self.crew - 15
        self._cup.release_all()
        return self.crew_0

class Player():

    def __init__(self,name):
        self._name=name
        self._score = 0
```

```
    def current_score(self):
        return self._score

    def reset_score(self):
        self._score = 0

    def play_turn(self,game):
        self._score=self._score + game.turn()
        return self._score

class PlayRoom():

    def __init__(self):
        self._game=ShipOfFoolsGame()
        self._players=[]
        self._winner=''

    def set_game(self,game):
        game.turn()

    def add_player(self,player):
        self._players.append(player)

    def reset_scores(self):
        self._game.crew_0 = 0

    def play_round(self):
        for play in self._players:
            play.play_turn(self._game)

    def game_finished(self):
        for plyr in self._players:
            if plyr.current_score() >= self._game.winning_score:
              return True

    def print_scores(self):
        for plyr in self._players:
            print(plyr._name,'',plyr.current_score())

    def print_winner(self):
        for plyr in self._players:
            if plyr.current_score() >= self._game.winning_score:
              print(plyr._name)

if __name__ == "__main__":
    room = PlayRoom()
    room.set_game(ShipOfFoolsGame())
    room.add_player(Player("Nora"))
    room.add_player(Player("Selman"))
    room.reset_scores()
    while not room.game_finished():
        room.play_round()
        room.print_scores()
```

```
room.print_winner()
```