이해한 것이 맞는지 확인 부탁 드립니다.

1.

```java
public class CharacterManager {
    // 이 클래스는 도대체 어떤 업무를 담당하는가 ?
    // 전체 캐릭터를 관리해주는 클래스다!
    // 1. 우선적으로 레이드 멤버를 관리하도록 한다.
    // 2. 그 외적인 요소들도 있겠지만 우선은 레이드만 신경쓰도록 해보자
    //    (그래야 신기능이 추가될때 또 고려할 것이 있다는 것을 볼 수가 있다)

    private final Scanner scan = new Scanner(System.in);

    // 여기서 SelectedCharacter라는 클래스를 추가한 이유는 무엇인가 ?
    // 실제로 Wizard, Knight 등을 이 member에 넣는다고 할 경우
    // 역시나 Object로 받아야 하는 문제점이 있다.
    // SelectedCharacter는 앞서서 RolePlayingGame에서
    // 두 개의 ArrayList를 사용해서 Integer와 Object를 처리 했는데
    // 이 작업 처리 자체를 SelectedCharacter로 위임하여 추상화를 하기 위함이다.
    private ArrayList<SelectedCharacter> member;

    public CharacterManager() { member = new ArrayList<>(); }
```

캐릭터매니저 보러가기

모든 캐릭터 관리하는 매소드 만듦.

저 ArrayList관련해서 얘기해보자면.

예전에는 해당 클래스에 (int,object) 이런식으로 했었는데, 이렇게 관리하면 코드가 복잡해보여서 추상화로 클래스를 하나 더 만들어서 거기서 값을 다 진행하도록 하는 것이 목적인 것 같음..

(맞는지 확인 부탁드립니다.)

생성자에서 ArrayList 초기화 완료

→ SelectedCharcter 클래스에 대한 ArrayList 생성완료.

2,

1- Bm(몬스터 매니저)의 raidTurnStart(CM→케릭터매니저) 매소드 시작.

이전에 배웠던 것은 클래스의 생성자 부분에서 객체를 new로 생성 해 줬었는데, 이번에는 따로 그렇게 하지 않고, (characterManager cm) 이런식으로 적으셨는데 생성자 같이 사용이 가능한 것이 맞죠?

```
public void raidTurnStart (CharacterManager cm) {
    switch (sc.getSelectedNum()) {
        case MonsterNumber.FENRYL:
            ((Fenryl) sc.getCharacter()).raidTurnStart(cm);
            break;
    }
}
```

1. Sc.getSelectedNum → 몬스터매니저의 셀렉케릭터는(10000,fenryl)
2. getSelectedNum = 10000 // monsterNumber.fenryl=10000 일치
3. (fenryl으로 형변환) (sc.getCharacter -> fenryl)(**raidTurnStart->fenryl클래스로 이동**)

```
public void raidTurnStart (CharacterManager cm) {
    // cm에 있는 member(ArrayList)를 활용해서
    // 아래 루틴을 돌 수 있게 하면
    // 앞으로 새로운 기능들을 추가할 때
    // 보다 편리하게 유지보수가 가능해질 것이다.
    SelectedCharacter sc;
    SelectedCharacter monsterSc = new SelectedCharacter(MonsterNumber.FENRYL, character: this);

    for (int i = 0; i < cm.memberSize(); i++) {
        sc = cm.getMemberArrayList().get(i);

        switch (sc.getSelectedNum()) {
            case CharacterNumber.KNIGHT:
                hp -= ((Knight) sc.getCharacter()).qSkill(monsterSc);
                break;
            case CharacterNumber.WIZARD:
                hp -= ((Wizard) sc.getCharacter()).qSkill(monsterSc);
                break;
            case CharacterNumber.SNIPER:
                hp -= ((Sniper) sc.getCharacter()).qSkill(monsterSc);
                break;
            case CharacterNumber.HOLYKING:
                hp -= ((HolyKing) sc.getCharacter()).qSkill(monsterSc);
                break;
            case CharacterNumber.ASSASSIN:
                hp -= ((Assassin) sc.getCharacter()).qSkill(monsterSc);
                break;
        }
    }
```

1) Fenryl) raidTurnStart매소드 들어감 / characterManager 의 값 받음.(..?)
2) selectedCharacter 객체 생성 = sc
3) 또다른 selectedCharacter 생성 = monsterSc →(그 안에 10000,fenryl넣음)

4) for문으로 cm.memberSize→ 캐릭터 매니저에서 캐릭터 3개 선택했었음

sc = cm.getMemberArrayList → 아래와같음. 이걸 이렇게도 만들수있어?

```
public int memberSize () { return member.size(); }
public ArrayList<SelectedCharacter> getMemberArrayList () {
    return member;
}
```

매소드 이름이 getMemberArrayList이고 그거에 대한 데이터타입은

ArrayList<SelectedCharacter>라서 객체 →member를 표시해주는건가요?

이렇게 생긴건 처음봐서 좀 어색하게 느껴집니다.

5) sc = cm.getMemberArrayList().get(i);

ArrayList<SelectedCharacter>를 객체로 표현했기 때문에

SelectedCharacter에 대한 I값(순서대로) 출력가능.

(내가 1,2,4번 출력했다면? →

[0: 1,knight] // [1: 2,wizard] //[3:4,holyking] 이런식으로 데이터 값이 입력되어있다는 뜻일까요?)

3,

(wizard 클래스의 qSkill 보러가기) – damageCalcRequestObject 먼저 보기

```java
public int calcSuperGravityFieldDamage (DamageCalcRequestObject dcro) {
    return (int) (100 * (5.5 * mAtk - dcro.getmDef()) * (iq - dcro.getMen()) * 1.1);
}

@Override
public int qSkill(SelectedCharacter monsterSc) {
    // 몬스터의 숫자를 가지고 어떤 객체로 처리해야 하는지 판정한다.
    // 판정한 몬스터의 객체값을 가지고 데미지 계산을 수행한다.
    // * 여기서 가장 골치 아픈 부분은 판정한 몬스터의 수치값을
    // 현재 이 위치의 calcSuperGravityFieldDamage() 등등과 같
    // 데미지 계산 공식에 적용해줘야 한다는 부분이다.
    // 그러므로 Request용 객체를 만들도록 한다.
    dcro.procDamageCalcRequestObject(monsterSc);

    //int damage = calcSuperGravityFieldDamage((Fenryl) obj);
    int damage = calcSuperGravityFieldDamage(dcro);

    System.out.printf("%10d - 초중력(단일기 - boost 게이지 50 사용)\n",
            damage);

    return damage;
}
```

3, DamageCalcRequestObject 클래스의 이유는,

각 직업군에 fenryl.getmDef 이렇게 쓰면 코드를 재사용하기 어려워서 새로운 클래스에서 알아서 걸러서 값을 체크할 수 있도록 하기 위함이 맞나요?

```java
public class DamageCalcRequestObject {
    private float pAtk, mAtk;
    private float hp, mp;
    private float pDef, mDef;
    private float str, con, dex, agi, iq, men;

    public void procDamageCalcRequestObject (SelectedCharacter monsterSc) {
        switch (monsterSc.getSelectedNum()) {
            case MonsterNumber.FENRYL:
                procAllData((Fenryl) monsterSc.getCharacter());
                break;

            case MonsterNumber.FIELD:
                procAllData((FieldMonster) monsterSc.getCharacter());
                break;
        }
    }

    public void procAllData (Fenryl fenryl) {
        pAtk = fenryl.pAtk;
        mAtk = fenryl.mAtk;
        hp = fenryl.hp;
        mp = fenryl.mp;
        pDef = fenryl.pDef;
        mDef = fenryl.mDef;
        str = fenryl.str;
        con = fenryl.con;
        dex = fenryl.dex;
        agi = fenryl.agi;
        iq = fenryl.iq;
        men = fenryl.men;
    }
```

4, selectedCharacter매소드의 monsterSc 객체 값을 쓰겠다는 뜻으로 보면 되나요?

현재 monsterSc의 값 = (10000,fenryl)

4,

CharacterManager에서

```
private ArrayList<SelectedCharacter> member;

public CharacterManager() {
    member = new ArrayList<>();
}
```

이런식으로 어레이리스트에 selectedCharacter 객체를 배열화 시켰고 그 이후

```
public void procUserInput (int num) {
    SelectedCharacter sc;

    switch (num) {
        case CharacterNumber.KNIGHT:
            Knight kni = new Knight();
            sc = new SelectedCharacter(
                    CharacterNumber.KNIGHT, kni);
            member.add(sc);
            break;

        case CharacterNumber.WIZARD:
            Wizard wiz = new Wizard();
            sc = new SelectedCharacter(
                    CharacterNumber.WIZARD, wiz);
            member.add(sc);
            break;
```

이렇게 사용을 하는데,

매소드 안에 selectedCharacter sc → 이렇게 사용한다면 해당 매소드 안에서만 사용되는 객체가 되는건가요?

그런데 number.add를 해서 값을 입력해서 이미 값을 넣었으니 sc가 매소드안에서 사용이 끝나서 소멸한다고 해도 문제는 없는건가요? → 지역변수 같은 느낌으로 보면되나요?

만약 SelectedCharacter sc를 변수 지정하는 곳에 작성해도 문제없나요?

여기에서 이렇게 선언하는 이유가 궁금합니다.

5, (wizard 클래스 중)

```
public int calcSuperGravityFieldDamage (DamageCalcRequestObject dcro) {
    return (int) (100 * (5.5 * mAtk - dcro.getmDef()) * (iq - dcro.getMen()) * 1.1);
}

@Override
public int qSkill(SelectedCharacter monsterSc) {
    // 몬스터의 숫자를 가지고 어떤 객체로 처리해야 하는지 판정한다.
    // 판정한 몬스터의 객체값을 가지고 데미지 계산을 수행한다.
    // * 여기서 가장 골치 아픈 부분은 판정한 몬스터의 수치값을
    // 현재 이 위치의 calcSuperGravityFieldDamage() 등등과 같은
    // 데미지 계산 공식에 적용해줘야 한다는 부분이다.
    // 그러므로 Request용 객체를 만들도록 한다.
    dcro.procDamageCalcRequestObject(monsterSc);

    //int damage = calcSuperGravityFieldDamage((Fenryl) obj);
    int damage = calcSuperGravityFieldDamage(dcro);

    System.out.printf("%10d - 초중력(단일기 - boost 게이지 50 사용)\n",
            damage);

    return damage;
}
```

dcro.procDamageCalcRequestObject(monsterSc);

➔ 의미가 fenryl의 피 값을 가지고오는건가요?

(아래는 DamageCalcRequestObject 클래스)

```
public class DamageCalcRequestObject {
    private float pAtk, mAtk;
    private float hp, mp;
    private float pDef, mDef;
    private float str, con, dex, agi, iq, men;

    public void procDamageCalcRequestObject (SelectedCharacter monsterSc) {
        switch (monsterSc.getSelectedNum()) {
            case MonsterNumber.FENRYL:
                procAllData((Fenryl) monsterSc.getCharacter());
                break;

            case MonsterNumber.FIELD:
                procAllData((FieldMonster) monsterSc.getCharacter());
                break;
        }
    }
```

```
public void procAllData (Fenryl fenryl) {
    pAtk = fenryl.pAtk;
    mAtk = fenryl.mAtk;
    hp = fenryl.hp;
    mp = fenryl.mp;
    pDef = fenryl.pDef;
    mDef = fenryl.mDef;
    str = fenryl.str;
    con = fenryl.con;
    dex = fenryl.dex;
    agi = fenryl.agi;
    iq = fenryl.iq;
    men = fenryl.men;
}
```

➔ 코드에서는 어떻게 사용이 되나요? wizard의 damage의 값이 fenryl 클래스의 qSkill(monsterSc)값이 된다고 생각하는데, fenryl클래스 내에서도 이 보스의 피 값을 다 알고 있지 않을까요? procDamageCalcRequestObject 클래스에서 한번 더 보스의 피 값을 초기화 해주는 이유가 궁금합니다.

```
public void raidTurnStart (CharacterManager cm) {
    SelectedCharacter sc;
    SelectedCharacter monsterSc = new SelectedCharacter(MonsterNumber.FENRYL,  character: this);

    for (int i = 0; i < cm.memberSize(); i++) {
        sc = cm.getMemberArrayList().get(i);

        switch (sc.getSelectedNum()) {
            case CharacterNumber.KNIGHT:
                hp -= ((Knight) sc.getCharacter()).qSkill(monsterSc);
                break;
            case CharacterNumber.WIZARD:
                hp -= ((Wizard) sc.getCharacter()).qSkill(monsterSc);
```