11/04 DFN - 알고리즘 진행 상황...

향후 계획

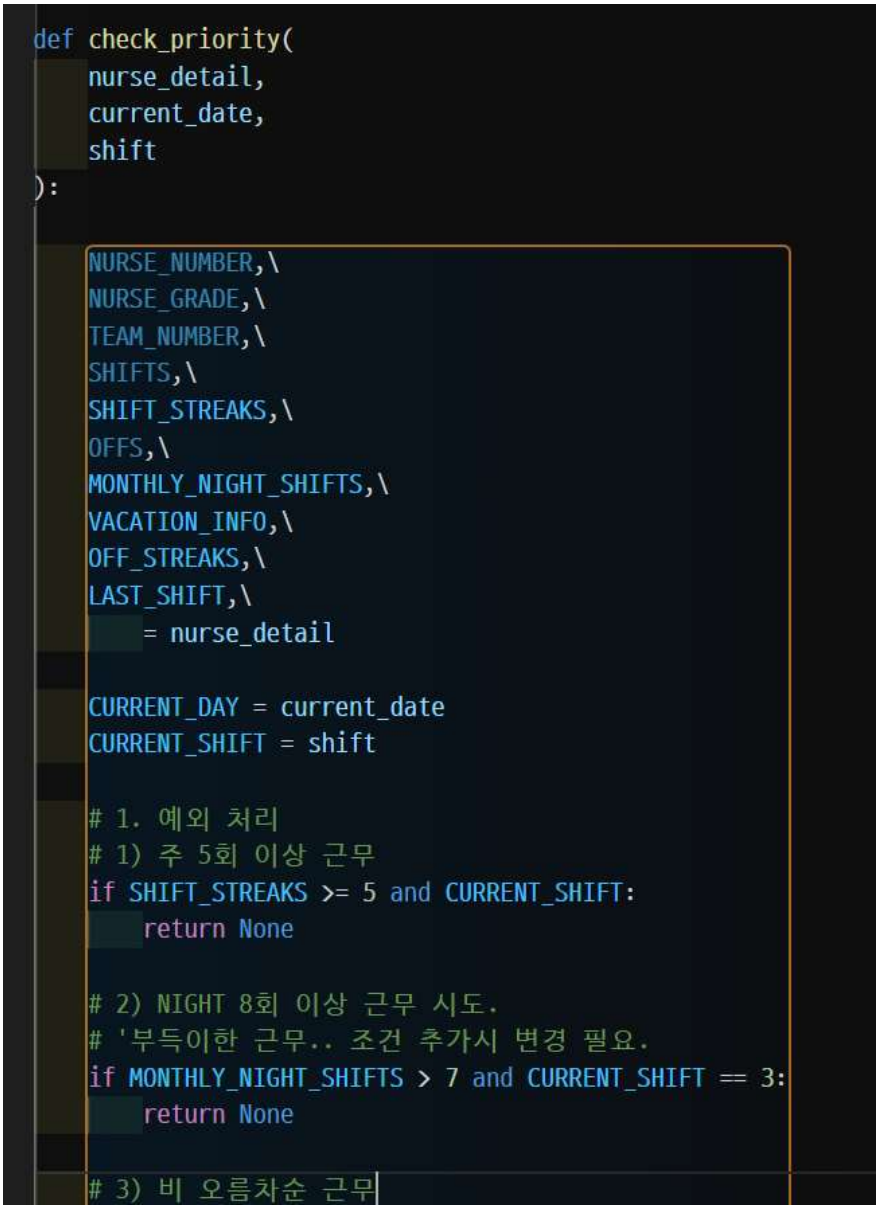
1. 모듈화

`PriorityManager` 에서 종합적으로 우선도 정보를 관리하고

`ScheduleManager` 에서 스케쥴 제작을 도맡아 할 것이다.

앞으로 더 코드를 수정 및 보강하기 위해서는 모듈화가 필수적으로 보인다.

1. 반복되는 코드가 너무 많아졌으며



```
def check_priority(
    nurse_detail,
    current_date,
    shift
):

    NURSE_NUMBER,\
    NURSE_GRADE,\
    TEAM_NUMBER,\
    SHIFTS,\
    SHIFT_STREAKS,\
    OFFS,\
    MONTHLY_NIGHT_SHIFTS,\
    VACATION_INFO,\
    OFF_STREAKS,\
    LAST_SHIFT,\
        = nurse_detail

    CURRENT_DAY = current_date
    CURRENT_SHIFT = shift

    # 1. 예외 처리
    # 1) 주 5회 이상 근무
    if SHIFT_STREAKS >= 5 and CURRENT_SHIFT:
        return None

    # 2) NIGHT 8회 이상 근무 시도.
    # '부득이한 근무.. 조건 추가시 변경 필요.
    if MONTHLY_NIGHT_SHIFTS > 7 and CURRENT_SHIFT == 3:
        return None

    # 3) 비 오름차순 근무
```

2. 유사한 변수명들이 많이 늘어서 변수의 직관성이 떨어졌고

```
# counter_sorted_list를 개인별 딕셔너리 형태로 출력물을 전환하는 함수.
def transfer_table_to_dict(team_list, whole_schedule, nurse_pk_list, days_of_month):
    """
    연결리스트 형식으로 정렬된 스케쥴 리스트를 입력받아
    KEY는 간호사의 PK, Value는 개인의 한 달 스케쥴 리스트인
    dict 개체를 반환하는 함수
    """

    result_dict = dict()
    # 팀 넘버마다 값을 받을 딕셔너리.. 추가.
    # for team_number in team_list:
    #     result_dict[team_number] = dict()
```

3. 함수의 역할이 무엇인지 더 이상 구분을 할 수 없게 되었다.

```
ideal_schedule = make_ideal_counter(needed_nurses_shift_by_team)
nurse_info, nurse_pk_list = divide_nurse_info_by_team(
    team_list=team_list,
    nurse_profile_dict=nurse_profile_dict,
    nurse_last_months_schedule_dict=nurse_last_month_schedule_dict
)

# 1. 종료 조건
```

모듈화가 되면 아래와 같이 코드를 추상화할 수 있을 것이다.

압축 후

`ScheduleManager.parse_nurse_info()`

압축 이전

```
def divide_nurse_info_by_team(
    team_list,
    nurse_profile_dict,
    nurse_last_months_schedule_dict
    ):
    # 1. 선언
    # 출력을 위한 dict 선언.
    divided_nurse_info_dict_by_team = dict()
    nurse_pk_by_team = dict()
    for team_number in team_list:
        divided_nurse_info_dict_by_team[team_number] = dict()
        nurse_pk_by_team[team_number] = list()

    # 2. 연산
    for nurse_detail in nurse_profile_dict.values():

        # 1) 앞부분에 필요한 값들.
        nurse_pk = nurse_detail[0]
        nurse_grade = nurse_detail[1]
        nurse_team = nurse_detail[2]
        nurse_offs = nurse_detail[3]
```

```
        # 2) 뒷부분에 필요한 값들.
        nurse_last_month_schedule = nurse_last_months_schedule_dict[nurse_pk]
        nurse_last_shift = nurse_last_month_schedule[-1]
        nurse_shift_streaks = 0
        if not nurse_last_shift and not nurse_last_month_schedule[-2]:
            nurse_off_streaks = 2
        elif not nurse_last_shift:
            nurse_off_streaks = 1
        else:
            nurse_off_streaks = 0
            for i in range(1, 6):
                if nurse_last_month_schedule[-i]:
                    nurse_shift_streaks += 1
                else:
                    break

        # 3) 딕셔너리에 삽입
        nurse_pk_by_team[nurse_team].append(nurse_pk)

        divided_nurse_info_dict_by_team[nurse_team][nurse_pk] = [
            nurse_pk,
            nurse_grade,
            nurse_team,
            0,
            nurse_shift_streaks,
            nurse_offs,
            0,
            0,
            nurse_off_streaks,
            nurse_last_shift
        ]
    # pprint(nurse_pk_by_team)
    return divided_nurse_info_dict_by_team, nurse_pk_by_team
```

압축 후

```
PriorityManager.compute_priority(shift)
```

압축 이전

```
def check_priority(
    nurse_detail,
    current_date,
    shift
):

    NURSE_NUMBER,\
    NURSE_GRADE,\
    TEAM_NUMBER,\
    SHIFTS,\
    SHIFT_STREAKS,\
    OFFS,\
    MONTHLY_NIGHT_SHIFTS,\
    VACATION_INFO,\
    OFF_STREAKS,\
```

```
    LAST_SHIFT,\
        = nurse_detail

    CURRENT_DAY = current_date
    CURRENT_SHIFT = shift

    # 1. 예외 처리
    # 1) 주 5회 이상 근무
    if SHIFT_STREAKS >= 5 and CURRENT_SHIFT:
        return None

    # 2) NIGHT 8회 이상 근무 시도.
    # '부득이한 근무.. 조건 추가시 변경 필요.
    if MONTHLY_NIGHT_SHIFTS > 7 and CURRENT_SHIFT == 3:
        return None

    # 3) 비 오름차순 근무
    if CURRENT_SHIFT and LAST_SHIFT > CURRENT_SHIFT:
        return None

    # 4) 휴가 전날 NIGHT 근무
    if VACATION_INFO == CURRENT_DAY + 1:
        return None


    # 2. off가 꼭 필요한 경우.
    # 1) 이틀 연속으로 쉬게 해주기.
    if OFF_STREAKS == 1 and not CURRENT_SHIFT:
        priority_token = random.randrange(1, 40)
        return priority_token

    # 2) 5연속 근무 이후 휴무
    if SHIFT_STREAKS >= 5 and not CURRENT_SHIFT:
        priority_token = random.randrange(1, 40)
        return priority_token

    # 3) 휴가 신청일일 경우.
    if VACATION_INFO == CURRENT_DAY:
        priority_token = 0
        return priority_token


    # 3. 기타 연속 근무
    if CURRENT_SHIFT and LAST_SHIFT == CURRENT_SHIFT:
        priority_token = random.randrange(100, 500)
        return priority_token

    # 4. 그 외의 근무
    # 원래는 1000~ 1200. 수정해보자..
    if CURRENT_SHIFT:
        priority_token = random.randrange(150, 600)
        return priority_token

    else:   # 조건 없는 off. 원래는 1000 상수로 리턴했음.
        priority_token = 700
        return priority_token
```

1) PriorityManager

간호사의 `근무 형태` 별 우선순위를 계산하기 위한 정보를 저장한다. 필요시 우선순위를 제공한다.

메서드

1. `self.personalize` 간호사의 이전 근무 기록을 기반으로 자신을 작성한다.
2. `self.push_new_shift` 새로운 근무 기록을 입력, 자신을 업데이트한다.
3. `self.compute_priority` 요청에 따라 우선도를 반환한다.

```
class PriorityManager:

    def __init__(self):
        self.nurse_pk = 0
        self.team_pk = 0
        self.nurse_grade = 0
        self.monthly_shift = 0
        self.monthly_night_shift = 0
        self.offs = 0
        self.last_shift = 0
        self.shift_streaks = 2
        self.weekly_shift = 0
        self.weekly_schedule = deque([0, 0, 0, 0, 0, 0, 0])

    def personalize(self, last_schedule):

        # 1. weekly_schedule 개인화.
        if last_schedule is not None:
            last_weeks_schedule = last_schedule[-8:]
            for shift in last_weeks_schedule:
                self.weekly_schedule.append(shift)
                self.weekly_schedule.popleft()

        # 2. last_shift, shift_streak, weekly_shift.
        for index in range(6, -1, -1):
            shift = self.weekly_schedule[index]

            if shift:
                self.weekly_shift += 1

            if index == 6:
                self.last_shift = shift
                self.shift_streaks = 1

            if index == 5 and shift == self.last_shift:
                self.shift_streaks = 2


    def update_a_shift(self, shift):

        shift_week_ago = self.weekly_schedule.popleft()
        if shift_week_ago: self.weekly_shift -= 1


        # 1. 가장 최근 근무 기록 갱신
```

```
        self.last_shift = shift

        # 1) 휴무
        # 만약 오늘 쉬었다면
        # 연속 근무일수 초기화
        if not shift:
            self.shift_streaks = 0
            self.off_streaks += 1
        # 2) 근무
        else:
            self.monthly_shift += 1
            self.shift_streaks += 1
            self.off_streaks = 0
        # 예외 - 야간 근무
        if shift == 3:
            self.monthly_night_shift += 1
```

2) ScheduleManager

주요 인자: Nurse_info_dict, PriorityManager..

인자값으로 parse된 dict, nurse info 등을 갖고 있다가

요청에 따라 스케쥴을 제작한다.

메서드

`ScheduleManager.create_monthly_schedule()`
`ScheduleManager.update_monthly_schedule()`
`ScheduleManager.check_validation()`