



4000번의 도전  
드디어 찾아낸 반응 단백질



2012 인텔 국제과학경진대회 고든무어상 수상  
(Intel ISEF 2012, Gordon E. Moore Award)



# Java 프로그래밍 수업내용 정리 및 실습 과제

책을 보거나 인터넷을 찾아보면서 배우고 거기에 저만의 방식을 추가했어요.



**JACK ANDRAKA**  
IS CREATING A REVOLUTION



SAN ANTONIO SPURS  
처음에는 막막한 느낌이지만 시간이 지나면 감이 오거든요.



더하기 빼기... 그게 거의 전부죠.



저는 여러분에게 컴퓨터를 통해 어떻게  
세상을 바꿀 수 있는 지 이야기하고 싶습니다

그러나 사실, 구글이나 위키피디아 등을 이용해  
알고 싶은 모든 것을 찾을 수 있고

GOOGLE, WIKIPEDIA  
AND STACKOVERFLOW

2020.04.20.월

B반 송명훈



처음 배울때 감이 안나는게 뭐가 있을까요?



프로그래밍은 천재들만 할 수 있는것이 아니에요. 의지력이 필요할 뿐입니다.



15세 소년의 유일한 발명 도구는 “더 좋은 진단 키트를 만들고 싶었다.”  
인터넷(internet)  
- 잭 안드라카

# 목 차

1. 수업 내용 정리
2. 실습화면 캡처
3. 연습문제 10-1
4. 연습문제 10-2
5. 연습문제 10-3



## 1. 수업 내용 정리

# 1. 수업 내용 정리 [Encapsulation]

**캡슐화 (Encapsulation)** : field와 method를 하나의 단위(class)로 묶는 것

1. 객체지향프로그래밍에서의 중요한 특징 중 하나로, 연관된 데이터와 함수(메소드)를 논리적으로 묶는 것.  
데이터 보호를 위해 접근 지정자(access modifier)를 통해 다른 객체의 접근을 제한가능
2. class 내부에 존재하는 field의 값을 실수로 덮어쓰거나 잘못 조작하는 등의 human error를 미연에 방지하는 기능을 한다. ⇨ 정보 은닉
3. 정보 은닉 (Information Hiding)
  - 프로그램의 세부 구현을 외부로 드러나지 않도록 특정 모듈 내부로 감추는 것
  - 내부의 구현은 감추고 모듈 내에서의 응집도를 높이며, 외부로의 노출을 최소화하여 모듈 간의 결합도를 떨어뜨려 유연함과 유지보수성을 높이는 개념
  - 클래스 외부에서는 바깥으로 노출된 특정 메소드에만 접근이 가능하며 클래스 내부에서 어떤 식으로 처리가 이루어지는지는 알지 못하도록 설계됨
4. 접근 지정자(Access modifier)
  - public: 클래스의 외부에서 사용 가능하도록 노출시킴
  - protected: 다른 클래스에게는 노출되지 않지만, 상속받은 자식 클래스에게는 노출시킴
  - private: 클래스의 내부에서만 사용되며 외부로 노출되지 않음

# 1. 수업 내용 정리 [Encapsulation]

## 멤버에 대한 Access control

제한 범위	명칭	설정 방법	접근 가능 범위
제한이 엄격	private	private	자기 자신의 클래스
↓	package private (default)	(아무 것도 안 씀)	자신과 같은 패키지에 소속된 클래스
	protected	protected	자신과 같은 패키지에 소속되던지, 자신을 상속받은 자식 클래스
	public	public	모든 클래스

1. 멤버에 관한 액세스 지정의 정석
  - Field: 모두 private로 지정
  - Method: 모두 public으로 지정
2. Class에 대한 액세스 지정의 정석
  - 별다른 사유가 없는 경우 public으로 지정

# 1. 수업 내용 정리 [Encapsulation]

## Class에 대한 액세스 제어

이름	기술 방법	접근 가능한 범위	제한
package private	(아무 것도 쓰지 않음)	자신과 같은 패키지에 소속된 클래스	엄격
public	public	모든 클래스	느슨

- non-public Class의 특징 : Class의 이름은 소스 파일명과 달라도 됨, 1개의 소스파일에 여러 개의 class 선언 가능

## getter와 setter

1. method를 경유한 field 조작을 의미
  - getter method는 private field의 값을 획득. Ex] getName()
  - setter method는 private field의 값을 지정 Ex] setName(name)
2. 모든 field를 private로 지정 해 다른 클래스로부터 접근이 안되도록 막는다.
  - method를 통해서 접근하도록 class를 설계하는 것이 기본
3. getter / setter 이용 시 장점
  - Read Only, Write Only field의 실현 가능
  - field의 이름 등 클래스 내부 설계를 자유롭게 변경가능
  - field로의 접근을 검사 가능 (setter method의 경우 타당성 검사 등)





## 2. 실습화면 캡처

## 2. 실습 화면 캡처

### 10-1. 액세스 제어 (access control) 되어 있지 않는 프로그램의 예

```
public class Hero {
    static int money;

    String name;
    Sword sword;
    int hp;

    // 기본 생성자 (Constructor)
    Hero() { this( name: "김영웅"); // 두번째 생성자 호출 }

    // 생성자 오버로드
    Hero(String name) {
        hp = 100;
        this.name = name;
    }

    void bye() {
        System.out.println("용자는 이별을 고했다");
    }

    void die() {
        System.out.println(this.name + "는 죽었다");
        System.out.println("Game Over");
    }

    void run() {
        System.out.println(this.name + "는 도망쳤다!");
        System.out.println("GAME OVER");
    }
}
```

```
void run() {
    System.out.println(this.name + "는 도망쳤다!");
    System.out.println("GAME OVER");
    System.out.println("최종 HP는 " + this.hp + " 입니다");
}

void sit(int sec) {
    this.hp += sec; // 앉은 초 만큼 HP 가 증가
    System.out.println(this.name + "는 " + sec + "초 앉았다");
    System.out.println("HP가 " + sec + "포인트 회복되었다");
}

void slip() {
    this.hp -= 5;
    System.out.println(this.name + "는 넘어졌다");
    System.out.println("5의 데미지!");
}

void sleep() {
    this.hp = 100;
    System.out.println(this.name + "는 잠을 자고 회복했다.");
}

void attack() {
    System.out.println(this.name + "는 공격했다");
    System.out.println("적에게 5포인트의 데미지를 주었다");
}
}
```



## 2. 실습 화면 캡처

### 10-2. 여관 클래스의 오류

```
public class Inn {  
    void checkIn(Hero hero) {  
        hero.hp = -100;  
    }  
}
```

### 10-3. 왕 클래스의 문제점

```
public class King {  
    void talk(Hero hero) {  
        System.out.println("왕 : 우리 성에 어서오시오. 용사 "  
            + hero.name + "이여");  
        System.out.println("왕 : 긴 여행에 피로하겠군");  
        System.out.println("왕 : 우선 성 아랫 마을을 보고 와도 좋소. "  
            + "그럼 또 봅시다");  
        hero.die();  
    }  
}
```

### 10-4. HP를 private로 지정한 샘플 코드

```
public class Hero {  
    static int money;  
  
    private int hp;  
    String name;  
    Sword sword;  
  
    // 기본 생성자 (Constructor)  
    Hero() { this( name: "김영웅"); // 두번째 생성자 호출 }  
  
    // 생성자 오버로드  
    Hero(String name) {  
        hp = 100;  
        this.name = name;  
    }  
  
    void sleep() {  
        this.hp = 100;  
        System.out.println(this.name + "는 잠을 자고 회복했다.");  
    }  
}
```

## 2. 실습 화면 캡처

### 10-5. die() 메소드를 private로서 지정

```
public class Hero {  
    private void die() {  
        System.out.println(this.name + "는 죽었다");  
        System.out.println("Game Over");  
    }  
}
```

```
public class Hero {  
    private void die() {  
        System.out.println(this.name + "는 죽었다");  
        System.out.println("Game Over");  
    }  
  
    public void attack(Kinoko enemy) {  
        System.out.println(this.name + "의 공격!");  
        System.out.println("괴물 버섯 " + enemy.suffix);  
        this.hp -= 2;  
        if (this.hp <= 0 ) {  
            this.die();  
        }  
    }  
}
```

### 10-6. attack() 메소드는 public으로 지정

### 10-7. 왕 클래스에서 이용되는 name 필드

```
public class King {  
    void talk(Hero hero) {  
        System.out.println("왕 : 우리 성에 어서오시오. 용사 "  
            + hero.name + "이여");  
    }  
}
```

### 10-8. Hero 클래스에 getName 메소드를 추가

```
public class Hero {  
    private String name;  
  
    public String getName() {  
        return name;  
    }  
}
```

### 10-9. King 클래스의 talk() 메소드를 수정

```
public class King {  
    void talk(Hero hero) {  
        System.out.println("왕 : 우리 성에 어서오시오. 용사 "  
            + hero.getName() + "이여");  
    }  
}
```

## 2. 실습 화면 캡처

### 10-10. setter 메소드의 예

```
public class Hero {  
    private String name;  
  
    public String getName() {  
        return name;  
    }  
  
    public void setName(String name) {  
        this.name = name;  
    }  
}
```

### 10-11. 캡슐화를 하기 전과 후

```
public class Hero {  
    String name;  
}
```

```
public class Hero {  
    private String name;  
  
    public String getName() {  
        return name;  
    }  
  
    public void setName(String name) {  
        this.name = name;  
    }  
}
```

## 2. 실습 화면 캡처

### 10-13. setter 메소드 안에서 값의 타당성을 검사

```
public class Hero {
    private String name;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        if ( name == null ) {
            throw new IllegalArgumentException("이름은 null이 아니어야 함");
        }
        if ( name.length() <= 1 ) {
            throw new IllegalArgumentException("이름 너무 짧음");
        }
        if ( name.length() >= 8 ) {
            throw new IllegalArgumentException("이름 너무 길");
        }
        this.name = name;
    }
}
```

### 10-14. setName이 바르게 기능하는지 확인

```
public class GameMain {
    public static void main(String[] args) {
        Hero hero = new Hero();
        hero.setName("");
    }
}
```

GameMain ×

"C:\Program Files\Java\jdk-13.0.2\bin\java.exe" "-javaagent:C:\Program Files\Java\jdk-13.0.2\bin\javaagent.jar" -Djava.class.path=C:\Program Files\Java\jdk-13.0.2\bin\java.class.path -Djava.ext.dirs=C:\Program Files\Java\jdk-13.0.2\bin\java.ext.dirs Exception in thread "main" java.lang.IllegalArgumentException: 이름 너무 짧음  
at Game.Hero.setName(Hero.java:15)  
at Game.GameMain.main(GameMain.java:6)

Process finished with exit code 1



### 3. 연습문제 10-1

### 3. 연습문제 10-1

#### 10-1. 문제

- 다음 2개의 클래스 "Wizard(마법사)", "Wand(지팡이)"의 모든 필드와 메소드에 대해, 캡슐화의 정석에 따라 접근 지정자를 추가하시오. (Wizard 클래스의 컴파일 에러는 일단 무시하시오)

```
1 public class Wand {
2     String name;    // 지팡이의 이름
3     double power;   // 지팡이의 마력
4 }
```

```
1 public class Wizard {
2     int hp;
3     int mp;
4     String name;
5     Wand wand;
6
7     void heal(Hero hero) {
8         int basePoint = 10;    // 기본회복 포인트
9         int recovPoint = (int) (basePoint * this.wand.power);    // 지팡이에 의한 증폭
10        hero.setHp(hero.getHp() + recovPoint);    // 용사의 HP를 회복
11    }
```

### 3. 연습문제 10-1

```
public class Wand {  
    private String name;    // 지팡이의 이름  
    private double power;   // 지팡이의 마력  
  
    public String getName() {  
        return name;  
    }  
  
    public double getPower() {  
        return power;  
    }  
}
```

- 멤버에 관한 액세스 지정의 정석
  - ✓ 필드는 전부 private
  - ✓ 메소드는 전부 public
- 클래스에 대한 액세스 지정의 정석
  - ✓ 별다른 이유가 없으면 public

```
public class Wizard {  
    private int hp;  
    private int mp;  
    private String name;  
    private Wand wand;  
  
    public void heal(Hero hero) {  
        int basePoint = 10;    // 기본 회복 포인트  
        int recovPoint = (int) (basePoint * this.wand.getPower()); // 지팡이에 의한 증폭  
        hero.setHp(hero.getHp() + recovPoint); // 용사의 HP를 회복  
    }  
}
```





## 4. 연습문제 10-2

## 4. 연습문제 10-2

### 10-2. 문제 : 아래의 방침에 따라, 생성자를 추가 하시오

- 문제 10-1에서 작성한 Wand 클래스와 Wizard 클래스의 모든 필드에 대해, 정석에 따라 getter 메소드와 setter 메소드를 작성하시오.
- 그리고, Wizard 클래스의 heal 메소드에서 발생하는 컴파일 에러를 해결하시오.
- setter 메소드에 대해서 인수의 타당성 검사는 하지 않아도 됨

## 4. 연습문제 10-2 : Wizard class

```
package Game;

public class Wizard {
    // 모든 필드에 대해 정석에 따라 getter 메소드와 setter 메소드를 작성
    public final static int MAX_HP = 50;
    public final static int MAX_MP = 10;

    private int hp;
    private int mp;
    private String name;
    private Wand wand;

    // Constructor
    public Wizard(String name, int hp, int mp) {
        this.name = name;
        this.hp = hp;
        this.mp = mp;
    }

    // Constructor Overload 1
    public Wizard(String name, int hp) {
        this(name, hp, MAX_MP);
    }

    // Constructor Overload 2
    public Wizard(String name) {
        this(name, MAX_HP, MAX_MP);
    }
}
```

Wizard class ①

```
/** getHp() ...*/
public int getHp() {
    return hp;
}
```

②

```
/** getMp() ...*/
public int getMp() {
    return mp;
}
```

```
/** getName() ...*/
public String getName() {
    return name;
}
```

```
/** getWand() ...*/
public Wand getWand() {
    return wand;
}
```

// setter 메소드에 대해서 인수의 타당성 검사

```
/** setHp() ...*/
public void setHp(int hp) {
    this.hp = Math.max(mp, 0);
    this.hp = Math.min(MAX_HP, hp);
}
```

③

```
/** setMp() ...*/
public void setMp(int mp) {
    this.mp = Math.max(mp, 0);
    this.mp = Math.min(MAX_MP, mp);
}
```

```
/** setName() ...*/
public void setName(String name) {
    this.name = name;
}
```

```
/** setWand() ...*/
public void setWand(Wand wand) {
    this.wand = wand;
}
```

## 4. 연습문제 10-2 : Wizard & Wand class

// heal 메소드에서 발생하는 컴파일 에러 해결

/\*\* heal(Hero hero) ...\*/

```
public void heal(Hero hero) {
    int basePoint = 10;    // 기본 회복 포인트
    int recovPoint = (int) (basePoint * this.wand.getPower()); // 지팡이에 의한 증폭
    int decMp = 3;          // MP 소모량
    if (hero.getHp() == Hero.MAX_HP) {
        System.out.println(hero.getName() + "의 HP를 더 회복할 수 없다!");
        System.out.println("=====");
    } else if (getMp() < 3) {
        System.out.println(getName() + "의 MP가 부족하다!");
        System.out.println("=====");
    } else if (getMp() >= 3) {
        int beforeHp = hero.getHp();
        hero.setHp(Math.min(Hero.MAX_HP, hero.getHp() + recovPoint)); // 용사의 HP를 회복
        System.out.println(hero.getName() + "의 HP가 " + (hero.getHp() - beforeHp) + "회복!");
        System.out.println("=====");
        setMp(getMp() - decMp);
    }
}
```

Wizard class ④

Wand

```
public class Wand {
    private String name;    // 지팡이의 이름
    private double power;   // 지팡이의 마력

    // Constructor
    public Wand(String name, double power) {
        this.name = name;
        this.power = power;
    }

    /** getName() ...*/
    public String getName() {
        return name;
    }

    /** getPower() ...*/
    public double getPower() {
        return power;
    }

    /** setName(String name) ...*/
    public void setName(String name) {
        this.name = name;
    }

    /** setPower(double power) ...*/
    public void setPower(double power) {
        this.power = power;
    }
}
```

## 4. 연습문제 10-2 : Heal method

### Heal method 에서 발생하는 컴파일 에러 해결

```
public void heal(Hero hero) {  
    int basePoint = 10;    // 기본 회복 포인트  
    int recovPoint = (int) (basePoint * this.wand.getPower()); // 지팡이에 의한 증폭  
    hero.setHp(hero.getHp() + recovPoint);    // 용사의 HP를 회복  
}
```

```
/** heal(Hero hero) ... */  
public void heal(Hero hero) {  
    int basePoint = 10;    // 기본 회복 포인트  
    int recovPoint = (int) (basePoint * this.wand.getPower()); // 지팡이에 의한 증폭  
    int decMp = 3;    // MP 소모량  
    if (hero.getHp() == hero.getMaxHp()) {  
        System.out.println(hero.getName() + "의 HP를 더 회복할 수 없다!");  
        System.out.println("=====");  
    } else if (getMp() < 3) {  
        System.out.println(getName() + "의 MP가 부족하다!");  
        System.out.println("=====");  
    } else if (getMp() >= 3) {  
        int beforeHp = hero.getHp();  
        hero.setHp(Math.min(hero.getMaxHp(), hero.getHp() + recovPoint)); // 용사의 HP를 회복  
        System.out.println(hero.getName() + "의 HP가 " + (hero.getHp() - beforeHp) + "회복!");  
        System.out.println("=====");  
        setMp(getMp() - decMp);  
    }  
}
```

해결 전



해결 후

### 컴파일 에러 발생 이유

- Hero class에 setter method인 setHp와 getter method인 getHp가 선언되지 않았기 때문
- 따라서 이를 해결하기 위해서는 Hero class에도 setter와 getter를 선언해야 한다.

## 4. 연습문제 10-2 : Hero class

### Hero class에 getter & setter method 선언

```
public class Hero {  
    public final static int MAX_HP = 100;  
    public static int money;  
  
    private String name;  
    private int hp;  
    private Sword sword;  
  
    // 기본 생성자 (Constructor)  
    public Hero() {  
        this( "김영웅"); // 두번째 생성자 호출  
    }  
  
    // 생성자 오버로드  
    public Hero(String name) {  
        hp = MAX_HP;  
        this.name = name;  
    }  
  
    /** getName() ...*/  
    public String getName() {  
        return name;  
    }  
}
```

Hero class ①

```
/** getHp() ...*/  
public int getHp() {  
    return hp;  
}
```

```
/** setName(String name) ...*/  
public void setName(String name) {  
    this.name = name;  
}
```

```
/** setHp(int hp) ...*/  
public void setHp(int hp) {  
    this.hp = Math.min(MAX_HP, hp);  
}
```

```
/** setMomey(int money) ...*/  
public void setMoney(int money) {  
    this.money = money;  
}
```

```
private void die() {  
    System.out.println(getName() + "는 죽었다");  
    System.out.println("Game Over");  
}
```

②

## 4. 연습문제 10-2 : Hero class

```
private void die() {  
    System.out.println(getName() + "는 죽었다");  
    System.out.println("Game Over");  
}  
  
public void attack(Kinoko enemy) {  
    System.out.println(this.name + "의 공격!");  
    System.out.println("괴물 버섯 " + enemy.suffix);  
    enemy.hp -= 2;  
}  
  
public void sleep() {  
    setHp(MAX_HP);  
    System.out.println(getName() + "는 잠을 자고 회복했다.");  
}  
  
public void bye() {  
    System.out.println("용자는 이별을 고했다");  
}
```

③

```
public void run() {  
    System.out.println(getName() + "는 도망쳤다!");  
    System.out.println("GAME OVER");  
    System.out.println("최종 HP는 " + getHp() + " 입니다");  
}  
  
public void sit(int sec) {  
    setHp(hp + sec); // 앓은 초 만큼 HP 가 증가  
    System.out.println(getName() + "는 " + sec + "초 앓았다");  
    System.out.println("HP가 " + sec + "포인트 회복되었다");  
}  
  
public void slip() {  
    setHp(hp - 5);  
    System.out.println(getName() + "는 넘어졌다");  
    System.out.println("5의 데미지!");  
    if (getHp() <= 0) {  
        die();  
    }  
}
```

④



## 4. 연습문제 10-2 : main method

### 결과 확인을 위한 메인 메소드 및 출력 결과 확인

```
public class GameMain {  
    static void printStatus(Hero hero) {  
        System.out.println(hero.getName() + "의 상태");  
        System.out.println("HP/maxHP: " + hero.getHp() + "/" + Hero.MAX_HP);  
        System.out.println("=====");  
    }  
  
    static void printStatus(Wizard wizard) {  
        System.out.println(wizard.getName() + "의 상태");  
        System.out.println("HP/maxHP: " + wizard.getHp() + "/" + Wizard.MAX_HP);  
        System.out.println("MP/maxMP: " + wizard.getMp() + "/" + Wizard.MAX_MP);  
        System.out.println("=====");  
    }  
  
    static void printStatus(Wand wand) {  
        System.out.println(wand.getName() + "의 정보");  
        System.out.println("Power: " + wand.getPower());  
        System.out.println("=====");  
    }  
}
```

```
public static void main(String[] args) {  
    Wizard wizard = new Wizard( name: "위자드");  
    Hero hero = new Hero( name: "히어로");  
    Wand wand = new Wand( name: "완드", power: 1.2);  
    wizard.setWand(wand);  
  
    printStatus(hero);  
    printStatus(wizard);  
    printStatus(wand);  
    int testCycle = 5;  
    for (int i = 0; i < testCycle; i++) {  
        if(i == 0) {  
            // hero hp == Hero.MAX_HP 일 때 heal 사용 test  
        } else {  
            hero.setHp(hero.getHp() - 15); // test를 위해 hp 감소  
            printStatus(hero);  
        }  
        System.out.println("heal 사용");  
        wizard.heal(hero);  
        printStatus(hero);  
        printStatus(wizard);  
    }  
}
```

# 4. 연습문제 10-2 : 출력 결과

## 결과 확인을 위한 메인 메소드 및 출력 결과 확인

```
히어로의 상태
HP/maxHP: 100/100
=====
위자드의 상태
HP/maxHP: 50/50
MP/maxMP: 10/10
=====
완드의 정보
Power: 1.2
=====
hero HP == MaxHP의 경우
heal 사용 히어로의 HP를 더 회복할 수 없다!
=====
히어로의 상태
HP/maxHP: 100/100
=====
위자드의 상태
HP/maxHP: 50/50
MP/maxMP: 10/10 MP 감소 없음
=====
히어로의 상태
HP/maxHP: 85/100 HP 감소
=====
heal 사용
히어로의 HP가 12회복!
```

```
heal 사용
히어로의 HP가 12회복!
=====
히어로의 상태
HP/maxHP: 97/100 HP 회복
=====
위자드의 상태
HP/maxHP: 50/50
MP/maxMP: 7/10 MP 감소
=====
히어로의 상태
HP/maxHP: 82/100
=====
heal 사용
히어로의 HP가 12회복!
=====
히어로의 상태
HP/maxHP: 94/100 HP 회복
=====
위자드의 상태
HP/maxHP: 50/50
MP/maxMP: 4/10 MP 감소
=====
히어로의 상태
HP/maxHP: 79/100
```

```
=====
heal 사용
히어로의 HP가 12회복!
=====
히어로의 상태
HP/maxHP: 91/100
=====
위자드의 상태
HP/maxHP: 50/50
MP/maxMP: 1/10 MP <= 3
=====
히어로의 상태
HP/maxHP: 76/100
=====
heal 사용
위자드의 MP가 부족하다!
=====
MP 부족
히어로의 상태
HP/maxHP: 76/100
=====
HP 회복 없음
위자드의 상태
HP/maxHP: 50/50
MP/maxMP: 1/10
=====
MP 감소 없음
=====
Process finished with exit code 0
```



## 5. 연습문제 10-3

## 5. 연습문제 10-3

### 10-3. 문제

- 문제 10-2에서 작성한 Wand클래스와 Wizard 클래스의 각 setter 메소드에 대해, 아래의 4가지 규칙에 따라 인수의 타당성 검사를 추가하십시오.
- 부정한 값이 설정 될 경우에는 "throw new IllegalArgumentException("에러메시지");"를 기술하고 프로그램을 중단시킵니다.
  1. 마법사나 지팡이의 이름은 null 일 수 없고, 반드시 3문자 이상이어야 한다
  2. 지팡이의 마력은 0.5 이상 100.0 이하여야 한다.
  3. 마법사의 지팡이는 null 일 수 없다.
  4. 마법사의 HP와 MP는 0 이상이어야 한다. 또한 HP가 음수가 되는 상황에서는 대신 0이 설정되도록 한다.

## 5. 연습문제 10-3 : SetValidation class

### SetValidation class 소스 코드

```
package MHUtils;

import Game.Hero;
import Game.Wand;
import Game.Wizard;

public class SetValidation {

    /**
     * Hero class의 setName method의 validation 수행
     * @param hero
     * @param name
     */
    public static void setNameValidate(Hero hero, String name) {
        if (name == null) {
            throw new IllegalArgumentException("이름은 null이 아니어야 함");
        }
        if (name.length() <= 1) {
            throw new IllegalArgumentException("이름 너무 짧음");
        }
        if (name.length() >= 8) {
            throw new IllegalArgumentException("이름 너무 길");
        }
    }
}
```

```
/**
 * Wizard class의 setName method의 validation 수행
 * @param wizard
 * @param name
 */
```

```
public static void setNameValidate(Wizard wizard, String name) {
    if (name == null) {
        throw new IllegalArgumentException("이름은 null이 아니어야 함");
    }
    if (name.length() < 3) {
        throw new IllegalArgumentException("이름은 3글자 이상이어야 함");
    }
}
```

```
/**
 * Wand class의 setName method의 validation 수행
 * @param wand
 * @param name
 */
public static void setNameValidate(Wand wand, String name) {
    if (name == null) {
        throw new IllegalArgumentException("지팡이의 이름은 null이 아니어야 함");
    }
}
```

```
/**
 * Wizard class setMp method의 validation 수행
 * @param wizard
 * @param mp
 */
public static void setMpValidate(Wizard wizard, int mp) {
    if (mp < 0) {
        throw new IllegalArgumentException("MP는 0 이상이어야 함");
    }
}
```

## 5. 연습문제 10-3 : SetValidation class

```
/**
 * Wizard의 setHp method의 validation 수행
 * @param wizard
 * @param hp
 */
public static void setHpValidate(Wizard wizard, int hp) {
    if (hp < 0) {
        throw new IllegalArgumentException("MP는 0 이상이어야 함");
    }
}
```

```
/**
 * Hero class의 setHp method의 validation 수행
 * @param hero
 * @param hp
 */
public static void setHpValidate(Hero hero, int hp) {
    if (hp < 0) {
        throw new IllegalArgumentException("MP는 0 이상이어야 함");
    }
}
```

```
/**
 * Wizard class의 setWand method의 validation 수행
 * @param wizard
 * @param wand
 */
public static void setWandValidate(Wizard wizard, Wand wand) {
    if (wand == null) {
        throw new IllegalArgumentException("지팡이는 null이 아니어야 함");
    }
}
```

```
/**
 * Wand class의 setPower method의 validation 수행
 * @param wand
 * @param power
 */
public static void setPowerValidate(Wand wand, double power) {
    if ((power > 100) || (power < 0.5)) {
        throw new IllegalArgumentException("지팡이의 마력은 0.5 이상 100.0 이하 이어야 함");
    }
}
```

## 5. 연습문제 10-3 : Wizard class

### Wizard class 소스 코드

```
public class Wizard {
    // 모든 필드에 대해 정석에 따라 getter 메소드와 setter 메소드를 작성
    public final static int MAX_HP = 50;
    public final static int MAX_MP = 10;

    private int hp;
    private int mp;
    private String name;
    private Wand wand;

    // Constructor
    public Wizard(String name, int hp, int mp) {
        this.name = name;
        this.hp = hp;
        this.mp = mp;
    }

    // Constructor Overload 1
    public Wizard(String name, int hp) {
        this(name, hp, MAX_MP);
    }

    // Constructor Overload 2
    public Wizard(String name) {
        this(name, MAX_HP, MAX_MP);
    }
}
```

```
/** getHp() ...*/
public int getHp() {
    return hp;
}

/** getMp() ...*/
public int getMp() {
    return mp;
}

/** getName() ...*/
public String getName() {
    return name;
}

/** getWand() ...*/
public Wand getWand() {
    return wand;
}

/** getMaxHp() ...*/
public int getMaxHp() {
    return MAX_HP;
}
```

```
/** setHp() ...*/
public void setHp(int hp) {
    SetValidation.setHpValidate( wizard: this, mp);
    this.hp = Math.max(mp, 0);
    this.hp = Math.min(MAX_HP, hp);
}
```

```
/** setMp() ...*/
public void setMp(int mp) {
    SetValidation.setMpValidate( wizard: this, mp);
    this.mp = Math.max(mp, 0);
    this.mp = Math.min(MAX_MP, mp);
}
```

```
/** setName() ...*/
public void setName(String name) {
    SetValidation.setNameValidate( wizard: this, name);
    this.name = name;
}
```

```
/** setWand() ...*/
public void setWand(Wand wand) {
    SetValidation.setWandValidate( wizard: this, wand);
    this.wand = wand;
}
```



## 5. 연습문제 10-3 : Wizard class

### Wizard class 소스 코드

```
// heal 메소드에서 발생하는 컴파일 에러 해결
/** heal(Hero hero) ...*/
public void heal(Hero hero) {
    int basePoint = 10;    // 기본 회복 포인트
    int recovPoint = (int) (basePoint * this.wand.getPower()); // 지팡이에 의한 증폭
    int decMp = 3;         // MP 소모량
    if (hero.getHp() == hero.getMaxHp()) {
        System.out.println(hero.getName() + "의 HP를 더 회복할 수 없다!");
        System.out.println("=====");
    } else if (getMp() < 3) {
        System.out.println(getName() + "의 MP가 부족하다!");
        System.out.println("=====");
    } else if (getMp() >= 3) {
        int beforeHp = hero.getHp();
        hero.setHp(Math.min(hero.getMaxHp(), hero.getHp() + recovPoint)); // 용사의 HP를 회복
        System.out.println(hero.getName() + "의 HP가 " + (hero.getHp() - beforeHp) + "회복!");
        System.out.println("=====");
        setMp(getMp() - decMp);
    }
}
```

## 5. 연습문제 10-3 : Wand class

### Wand class 소스 코드

```
public class Wand {  
    private String name;    // 지팡이의 이름  
    private double power;   // 지팡이의 마력  
  
    // Constructor  
    public Wand(String name, double power) {  
        this.name = name;  
        this.power = power;  
    }  
  
    /** getName() ...*/  
    public String getName() {  
        return name;  
    }  
  
    /** getPower() ...*/  
    public double getPower() {  
        return power;  
    }  
}
```

```
/** setName(String name) ...*/  
public void setName(String name) {  
    SetValidation.setNameValidate(wand: this, name);  
    this.name = name;  
}
```

```
/** setPower(double power) ...*/  
public void setPower(double power) {  
    SetValidation.setPowerValidate(wand: this, power);  
    this.power = power;  
}
```

## 5. 연습문제 10-3 : 출력 결과

### Error 메시지 확인을 위한 메인 메소드 및 출력 결과 - wand

```
public class GameMain {  
    static void printStatus(Hero hero) {...}  
    static void printStatus(Wizard wizard) {...}  
    static void printStatus(Wand wand) {...}  
  
    public static void main(String[] args) {  
        Wizard wizard = new Wizard( name: "위자드");  
        Hero hero = new Hero( name: "히어로");  
        Wand wand = new Wand( name: "완드", power: 1.2);  
        wizard.setWand(wand);  
  
        wizard.setWand(null);  
    }  
}
```

Wand에 null 값 저장

GameMain x

"C:\Program Files\Java\jdk-13.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA\bin\idea\_rt.jar" -Dfile.encoding=UTF-8

Exception in thread "main" java.lang.IllegalArgumentException: 지팡이는 null이 아니어야 함

at Game.Wizard.setWand(Wizard.java:149)

at Game.GameMain.main(GameMain.java:27)

IllegalArgumentException

출력 확인

Process finished with exit code 1

## 5. 연습문제 10-3 : 출력 결과

Error 메시지 확인을 위한 메인 메소드 및 출력 결과 - wand 이름

```
public class GameMain {  
    static void printStatus(Hero hero) {...}  
    static void printStatus(Wizard wizard) {...}  
    static void printStatus(Wand wand) {...}  
  
    public static void main(String[] args) {  
        Wizard wizard = new Wizard( name: "위자드");  
        Hero hero = new Hero( name: "히어로");  
        Wand wand = new Wand( name: "완드", power: 1.2);  
  
        wizard.setWand(wand);  
        wand.setName(null);  
    }  
}
```

wand instance 이름에 null 값 저장

GameMain ×

"C:\Program Files\Java\jdk-13.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ I

Exception in thread "main" java.lang.IllegalArgumentException: 지팡이의 이름은 null이 아니어야 함

at Game.Wand.setName(Wand.java:41)

at Game.GameMain.main(GameMain.java:27)

IllegalArgumentException

출력 확인

Process finished with exit code 1

## 5. 연습문제 10-3 : 출력 결과

### Error 메시지 확인을 위한 메인 메소드 및 출력 결과 - 마력

```
public class GameMain {  
    static void printStatus(Hero hero) {...}  
    static void printStatus(Wizard wizard) {...}  
    static void printStatus(Wand wand) {...}  
  
    public static void main(String[] args) {  
        Wizard wizard = new Wizard( name: "위자드");  
        Hero hero = new Hero( name: "히어로");  
        Wand wand = new Wand( name: "완드", power: 1.2);  
  
        wizard.setWand(wand);  
        wand.setPower(0.4); wand 마력에 범위 밖의 값 저장  
    }  
}
```

GameMain ×

"C:\Program Files\Java\jdk-13.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Commun

Exception in thread "main" java.lang.IllegalArgumentException: 지팡이의 마력은 0.5 이상 100.0 이하 이어야 함

at Game.Wand.setPower(Wand.java:54)

at Game.GameMain.main(GameMain.java:27)

IllegalArgumentException

출력 확인

Process finished with exit code 1

## 5. 연습문제 10-3 : 출력 결과

### Error 메시지 확인을 위한 메인 메소드 및 출력 결과 - HP

```
public class GameMain {  
    static void printStatus(Hero hero) {...}  
    static void printStatus(Wizard wizard) {...}  
    static void printStatus(Wand wand) {...}  
  
    public static void main(String[] args) {  
        Wizard wizard = new Wizard( name: "위자드");  
        Hero hero = new Hero( name: "히어로");  
        Wand wand = new Wand( name: "완드", power: 1.2);  
  
        wizard.setWand(wand);  
        wizard.setHp(-1);  
    }  
}
```

wizard instance의 HP에 -1 저장

GameMain ×

"C:\Program Files\Java\jdk-13.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBra

Exception in thread "main" java.lang.IllegalArgumentException: HP는 0 이상이어야 함

at Game.Wizard.setHp(Wizard.java:99)

at Game.GameMain.main(GameMain.java:27)

IllegalArgumentException

출력 확인

Process finished with exit code 1

## 5. 연습문제 10-3 : 출력 결과

### Error 메시지 확인을 위한 메인 메소드 및 출력 결과 - MP

```
public class GameMain {  
    static void printStatus(Hero hero) {...}  
    static void printStatus(Wizard wizard) {...}  
    static void printStatus(Wand wand) {...}  
  
    public static void main(String[] args) {  
        Wizard wizard = new Wizard( name: "위자드");  
        Hero hero = new Hero( name: "히어로");  
        Wand wand = new Wand( name: "완드", power: 1.2);  
  
        wizard.setWand(wand);  
        wizard.setMp(-1);  
    }  
}
```

GameMain ×

"C:\Program Files\Java\jdk-13.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBra

Exception in thread "main" java.lang.IllegalArgumentException: MP는 0 이상이어야 함  
 at Game.Wizard.setMp(Wizard.java:116)  
 at Game.GameMain.main(GameMain.java:27)

Process finished with exit code 1

wizard instance의 MP에 -1 저장

IllegalArgumentException

출력 확인



## 5. 연습문제 10-3 : 출력 결과

Error 메시지 확인을 위한 메인 메소드 및 출력 결과 - wizard instance name

```
public class GameMain {  
    static void printStatus(Hero hero) {...}  
    static void printStatus(Wizard wizard) {...}  
    static void printStatus(Wand wand) {...}  
  
    public static void main(String[] args) {  
        Wizard wizard = new Wizard( name: "위자드");  
        Hero hero = new Hero( name: "히어로");  
        Wand wand = new Wand( name: "완드", power: 1.2);  
  
        wizard.setWand(wand);  
        wizard.setName("법사"); wizard instance의 이름에 2글자 저장  
    }  
}
```

GameMain ×

```
"C:\Program Files\Java\jdk-13.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\I  
Exception in thread "main" java.lang.IllegalArgumentException: 이름은 3글자 이상이어야 함  
    at Game.Wizard.setName(Wizard.java:136)  
    at Game.GameMain.main(GameMain.java:27)
```

IllegalArgumentException

출력 확인

Process finished with exit code 1

## 5. 연습문제 10-3 : 출력 결과

Error 메시지 확인을 위한 메인 메소드 및 출력 결과 - wizard instance name

```
public class GameMain {  
    static void printStatus(Hero hero) {...}  
    static void printStatus(Wizard wizard) {...}  
    static void printStatus(Wand wand) {...}  
  
    public static void main(String[] args) {  
        Wizard wizard = new Wizard( name: "위자드");  
        Hero hero = new Hero( name: "히어로");  
        Wand wand = new Wand( name: "완드", power: 1.2);  
  
        wizard.setWand(wand);  
        wizard.setName("마법사");  
    }  
}
```

wizard instance의  
이름에 3글자 저장

GameMain ×

"C:\Program Files\Java\jdk-13.0.2\bin\java.exe" "-javaage

Exception 미출력 확인

Process finished with exit code 0