< InterfaceProblem >

[InterfaceProblem(main) - RolePlayingGame - Wizard / MacroSet]

(상속) 1. Adventurer - Magician/Warrior - Wizard/Knight
2. MacroSet - RolePlayingGame
1. Adventurer 클래스로부터 상속을 받은 하위 클래스들은
부모 클래스인 Adventurer의 변수와 메소드를 사용할 수 있다.
2. MacroSet 클래스로부터 상속을 받은 하위 클래스 RolePlayingGame 클래스는
부모 클래스인 MacroSet의 변수와 메소드를 사용할 수 있다.

(인터페이스) Skill - Adventurer
SKill이라는 인터페이스를 Adventurer 클래스에서 구현한다.

- InterfaceProblem(main)

```
RolePlayingGame rpg = new RolePlayingGame();

rpg.gameStart();
```

생성자를 호출해 RolePlayingGame 타입의 객체 rpg를 생성한다.
rpg 객체의 메소드 gameStart를 호출한다.

- RolePlayingGame

1. 변수/클래스 선언

```
Wizard wiz;

final int THIRD = 3;
final int MAX = 10000;
final int MIN = 100;

int range;
```

매크로를 적용할 직업인 Wizard 클래스의 객체 wiz를 선언한다.
wizard에 대한 상수값 THIRD를 선언한다.
경험치는 100~ 10000 사이의 난수로 처리된다.
이 때 경험치 발생 범위를 구하기 위한 변수 MAX, MIN, range를 선언한다.

2. 생성자

```
public RolePlayingGame () {
    wiz = new Wizard();
    range = MAX - MIN + 1;
}
```

이 클래스의 생성자가 호출되면 Wizard 클래스의 생성자를 호출해 wiz 객체를 생성하고 range값을 계산한다.

3. gameStart

```
public void gameStart () throws InterruptedException {
    // 와일 루프가 매턴으로 계산됨
    // 매크로 돌린다 가정하고 특정 패턴을 반복한다 가정
    while (true) {
        doMacroSet(THIRD, wiz);
        wiz.calcCharcterExp(calcExps(), THIRD);
        wiz.printInfo();
    }
}
```

RolePlayingGame 클래스는 부모 클래스인 MacroSet 클래스의 메소드 doMacroSet을 사용할 수 있다.
doMacroSet 메소드를 통해 정해진 직업의 행동패턴이 무한반복문 안에서 수행된다.
이 때 변수 THIRD와 객체 wiz를 메소드의 매개변수로 넘겨준다.

doMacroSet이 완료되면 객체 wiz의 메소드 calcCharacterExp 메소드를 호출해
캐릭터 경험치를 계산하고, 갱신된 wiz객체의 정보를 출력한다.

- Wizard 클래스

1. Wizard 클래스의 상속관계
   : [Adventurer - Magician - Wizard]

- Adventurer 클래스
1). 변수 선언

```
private float pAtk, mAtk;
private float hp, mp;
private float pDef, mDef;
private float str, con, dex, agi, iq, men;
private int level;
private int reqExp;
private int curExp;   // 기본 스탯 선언
```

: 캐릭터의 기본 스탯

```
final float PATK = 50;
final float MATK = 50;
final float HP = 100;
final float MP = 100;
final float PDEF = 1;
final float MDEF = 1;
final int STR = 10;
final int CON = 10;
final int DEX = 10;
final int AGI = 10;
final int IQ = 10;
final int MEN = 10;
final int LEVEL = 1;
final int REQEXP = 100;
final int CUREXP = 0;    //초기 스탯
```

: 모험가의 초기 스탯

```
final int MAJOR = 1000;
final int MINOR = 500;  //전직 시 더해질 값

final int WIZARD = 3;// 2차전직 법사
```

: 전직 시 주스탯 증가량과 부스탯 증가량
2차 전직 법사를 의미하는 상수 3

2) 생성자

```
public Adventurer () {  //생성자
    pAtk = PATK;
    mAtk = MATK;
    hp = HP;
    mp = MP;
    pDef = PDEF;
    mDef = MDEF;
    str = STR;
    con = CON;
    dex = DEX;
    agi = AGI;
    iq = IQ;
    men = MEN;
    level = LEVEL;
    reqExp = REQEXP;
    curExp = CUREXP;  // 스탯 초기값 세팅
}
```

:모험가 클래스의 생성자가 호출되면 기본스탯과 레벨, 경험치에 지정된 초기값들을 대입한다.

3) 인터페이스 구현

```
public interface Skill {
    public void attack ();
    public void qSkill ();
    public void wSkill ();
}
```

```
public class Adventurer implements Skill {
```

인터페이스 스킬의 메소드 프로토타입을 가져와 Adventurer 클래스에서 구현한다.

```java
@Override
public void attack() { System.out.printf("%10d - 평타\n", calcAttackDamage()); }   // 평타

 @Override
 public void qSkill() { System.out.printf("%10d - 돌팔매(원거리)\n", calcQuackDamage()); } //Q스킬

@Override
public void wSkill() { System.out.printf("%10d - 에너지 스트라이크(근거리)\n", calcEnergyStrikeDamage()); } //w스킬
```

평타와 각 스킬을 출력하고, 각 스킬의 calc메소드를 호출해 계산된 데미지를 출력한다.

- calc~

```java
public int calcAttackDamage () { return (int) (pAtk * (str + 0.3 * dex)); }

public int calcQuackDamage () { return (int) (pAtk * (0.6 * str + 0.3 * dex)); }

public int calcEnergyStrikeDamage () { return (int) (mAtk * (1.2 * iq) + pAtk * (0.5 * str)); }
```

설정된 기본 스탯을 가지고 계산한 데미지 값을 리턴한다.

4) 레벨업이나 전직으로 인해 스탯 상승이 이루어진다. 각각의 스탯과 경험치, 레벨을 업데이트하고 업데이트된 값을 불러오는 메소드가 필요하다.

- ~Update : 증가된 스탯값을 가져와 Adventurer 클래스의 스탯값에 더한다.

```java
public void pAtkUpdate(float pAtkAdd) { this.pAtk += pAtkAdd; }

public void mAtkUpdate(float mAtkAdd) { this.mAtk += mAtkAdd; }

public void HpUpdate(float hpAdd) { this.hp += hpAdd; }
```

- get ~ : 호출 시 스탯이나 레벨, 경험치 등의 값을 리턴한다.

```java
public float getMen() { return men; }       //스탯 받아오기

public int getLevel() { return level; }    //레벨 받아오기

public int getReqExp() { return reqExp; }    //필요 경험치 받아오기

public int getCurExp() { return curExp; }  //현재 경험치 받아오기
```

5) calcCharacterExp
: RoleplayingGame 클래스의 매 루프에서 매크로 동작이 끝날 때,
획득 경험치를 넘겨주며, wiz 객체의 calcCharacterExp 메소드를 호출한다.

```
wiz.calcCharcterExp(calcExps(), THIRD);
```

wiz 객체는 Adventurer 클래스로부터 상속을 받기 때문에 Adventurer 클래스의
calcCharacterExp 메소드도 사용할 수 있다.

```
public void calcCharcterExp (int gettingExps, int charNum) {

    curExp += gettingExps;

    while (curExp - reqExp > 0) {
        level++;

        incStat(charNum);

        curExp -= reqExp;
        reqExp *= 1.1;
    }
```

gettingExps는 획득한 경험치로 현재 경험치에 이 값을 더한다.
(현재 경험치 - 요구 경험치) 가 0보다 크다는 것은 요구 경험치를 모두 채웠다는
의미이므로 level 값을 증가시킨다.
각 직업마다 지정된 스탯증가량만큼의 스탯증가를 수행하는 incStat 메소드를 호출하며
RolePlayingGame 클래스로부터 넘겨받은 charNum값을 incStat 메소드의 매개변수로
넘겨준다.
레벨업을 하면 경험치는 0으로 초기화되야 하기 때문에
현재 경험치에서 요구 경험치를 뺀다.
그리고 레벨업마다 다음 레벨에 대한 요구 경험치는 10%씩 증가한다.

6) incStat

```
public void incStat (int charNum) {
    switch (charNum) {
        case WIZARD:
            incMagStat();
            break;
    }
}
```

전달받은 charNum은 상수값 WIZARD와 일치한다.
charNum에 해당하는 case문을 실행한다.
레벨업을 했을 때 마법사 스탯 증가량만큼의 스탯 증가가 이루어지는
incMagStat 메소드를 호출한다.

7) incMagStat

```
public void incMagStat () {
    hp += 100;
    mp += 10;

    str += 1;
    con += 1;
    dex += 1;
    agi += 1;
    iq += 5;
    men += 4;
}
```

마법사 직업의 지정된 스탯증가량만큼 스탯 증가가 이루어진다.

- Magician 클래스
: Adventurer 클래스로부터 상속을 받았다.
1) 생성자

```
public class Magician extends Adventurer {

    public Magician () {
        super();

        strUpdate(MINOR);
        conUpdate(MINOR);
        dexUpdate(MINOR);
        agiUpdate(MINOR);
        iqUpdate(MAJOR);
        menUpdate(MAJOR);
    }
```

부모인 Adventurer의 생성자를 호출한다.
1차전직 : 주스탯을 MAJOR만큼 증가, 부스탯을 MINOR만큼 증가시킨다.

2) 메소드

```
public int calcChainLightningDamage () { return (int) (getmAtk() * getIq() * 2.5); }

@Override
public void qSkill() { System.out.printf("%10d - 체인 라이트닝(원거리)\n", calcChainLightningDamage()); }

public int calcFrostNovaDamage () { return (int) (getmAtk() * getIq() * 0.25); }

@Override
public void wSkill() { System.out.printf("%10d - 프로스트 노바(광범위): 범위 슬로우\n", calcFrostNovaDamage()); }
```

: 부모 클래스의 인터페이스 메소드를 재정의하고, 갱신된 스탯으로 데미지를 계산한다.

2. 변수 선언

```
private int boost;
```

: 2차 전직 시 사용하게 되는 boost

3. 생성자

```
public Wizard () {
    super();

    strUpdate(MINOR);
    conUpdate(MINOR);
    dexUpdate(MINOR);
    agiUpdate(MINOR);
    iqUpdate(MAJOR);
    menUpdate(MAJOR);

    boost = 0;
}
```

부모인 Magician의 생성자를 호출한다.
2차전직 : 주스탯을 MAJOR만큼 증가, 부스탯을 MINOR만큼 증가시킨다.
boost를 0으로 초기화한다.

4. 메소드

```
public int calcSuperGravityFieldDamage () { return (int) (100 * getmAtk() * getIq() * 1.1); }

@Override
public void qSkill() {
    System.out.printf("%10d - 초중력(단일기 - boost 게이지 50 사용)\n",
            calcSuperGravityFieldDamage());
}

public int calcAltemaDamage () { return (int) (20 * getmAtk() * getIq() * 0.7); }

@Override
public void wSkill() {
    System.out.printf("%10d - 알테마(범위기 - boost 게이지 20 사용)\n",
            calcAltemaDamage());
}
```

: 부모 클래스의 인터페이스 메소드를 재정의하고, 갱신된 스탯으로 데미지를 계산한다.

5. printInfo

```
public void printInfo () {
    System.out.printf("hp: %d, mp: %d, " + "\n" +
            "str: %d, con: %d, dex: %d, agi: %d, iq: %d, men: %d, " + "\n" +
            "level: %d, exp: %d / %d\n",
            (int)getHp(), (int)getMp(),
            (int)getStr(), (int)getCon(), (int)getDex(),
            (int)getAgi(), (int)getIq(), (int)getMen(),
            getLevel(), getCurExp(), getReqExp());
}
```

현재 캐릭터의 정보를 출력한다.

- MacroSet 클래스

1. 변수 선언

```
// 전사 매크로
final int SECOND = 2;
// 법사 매크로
final int THIRD = 3;
```

각 직업의 매크로에 대한 상수값을 지정한다.

2. 생성자

```
public MacroSet () { System.out.println("매크로 구동 준비 완료!"); }
```

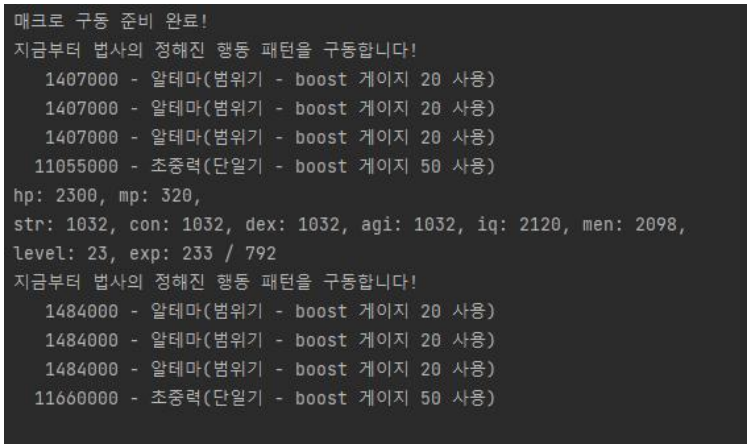
3. doMacroSet

```java
public void doMacroSet (final int num, Object obj) throws InterruptedException {
    // Object는 모든 데이터 타입을 받을 수 있는 가장 큰 집합임
    // 1. Object를 넘겨 받았으므로 전사 루틴을 처리할지 법사 루틴을 처리할지 결정하도록 만듬
    // 2. 루틴이 정해지면 각각에 맞게 매크로 루틴을 동작시킴
    switch (num) {
        case SECOND:
            System.out.println("지금부터 전사의 정해진 행동 패턴을 구동합니다!");
            Knight kni = (Knight) obj;
            // 각자 지정한 매크로에 따라 ~ 알아서 동작
            break;

        case THIRD:
            System.out.println("지금부터 법사의 정해진 행동 패턴을 구동합니다!");
            Wizard wiz = (Wizard) obj;
            for (int i = 0; i < 3; i++) {
                wiz.wSkill();
            }
            wiz.qSkill();
            Thread.sleep( millis: 3000);      // ms = 10^-3 s
            break;

        default:
            System.out.println("없는 매크로 입니다!");
            break;
    }
}
```

object는 모든 데이터 타입을 받을 수 있는 가장 큰 집합이므로
object를 사용해 객체 wizard를 넘긴다.
어떤 직업의 매크로를 돌릴지 결정하는 num값과 객체를 받아와
해당하는 직업의 매크로를 수행한다.
case THIRD
WIZARD의 W스킬을 2번, Q스킬을 한 번 호출한다.
Thread.sleep(millis:3000) : 3초 대기한다. sleep은 ms 단위로 계산된다.



-스탯-
모험가
- 지능, 정신 : 10,10
1차전직 마법사
- 지능, 정신 : 1010, 1010
2차전직 마법사
- 지능, 정신 : 2010, 2010

레벨업 시 incMagStat 메소드 : 지능 +5, 정신 +4
lv1 ~ lv 23까지 22레벨이 올랐으므로
지능iq : 2010 + (5*22) = 2120
정신men : 2010 + (4 *22) = 2098