2022.1.14 (19교시 복습) 게임만들기 최종(아마도)

(오늘도 레전드 찍었다. 오늘은 클래스가 심지어 20개네 ㅋ.. 필요없는 부분 말고, 중요하게 봐야 할 부분만 집중적으로 판다. 이건 ArrayList의 문제가 아님.. 휴..)

선생님이 생각하신 어제 만든 코드의 문제점

```
1. RolePlayingGame 클래스에 그냥 대놓고 사용하려고 하는 모든 클래스가 미리 선 할당되어 있음
-> 파생되는 문제점
   -> 다루는 캐릭터들이 많아지고 확장됨에 따라
      모든 클래스를 다 때려 넣어야 하기에 상당히 지저분한 코드가 만들어짐
      (결국 유지보수가 매우 난해해진다는 문제점이 있음)

2. ArrayList 가 member 와 이를 검사하는 구조로 비효율적으로 구성됨
-> 파생되는 문제점
   -> 가독성이 떨어짐 (레이드 멤버까지 알겠다가 갑자기 숫자는 뭐지 ???)
   -> ArrayList 를 다루는데 있어 논리적으로 복잡해지는 문제점이 존재함
      (한마디로 한 눈에 대략 뭐 하는지 감을 잡기가 어려워짐)

3. chooseMember()와 같은 역할은 CharacterManager 가 해주는 것이 맞음
-> 파생되는 문제점
   -> 하나의 도메인이 여러 작업을 맡아서 진행하기 때문에 복잡성이 증대함
      (결론적으로 규모가 커지면 혼선이 발생할 수 있음)

4. 시간내 완성하기 위해 기존에 잘 돌던 코드가 동작하지 않게 망쳐놓음
-> 파생되는 문제점
   -> 어떤 프로그램을 만들때 기존에 잘 돌던것이 돌지 않게 되면 매우 심각한 문제임 (현업에서)
      (서비스가 돌고 있는 시점에 서비스가 폭파하는 '대.참.사' 가 발생함)
   # 우리는 교육이라는 특성상 정해진 시간내 마무리를 위해서 일종의 꼼수를 사용했지만
     현업에서는 이런 상황이라면 차분하게 그냥 하루 미루는 것이 좋습니다.

5. 보스 레이드 컨텐츠 개발 자체에 매우 심각한 문제가 있음
-> 파생되는 문제점
   -> 펜릴은 Fenryl 추가하고 다른 보스는 또 추가하고, 또 다른건 또 추가 해야 하는가 ?
      (유지보수 차원에서 말도 안되게 지저분한 상황이 연출될 것임)

6. 일단 펜릴만 존재했기 때문에 확장성을 고려하지 않고
   데미지 계산식에 펜릴만을 설정하였음
-> 파생되는 문제점
   -> 다른 몬스터가 추가되는 경우 타입 캐스팅 등의 문제점들이 발생할 수 있음
      (결국 다른 몬스터 처리가 안되는 상황이 발생함)
   [1] 여기서 최악의 수는 무엇인가 ?
       -> 몬스터마다 처리하는 매서드를 만드는 것임
          -> 이게 왜 최악의 수인가 ?
             -> 중복 코드로 인한 관리가 안됨
                (중복이 안좋은 이유는 하도 많이 중복되다보니
                 뭐가 어디에 있는지 알 수가 없게 되는 상황이라 지양해야 하는 것임)
```

(바꾸기 전) 문제가 된다고 했던 부분 1,2

직접 하나하나 클래스 입력 이거 디게 비효율적임.

캐릭터 늘어나면 하나하나 다 클래스 입력해줘야함..

ArrayList도 두개나 있어서 관리하기가 너무 번거로움

```
public class RolePlayingGame extends MacroSet {
    // 정석은 CharacterManager를 하나 만들어서
    // 이 가상 세계에 있는 캐릭터들의 턴 관리를 해줘야 한다.
    // 현재 예제에서는 위자드 클래스 하나만 활용하므로
    // 별도의 CharacterManager를 생성하지 않도록 한다.
    Wizard wiz;
    Knight kni;
    Sniper sni;
    HolyKing hk;
    Assassin sin;
    // 향후 CharacterManager로 업데이트!

    ArrayList<Object> raidMember;
    ArrayList<Integer> memberCheckList;
```

(바뀐 클래스)

```
public class RolePlayingGame extends MacroSet {

    CharacterManager cm;
    MonsterManager bm;

    final int MAX = 10000;
    final int MIN = 100;

    int range;
```

캐릭터 매니저 & 몬스터 매니저를 배치하여,

캐릭터 매니저 → 기사, 법사 등등 직업가진 캐릭터를 사용하기 쉽게 객체를 만들었다.

중요한 점은 각각의 직업군이 들어나지 않기 때문에, 만약 캐릭터가 추가된다면

캐릭터 매니저에서만 추가 → rolePlayingGame 클래스는 따로 변동없이 새로운 캐릭터 사용이 가능하다.

만약?? 이전처럼 캐릭터 클래스를 모두 객체화 했다면? 뭐 하나 수정하면 고치기가 매우 애매하거나, 번거로웠을 것이다.

독립성을 가지면서 수정, 보안이 편하게 만들었음.

(문제되는 부분 3)

➔ ploePlayingGame에서 멤버선택을 하는건 아무래도 클래스 취지와는 맞지 않아서인 듯 하다.

➔ 직업을 선택하는건 캐릭터를 관리하는 매니져 클래스에서 하는 것이 더 좋을 것이라고 판단

```
public void chooseMember () {
    System.out.println("1번: 기사, 2번: 위자드, 3번: 스나이퍼, 4번: 성황, 5번: 어쌔신");

    for (int i = 0; i < 3; i++) {
        System.out.print("번호를 입력하세요: ");
        int num = scan.nextInt();

        switch (num) {
            case KNIGHT:
                raidMember.add(kni);
                memberCheckList.add(KNIGHT);
                break;
            case WIZARD:
                raidMember.add(wiz);
                memberCheckList.add(WIZARD);
                break;
            case SNIPER:
                raidMember.add(sni);
                memberCheckList.add(SNIPER);
                break;
            case HOLYKING:
                raidMember.add(hk);
                memberCheckList.add(HOLYKING);
                break;
            case ASSASSIN:
                raidMember.add(sin);
                memberCheckList.add(ASSASSIN);
                break;
            default:
                continue;
        }
    }
}
```

```
3. chooseMember()와 같은 역할은 CharacterManager가 해주는 것이 맞음
-> 파생되는 문제점
   -> 하나의 도메인이 여러 작업을 맡아서 진행하기 때문에 복잡성이 증대함
      (결론적으로 규모가 커지면 혼선이 발생할 수 있음)
```

```
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)
```

또 문제가 되는 코드는 아래와 같다.

```
6. 일단 펜릴만 존재했기 때문에 확장성을 고려하지 않고
   데미지 계산식에 펜릴만을 설정하였음
-> 파생되는 문제점
   -> 다른 몬스터가 추가되는 경우 타입 캐스팅 등의 문제점들이 발생할 수 있음
      (결국 다른 몬스터 처리가 안되는 상황이 발생함)
   [1] 여기서 최악의 수는 무엇인가 ?
       -> 몬스터마다 처리하는 매서드를 만드는 것임
          -> 이게 왜 최악의 수인가 ?
             -> 중복 코드로 인한 관리가 안됨
                (중복이 안좋은 이유는 하도 많이 중복되다보니
                 뭐가 어디에 있는지 알 수가 없게 되는 상황이라 지양해야 하는 것임)
```

(Wizard 클래스)_변경전

```java
public int calcSuperGravityFieldDamage (Fenryl target) {
    return (int) (100 * (5.5 * mAtk - target.mDef) * (iq - target.men) * 1.1);
}

@Override
public int qSkill(Object obj) {
    int damage = calcSuperGravityFieldDamage((Fenryl) obj);
    System.out.printf("%10d - 초중력(단일기 - boost 게이지 50 사용)\n",
            damage);

    return damage;
}
```

어떤 게 문제인지 알겠는가??

바로 다른 보스는 생각하지 않고, Fenryl만 적용 가능한 클래스를 만들었다는 것이다.

만약, 다른 보스를 만들고 싶다?

그럼 복붙해서, Fenrly 라고 되어있는 곳에 다른 보스 클래스를 작성해야 할까?

그렇게 되면 코드 중복이 심해질 것이다.

(변경 후 Wizard 클래스)_ 변경후

```
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;
}
```

이렇게 보면 뭔말인지 잘 모르겠음.... → 이건 한번 자세히 살펴봐야할 것 같다.

한가지 좀 봐야할 것은. 아까 fenryl이라고 써져있었던 곳에

SelectedCharacter monsterSc가 들어왔다. 이 얘기는 나중에 하겠지만 몬스터 매니져 클래스를 사용하여 어떤 보스를 고를지 체크해서 여기에 값을 대입하는 것.

동일하게 보스를 추가할 때, 해당 클래스는 건들이지 않아도 구동이 가능하다.

이런게 바로 잘 만들어진 코드라고 할 수 있다.

자, 그럼 이제 코드를 좀 더 자세히 보러 가자.. 덜덜 두렵다.

** 추가된 클래스**

(오늘진행한 클래스는 총 19개로 8개가 더 추가되었다. 추가된 것만 살펴보자)

기본 : adventurer, 각 직업 클래스, rolePlayingGame, 메인클래스 정도

추가 : 캐릭터 매니저, 캐릭터 넘버 // DamageCalcRequestObject(몬스터 데미지 체크용??) //

Fenryl(보스), 필드 몬스터(보스x)// 몬스터 매니저, 몬스터 넘버 // SelectedCharacter(..?)

이렇게 볼 수 있다. 하나씩 살펴보자.

```
public class FantasyCollectors {
    public static void main(String[] args) throws InterruptedException {
        RolePlayingGame rpg = new RolePlayingGame();


        rpg.bossRaid();
    }
}
```

RolePlayingGame 클래스 호출 / bossRaid매소드 요청했음

```
매크로 구동 준비 완료!
레이드 보스를 선택하세요!
1번: 펜릴, 2번: 미구현, 3번: 미구현, 4번: 미구현, 5번: 미구현
번호를 입력하세요: 1
파티 멤버를 선별하세요!
1번: 기사, 2번: 위자드, 3번: 스나이퍼, 4번: 성황, 5번: 어쌔신
번호를 입력하세요: 1
번호를 입력하세요: 2
번호를 입력하세요: 4
1 턴입니다
절대자의 가호(버프)
5초간 모든 데미지에 면역(상태 이상 면역)
  25535400 - 초중력(단일기 - boost 게이지 50 사용)
얼티메이트 레이즈
죽은 아군 전원 부활
2 턴입니다
절대자의 가호(버프)
5초간 모든 데미지에 면역(상태 이상 면역)
  25535400 - 초중력(단일기 - boost 게이지 50 사용)
얼티메이트 레이즈
죽은 아군 전원 부활
```

메인매소드 & 출력값 미리보기.

문제없이 출력이 되고있음. 어떻게 진행된 건지 하나씩 살펴보자.

(RolePlayingGame 클래스의 huntStart 매소드 보기)

보기 전 해당 클래스 전체적으로 보러 가보자.

1.변수생성 및 생성자 만들기 (캐릭터매니저/몬스터매니저 클래스 만들어서 객체생성)

```
public class RolePlayingGame extends MacroSet {

    CharacterManager cm;
    MonsterManager bm;

    final int MAX = 10000;
    final int MIN = 100;

    int range;

    public RolePlayingGame () {
        // 몬스터 관리 객체
        bm = new MonsterManager();
        // 캐릭터 관리 객체
        cm = new CharacterManager();

        range = MAX - MIN + 1;
    }
```

2, 캐릭터매니저 보러가기

```
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<>(); }
```

생성자에서 ArrayList 초기화 완료

→ SelectedCharcter 클래스에 대한 ArrayList 생성완료.

모든 캐릭터 관리하는 매소드 만듦.

저 ArrayList관련해서 얘기해보자면.

예전에는 (int,object) 이런식으로 했었는데, 이렇게 관리하기가 귀찮으니 추상화로 클래스를 하나 더 만들어서 거기서 값을 다 진행하도록 하는 것이 목적인 것 같음..

(맞는지 확인 부탁드립니다.)

(SelectedCharacter 클래스)

```
public class SelectedCharacter {
    private Integer selectedNum;
    private Object character;

    public SelectedCharacter (Integer selectedNum, Object character) {
        this.selectedNum = selectedNum;
        this.character = character;
    }

    public Integer getSelectedNum() { return selectedNum; }

    public Object getCharacter() { return character; }
}
```

selectedNum → Integer인 이유

ArrayList를 받는 클래스인 경우 int가 아니고 Integer로 써야함.

selectedNum -> 숫자 받음 // character -> 클래스 받음 → 전에 ArrayList두개 만들었던거랑 똑같은 기능 → 즉 클래스 하나 더 추가해서 추상화를 해준다는 뜻.

3, RolePlayingGame 클래스의 bossRaid 매소드 보기(중요)

```
public void bossRaid () throws InterruptedException {
    // 레이드를 치룰 보스를 선택합니다!
    chooseRaidBoss();

    // 레이드 출전 팀을 선별합니다!
    chooseMember();

    int turnCnt = 1;

    // 보스가 죽었는지 판정
    while (!bm.isDead()) {
        System.out.printf("%d 턴입니다\n", turnCnt++);
        bm.raidTurnStart(cm);
        Thread.sleep( millis: 2000);
    }

    System.out.println("승리!!!");
}
```

3_1) chooseRaidBoss() 매소드 보기

```
public void chooseRaidBoss () {
    System.out.println("레이드 보스를 선택하세요!");
    bm.chooseRaidBoss();
}
```

3_2) bm(몬스터매니저 클래스)의 chooseRaidBoss 매소드 가기

```
public void chooseRaidBoss () {
    int cnt = 0;

    System.out.println("1번: 펜릴, 2번: 미구현, 3번: 미구현, 4번: 미구현, 5번: 미구현");

    while (cnt < 1) {
        System.out.print("번호를 입력하세요: ");
        int num = scan.nextInt();

        if (num >= 6 || num <= 0) {
            continue;
        }

        cnt++;

        procUserInput(num);
    }
}
```

보스 선택은 몬스터매니저 클래스로가서 해줘야함. 나머지는 문제없으니,

procUserInput(num<-scan출력) 매소드로 이동

3_3) bm(몬스터매니저 클래스)의 procUserInput 매소드 가기

```
final int BIAS = 9999;

public void procUserInput (int num) {
    switch (num + BIAS) {
        case MonsterNumber.FENRYL:
            Fenryl fenryl = new Fenryl();
            sc = new SelectedCharacter(
                    MonsterNumber.FENRYL, fenryl);
            break;

        default:
            break;
    }
}
```

Num+BIAS로 한 이유를 보려면 잠시 monsterNumber 클래스 좀 가봐야할듯

```
public class MonsterNumber {
    static final int FENRYL = 10000;
    static final int FIELD = 10001;
}
```

Num ->1입력 시, 9999더해줘야 10000되니까 BIAS 추가 한 것임.

보통 상수 입력할 때, 1의 자리 수가 이미 끝났으면, 그 다음엔 다음 캐릭터가 추가 될 수도 있다는 것을 생각해서 적당히 큰 수 10000 이런 식으로 해서 겹치지 않도록 체크하는 것이 좋다.

하나 더 봐야하는 것, 바로 MonsterNumber.FENRYL 이다.

급 생각났던게,.. 분명 이런식으로 남의 클래스의 변수 쓰지 말라고 했던 것 같은데 이번에 가능한 이유는 몬스터 넘버 클래스에 static final로 전역변수와 상수를 선언했기 때문에 변동 될 가능성이 없기 때문이다. 그래서 이렇게 써도 문제 없음.

```
Fenryl fenryl = new Fenryl();
                sc = new SelectedCharacter(
                        MonsterNumber.FENRYL, fenryl);
```

이것도 좀 봐보자, 1-fenryl 클래스를 객체화시킴

2- SelectedCharacter 객체생성하면서 > MonsterNumber.FENRYL(**10000**), fenryl(해당 **클래스**) 입력

3_4) SelectedCharacter 클래스 가보기

```
public class SelectedCharacter {
    private Integer selectedNum;
    private Object character;

    public SelectedCharacter (Integer selectedNum, Object character) {
        this.selectedNum = selectedNum;
        this.character = character;
    }

    public Integer getSelectedNum() { return selectedNum; }

    public Object getCharacter() { return character; }
}
```

아까 넣은 값 10000,fenryl 이 값이 해당 클래스에서 초기화를 해주면서 값을 가지게 되었다.

몬스터매니져에 intput 매소드의 값이 (10000,fenryl) 이라고 봐야하나? 그리고 이 값을

RolePlayingGame으로 가지고온다.

bm.chooseRaidBoss(); 의 값은 (10000,fenryl)이다.

3, RolePlayingGame 클래스의 huntStart 매소드 보기(중요)

```
public void bossRaid () throws InterruptedException {
    // 레이드를 치룰 보스를 선택합니다!
    chooseRaidBoss();

    // 레이드 출전 팀을 선별합니다!
    chooseMember();

    int turnCnt = 1;

    // 보스가 죽었는지 판정
    while (!bm.isDead()) {
        System.out.printf("%d 턴입니다\n", turnCnt++);
        bm.raidTurnStart(cm);
        Thread.sleep( millis: 2000);
    }

    System.out.println("승리!!!");
}
```

chooseRaidBoss완료 // chooseMember 매소드 가보기.

3_1) 동일 클래스의 chooseMember 매소드 가기

```
public void chooseMember () {
    System.out.println("파티 멤버를 선별하세요!");
    cm.chooseMember();
}
```

3_2) CM(캐릭터매니저)의 chooseMember매소드 가보기

```
public void chooseMember () {
    int cnt = 0;

    System.out.println("1번: 기사, 2번: 위자드, 3번: 스나이퍼, 4번: 성황, 5번: 어쌔신");

    while (cnt < 3) {
        System.out.print("번호를 입력하세요: ");
        int num = scan.nextInt();

        if (num >= 6 || num <= 0) {
            continue;
        }

        cnt++;

        procUserInput(num);
    }
}
```

맴버 선택은 캐릭터 매니저에서 하는 것이 좋다. 그래서 여기로 넘어옴.

전체적인 흐름은 이해가 가니, procUserInput(num)<-scan했음 // 으로 가보자.

3_3) CharacterManager 클래스의 procUserInput 매소드 가보기

```
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;
```

(너무 길어 생략)

1- SelectedCharacter 객체 생성

2- characterNumber클래스에서 wizard 를 호출함 → case 2 가 됨.

```
public class CharacterNumber {


    static final int KNIGHT = 1;
    static final int WIZARD = 2;
    static final int SNIPER = 3;
    static final int HOLYKING = 4;
    static final int ASSASSIN = 5;
}
```

(wizard = 2, 아까와 동일하게 상수로써 표현)

3- wizard 생성자 가지고오기.

4- sc(SelectedCharacter클래스) 객체 생성하면서(2, wizard 클래스 입력)

5- member (ArrayList_SelectedCharacter클래스)에 add(sc(SelectedCharacter클래스)→(2,wizard)

즉, member[0]->(2,wizard가 들어있는것이다.) 휴...

3, RolePlayingGame 클래스의 huntStart 매소드 보기(중요)

```
public void bossRaid () throws InterruptedException {
    // 레이드를 치룰 보스를 선택합니다!
    chooseRaidBoss();

    // 레이드 출전 팀을 선별합니다!
    chooseMember();

    int turnCnt = 1;

    // 보스가 죽었는지 판정
    while (!bm.isDead()) {
        System.out.printf("%d 턴입니다\n", turnCnt++);
        bm.raidTurnStart(cm);
        Thread.sleep( millis: 2000);
    }

    System.out.println("승리!!!");
}
```

chooseRaidBoss, chooseMember 완료 // while문 보기.

turnCnt는 몇번돌면서 보스를 물리쳤냐 에 대한 내용

3_1)bm(몬스터매니저 클래스)의 isDead매소드 보기

```
public boolean isDead () {
    boolean deadFlag = false;

    switch (sc.getSelectedNum()) {
        case MonsterNumber.FENRYL:
            deadFlag = ((Fenryl) sc.getCharacter()).isDead();
            break;
    }
```

1- deadFlag →false로 작성

2- switch의 sc(SelectedCharacter클래스)의 getSelectedNum 호출

(bm : scanner로 fenryl로 하기로 결정했었음 → selectNum=10000)(맞을까요?)

```java
public class SelectedCharacter {
    private Integer selectedNum;
    private Object character;

    public SelectedCharacter (Integer selectedNum, Object character) {
        this.selectedNum = selectedNum;
        this.character = character;
    }

    public Integer getSelectedNum() {
        return selectedNum;
    }
```

3- switch(10000) 실행 후

```java
public boolean isDead () {
    boolean deadFlag = false;

    switch (sc.getSelectedNum()) {
        case MonsterNumber.FENRYL:
            deadFlag = ((Fenryl) sc.getCharacter()).isDead();
            break;
    }

    return deadFlag;
```

Case 10000 으로 실행

deadFlag = ~~ 부분의 뜻이...

fenryl 클래스일때 sc.getCharacter → fenryl이다.(??)

다시보면 (fenryl) fenryl의 isDead 매소드를 보러가자는 뜻??(맞는지 확인 부탁드리기)

4- ((Fenryl) sc.getCharacter()).isDead() 이부분 확인하러 가기

```java
public boolean isDead () {
    if (hp <= 0) {
        return true;
    } else {
        return false;
    }
}
```

Fenryl 클래스→ 아직 게임 시작도 안함 → hp는 0 이상 → false

3, RolePlayingGame 클래스의 huntStart 매소드 보기(중요)

```
public void bossRaid () throws InterruptedException {
    // 레이드를 치룰 보스를 선택합니다!
    chooseRaidBoss();

    // 레이드 출전 팀을 선별합니다!
    chooseMember();

    int turnCnt = 1;

    // 보스가 죽었는지 판정
    while (!bm.isDead()) {
        System.out.printf("%d 턴입니다\n", turnCnt++);
        bm.raidTurnStart(cm);
        Thread.sleep( millis: 2000);
    }

    System.out.println("승리!!!");
}
```

chooseRaidBoss, chooseMember 완료 // while문 보기.

1- 아까 isDead = false // !false = true ➔while문 진행 가능

2- turnCnt는 while문이 진행하면서 +1하게 됨.

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

저기 안에 CM을 왜 넣은거지?? 봐야 알 듯...

```
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**)

```java
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 객채 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?

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

   매소드 이름이 getMemberArrayList이고 그거에 대한 데이터타입은

   ArrayList<SelectedCharacter>라서 객체 →member를 표시해주는건가.

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 출력이 된다는 뜻일까요?)

```
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;
```

이부분 너무 기니까 좀 끊어서 보자. 앞에 [0: 1,knight],[1: 2,wizard],[3:4,holyking] 이게 맞다면

Wizard 기준 ChracterNumber.WIZARD(2)

Hp -= (wizard 데이터타입→ wizard클래스에서 값을 도출해내라.?)

(Sc.getCharacter() →wizard)의 qSkill매소드 들어가는데(monsterSc :10000,fenryl) 값도 같이 들고가야한다.

➔(wizard형변환)wizard.qSkill(monsterSc:10000,fenryl) – 이런느낌으로 보면 되겠다.

참고 : 현재 monsterSc 는(0000,fenryl) 이다.

(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 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;
}
```

2, 전체적으로 이해는 가는데 *100 // *1.1을 한 이유가 궁금합니다.

1, DamageCalcRequestObject의 객체 선언을 따로

하지 않았는데도 이렇게 사용이 괜찮은가요??

소가로 안에서 DamageCalcRequestObject의 객체 dcro를 선언한 것 같은데 보통 저희는 객체 선언할 때 new~ 이런식으로

사용했어서, 이게 객체가 아닌건지 궁금합니다.

이 글의 뜻은 DamageCalcRequestObject의 클래스를 쓰고

그것을 dcro로 부르겠습니다. -> 이것이 맞을까요?

```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;
    }
```

3, DamageCalcRequestObject 클래스의 이유는,

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

4, selectedCharacter매소드의 monsterSc 객체 값을 쓰겠다는 뜻으로 보면 되나요?

현재 monsterSc의 값 = (10000,fenryl)

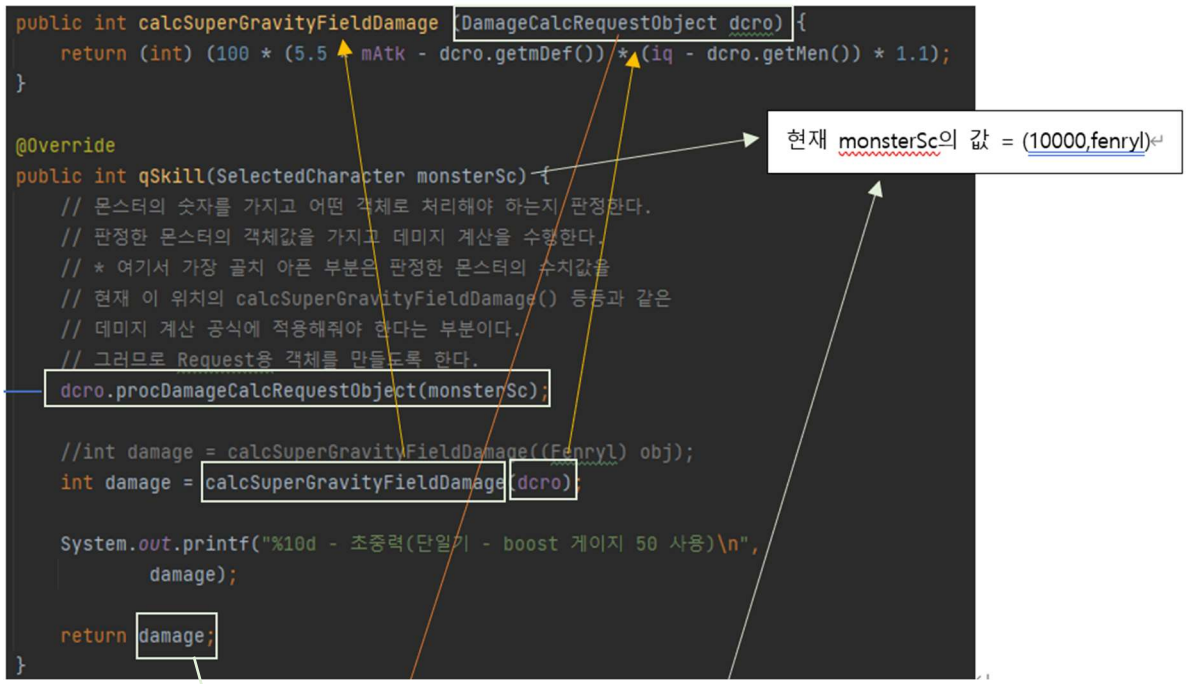
(wizard 클래스의 qSkill 보러가기)

```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 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;
}
```

현재 monsterSc의 값 = (10000,fenryl)

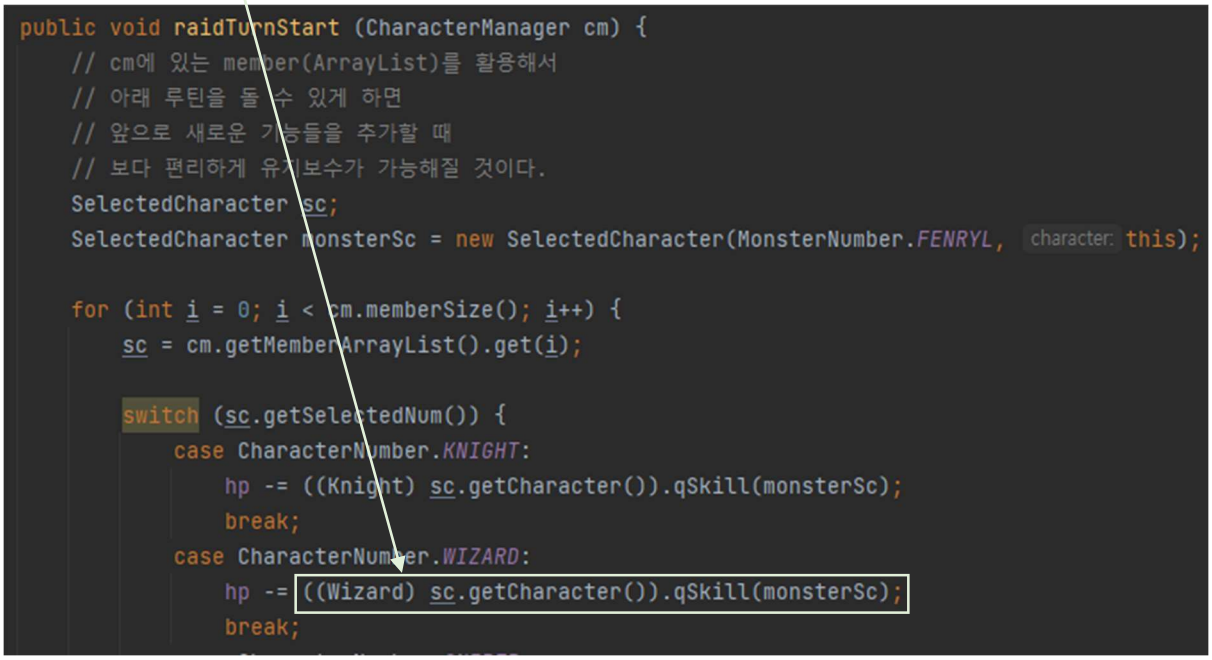
```java
public class DamageCalcRequestObject {
    private float pAtk, mAtk;
    private float hp, mp;
    private float pDef, mDef;
    private float str, con, dex, agi, iq, men;

    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;
    }
```

이 매소드에서 fenryl의 정보값을 불러서 맞춰주고있음.



calcSuperGravityFieldDamage의 값은 → damage

이 값을 아래에 넣는다.



Hp에 damage값을 뺀다. 첨으로 다시 돌아가자.

```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 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;
}
```

calcSuperGravityFieldDamage 매소드에서 dcro를 쓸 수 있는 것은 소가로 안에 생성이 되었다고 생각한다면, 지역변수 느낌으로 생성되는 건 줄 알았는데,

그게 아니고 어디서든 쓸 수 있도록 객체생성이 완료되어서 다른 매소드에서도 사용이 가능한 걸로 보면 될까요?

New ~ 이렇게 객체 생성 안하고 그냥 저런식으로 써도 알아서 객체 생성이 되는건가요?

```java
protected DamageCalcRequestObject dcro;
```

찾아보니!! adventure에 이렇게 이미 객체 생성이 되었기 때문에 사용이 가능한거구나!! 오켕~