

[연습문제 - 모범답안]

[6-1] 다음과 같은 멤버변수를 갖는 SutdaCard클래스를 정의하시오.

타입	변수명	설명
int	num	카드의 숫자.(1~10사이의 정수)
boolean	isKwang	광(光)이면 true, 아니면 false

[정답]

```
class SutdaCard {  
    int num;  
    boolean isKwang;  
}
```

[6-2] 문제6-1에서 정의한 SutdaCard클래스에 두 개의 생성자와 info()를 추가해서 실행 결과와 같은 결과를 얻도록 하시오.

[연습문제]/ch6/Exercise6_2.java

```
class Exercise6_2 {
    public static void main(String args[]) {
        SutdaCard card1 = new SutdaCard(3, false);
        SutdaCard card2 = new SutdaCard();

        System.out.println(card1.info()); // 3이 출력된다.
        System.out.println(card2.info()); // 1K가 출력된다.
    }
}

class SutdaCard {
    int num;
    boolean isKwang;

    SutdaCard() {
        this(1, true); // SutdaCard(1, true)를 호출한다.
    }

    SutdaCard(int num, boolean isKwang) {
        this.num = num;
        this.isKwang = isKwang;
    }

    String info() { // 숫자를 문자열로 반환한다. 광(光)인 경우 K를 덧붙인다.
        return num + (isKwang? "K" : "");
    }
}
```

[실행결과]

```
3
1K
```

[해설] 객체를 생성할 때 두 개의 생성자를 사용했으므로 두 개의 생성자를 정의해야 한다.

```
SutdaCard card1 = new SutdaCard(3, false); // 3
SutdaCard card2 = new SutdaCard();           // 1K
```

일단 매개변수가 있는 생성자를 살펴보면, 카드의 num과 isKwang의 값을 매개변수로 받는 것을 알 수 있다. 그리고 매개변수가 없는 기본 생성자는 실행결과에서 "1K"가 출력된 것으로 봐서 num과 isKwang의 값을 각각 1과 true로 하였다는 것을 알 수 있다.

```

SutdaCard() {
    this.num = 1;
    this.isKwang = true;
}

SutdaCard(int num, boolean isKwang)
{
    this.num = num;
    this.isKwang = isKwang;
}

SutdaCard() {
    this(1, true);
}

SutdaCard(int num, boolean isKwang)
{
    this.num = num;
    this.isKwang = isKwang;
}

```

매개변수가 없는 기본 생성자를 정의할 때, 왼쪽의 코드와 같이 할 수도 있지만 오른쪽 코드와 같이 기존의 코드를 호출하는 것이 더 좋은 코드이다. 재사용성이 더 높고 나중에 코드를 수정할 때도 유리하다.

`info()`메서드는 `card`인스턴스의 정보를 문자열로 반환하기 위한 것이다. `card`인스턴스의 멤버변수 `num`과 `isKwang`의 값을 문자열로 만들어서 반환하면 된다. `isKwang`의 값이 `true`인 경우에는 숫자 뒤에 "K"를 붙이도록 삼항연산자를 사용했다.

```

String info() { // 숫자를 문자열로 반환한다. 광(光)인 경우 K를 덧붙인다.
    return num + ( isKwang? "K" : "" );
}

```

변수 `num`은 타입이 `int`이지만 문자열과 덧셈연산을 하기 때문에 최종적으로는 문자열을 반환하게 된다.

[6-3] 다음과 같은 멤버변수를 갖는 Student클래스를 정의하시오.

타입	변수명	설명
String	name	학생이름
int	ban	반
int	no	번호
int	kor	국어점수
int	eng	영어점수
int	math	수학점수

[정답]

```
class Student {  
    String name;  
    int ban;  
    int no;  
    int kor;  
    int eng;  
    int math;  
}
```

[6-4] 문제6-3에서 정의한 Student클래스에 다음과 같이 정의된 두 개의 메서드 getTotal()과 getAverage()를 추가하시오.

1. 메서드명 : getTotal

기 능 : 국어(kor), 영어(eng), 수학(math)의 점수를 모두 더해서 반환한다.

반환타입 : int

매개변수 : 없음

2. 메서드명 : getAverage

기 능 : 총점(국어점수+영어점수+수학점수)을 과목수로 나눈 평균을 구한다.

소수점 둘째자리에서 반올림할 것.

반환타입 : float

매개변수 : 없음

【연습문제】/ch6/Exercise6_4.java

```
class Exercise6_4 {
    public static void main(String args[]) {
        Student s = new Student();
        s.name = "홍길동";
        s.ban = 1;
        s.no = 1;
        s.kor = 100;
        s.eng = 60;
        s.math = 76;

        System.out.println("이름:"+s.name);
        System.out.println("총점:"+s.getTotal());
        System.out.println("평균:"+s.getAverage());
    }
}

class Student {
    String name;
    int ban;
    int no;
    int kor;
    int eng;
    int math;

    int getTotal() {
        return kor + eng + math;
    }

    float getAverage() {
        return (int)(getTotal() / 3f * 10 + 0.5f) / 10f;
    }
}
```

【실행결과】

이름:홍길동

총점:236

평균:78.7

【정답】

```
class Student {
```

```

String name;
int ban;
int no;
int kor;
int eng;
int math;

int getTotal() {
    return kor + eng + math;
}

float getAverage() {
    return (int)(getTotal() / 3f * 10 + 0.5f) / 10f;
}
}

```

[해설] 총점과 평균을 구하는 문제인데, 평균을 구할 때 소수점 둘째 자리에서 반올림을 하는 부분에서 생각을 좀 해야 할 것이다.

총점의 타입이 int이기 때문에 3으로 나누면 int와 int간의 연산이므로 결과를 int로 얻는다. 즉, 소수점 이하의 값은 버려지게 된다. 그래서 float타입의 리터럴인 3f로 나누어야 소수점 이하의 값을 얻을 수 있다. 그리고, 소수점 둘째 자리에서 반올림하려면 10을 곱하고 0.5를 더한 다음 다시 10f로 나누면 된다.

```

236 / 3 → 78
236 / 3f → 78.666664
236 / 3f * 10 → 786.66664
236 / 3f * 10 + 0.5 → 787.16664
(int)(236 / 3f * 10 + 0.5) → (int)787.16664 → 787
(int)(236 / 3f * 10 + 0.5) / 10 → 78
(int)(236 / 3f * 10 + 0.5) / 10f → 78.7

```

[6-5] 다음과 같은 실행결과를 얻도록 Student클래스에 생성자와 info()를 추가하시오.

【연습문제】/ch6/Exercise6_5.java

```
class Exercise6_5 {
    public static void main(String args[]) {
        Student s = new Student("홍길동",1,1,100,60,76);

        System.out.println(s.info());
    }
}

class Student {
    String name;
    int ban;
    int no;
    int kor;
    int eng;
    int math;

    Student(String name, int ban, int no, int kor, int eng, int math) {
        this.name = name;
        this.ban = ban;
        this.no = no;
        this.kor = kor;
        this.eng = eng;
        this.math = math;
    }

    int getTotal() {
        return kor+eng+math;
    }

    float getAverage() {
        return (int)(getTotal() / 3f * 10 + 0.5f) / 10f;
    }

    public String info() {
        return name
            +", "+ban
            +", "+no
            +", "+kor
            +", "+eng
            +", "+math
            +", "+getTotal()
            +", "+getAverage()
            ;
    }
}
```

【실행결과】

홍길동,1,1,100,60,76,236,78.7

[해설] 학생의 이름, 반, 번호, 과목별 성적을 매개변수로 받는 생성자를 추가하고, 학생의 정보를 출력하는 info()메서드를 정의하는 문제이다. 답을 보는 것만으로도 충분히 이해할 수 있는 문제이므로 설명은 생략한다.

[6-6] 두 점의 거리를 계산하는 `getDistance()`를 완성하시오.

[Hint] 제곱근 계산은 `Math.sqrt(double a)`를 사용하면 된다.

[연습문제]/ch6/Exercise6_6.java

```
class Exercise6_6 {
    // 두 점 (x, y)와 (x1, y1)간의 거리를 구한다.
    static double getDistance(int x, int y, int x1, int y1) {
        return Math.sqrt((x-x1)*(x-x1) + (y-y1)*(y-y1)); // x, y는 지역변수
    }

    public static void main(String args[]) {
        System.out.println(getDistance(1,1,2,2));
    }
}
```

[실행결과]

```
1.4142135623730951
```

[정답]

```
return Math.sqrt((x-x1)*(x-x1) + (y-y1)*(y-y1));
```

[해설] 두 점 (x, y) 와 $(x1, y1)$ 의 거리를 구하는 공식은 $\sqrt{(x-x1)^2 + (y-y1)^2}$ 이다.

제곱근 계산은 `Math` 클래스의 `sqrt(double a)`를 사용하면 된다. 제곱도 `Math.pow(double a, double b)`를 사용하면 되지만, 2제곱이므로 그냥 곱셈연산자를 사용했다. 어느 쪽을 사용해도 괜찮지만, 메서드를 호출하는 것은 곱셈연산보다 비용이 많이 드는 작업이라는 것은 기억해두자. 그렇다고 해서 보다 빠른 코드를 만들겠다고 코드를 복잡하게 하는 것은 좋지 않다.

참고로 `Math.pow(double a, double b)`를 사용한 코드는 다음과 같다.

```
static double getDistance(int x, int y, int x1, int y1) {
    return Math.sqrt(Math.pow(x-x1,2) + Math.pow(y-y1,2));
}
```

[6-7] 문제6-6에서 작성한 클래스메서드 `getDistance()`를 `MyPoint`클래스의 인스턴스메서드로 정의하시오.

[연습문제]/ch6/Exercise6_7.java

```
class MyPoint {
    int x; // 인스턴스 변수
    int y; // 인스턴스 변수

    MyPoint(int x, int y) {
        this.x = x;
        this.y = y;
    }

    double getDistance(int x1, int y1) {
        return Math.sqrt((x-x1)*(x-x1) + (y-y1)*(y-y1)); // x, y는 인스턴스 변수
    }
}

class Exercise6_7 {
    public static void main(String args[]) {
        MyPoint p = new MyPoint(1,1);

        // p와 (2,2)의 거리를 구한다.
        System.out.println(p.getDistance(2,2));
    }
}
```

[실행결과]

1.4142135623730951

[정답]

```
double getDistance(int x1, int y1) {
    return Math.sqrt((x-x1)*(x-x1) + (y-y1)*(y-y1));
}

[해설] 이전 문제의 static메서드를 인스턴스 메서드로 변경하는 문제인데, static메서드와 인스턴스 메서드의 차이를 이해하는 것은 매우 중요하다. static메서드인 경우에는 메서드 내에서 인스턴스 변수를 사용하지 않았다. 대신 매개변수(지역변수)로 작업에 필요한 값을 제공받아야했다. 그래서, 인스턴스와 관계가 없으므로(인스턴스변수를 사용 안했으니까) static메서드로 선언할 수 있는 것이다.
```

```
static double getDistance(int x, int y, int x1, int y1) {
    return Math.sqrt((x-x1)*(x-x1) + (y-y1)*(y-y1)); // x, y는 지역변수
}
```

그러나, 인스턴스 메서드는 인스턴스 변수 `x`, `y`를 사용해서 작업하므로 매개변수로 `x1`과 `y1`만을 제공받으면 된다. 인스턴스와 관계가 있으므로(인스턴스 변수를 사용했으니까) `static`을 붙일 수 없다.

```
double getDistance(int x1, int y1) {
    return Math.sqrt((x-x1)*(x-x1) + (y-y1)*(y-y1)); // x, y는 인스턴스 변수
}
```

아래의 코드는 인스턴스 메서드를 사용할 때와 static메서드를 사용할 때의 차이를 보여주기 위한 것이다. 어떤 차이가 있는지 잘 살펴보자.

1. static메서드의 사용

```
System.out.println(Exercise6_6.getDistance(1,1,2,2));
```

2. 인스턴스 메서드의 사용

```
MyPoint p = new MyPoint(1,1);
System.out.println(p.getDistance(2,2));
```

MyPoint클래스에 두 점간의 거리를 계산하는 메서드 getDistance()를 넣는다면, static메서드 보다는 인스턴스메서드로 정의하는 것이 더 적합하다.

[6-8] 다음의 코드에 정의된 변수들을 종류별로 구분해서 적으시오.

```
class PlayingCard {
    int kind;
    int num;

    static int width;
    static int height;

    PlayingCard(int k, int n) {
        kind = k;
        num = n;
    }

    public static void main(String args[]) {
        PlayingCard card = new PlayingCard(1,1);
    }
}
```

- 클래스변수(static변수) : width, height
- 인스턴스변수 : kind, num
- 지역변수 : k, n, card, args

[해설] 변수가 선언된 위치를 보면 변수의 종류를 알 수 있다. 클래스 블럭{}내에 선언된 변수는 인스턴스 변수이고, static이 붙은 것은 static변수(클래스 변수)이다. 그리고 나머지는 모두 지역 변수이다.

```
class Variables
{
    int iv;           // 인스턴스변수
    static int cv;    // 클래스변수(static변수, 공유변수)

    void method()
    {
        int lv = 0;    // 지역변수
    }
}
```

클래스영역

메서드영역

변수의 종류	선언위치	생성시기
클래스변수 (class variable)	클래스 영역	클래스가 메모리에 올라갈 때
인스턴스변수 (instance variable)		인스턴스가 생성되었을 때
지역변수 (local variable)	클래스 영역 이외의 영역 (메서드, 생성자, 초기화 블럭 내부)	변수 선언문이 수행되었을 때

[6-9] 다음은 컴퓨터 게임의 병사(marine)를 클래스로 정의한 것이다. 이 클래스의 멤버 중에 static을 붙여야 하는 것은 어떤 것들이고 그 이유는 무엇인가?
 (단, 모든 병사의 공격력과 방어력을 같아야 한다.)

```
class Marine {
    int x=0, y=0;           // Marine의 위치좌표(x,y)
    int hp = 60;            // 현재 체력
    static int weapon = 6;   // 공격력
    static int armor = 0;    // 방어력

    static void weaponUp() {
        weapon++;
    }

    static void armorUp() {
        armor++;
    }

    void move(int x, int y) {
        this.x = x; // this.x는 인스턴스 변수, x는 지역변수
        this.y = y; // this.y는 인스턴스 변수, y는 지역변수
    }
}
```

[정답]

weapon, armor - 모든 Marine인스턴스에 대해 동일한 값이어야 하므로.
weaponUp(), armorUp() - static변수에 대한 작업을 하는 메서드이므로

[해설] 인스턴스마다 개별적인 값을 가져야하는 변수는 인스턴스변수로, 모든 인스턴스가 공통적인 값을 가져야하는 변수는 클래스 변수(static변수)로 선언해야한다.

그래서 위의 코드에서 어떤 변수들이 모든 인스턴스에서 공통적인 적인 값을 가져야하는지 알아내야한다.

병사(marin)의 위치는 모든 병사가 서로 다른 위치에 있어야 하므로 개별적인 값이어야하고, 병사들마다 부상의 정도가 다를 것이므로 병사들의 체력(hp) 역시 개별적인 값이어야 한다. 그러나 모든 병사들의 공격력과 방어력을 같아야 한다.(게임이니까)

그래서 공격력을 의미하는 변수 weapon과 방어력을 의미하는 변수 armor에 static을 붙어야한다.

그 다음은 메서드인데, 어떤 메서드에 static을 붙이고 어떤 메서드에는 static을 붙이지 않아야하는 것일까?

메서드는 어떠한 작업을 하는 것인데, 이 작업을 할 때 인스턴스변수를 사용하면 인스턴스 메서드로 하고, 그렇지 않으면 static메서드로 하면 된다. 보통 인스턴스메서드는 인스턴스변수와 관련된 작업을 하고, static메서드는 static변수와 관련된 작업을 하기 때문이다.

메서드 weaponUp()과 armorUp()은 각각 static변수 weapon과 armor를 가지고 작업을 하기 때문에 static을 붙이는 것이 맞다. 반면에 메서드 move(int x, int y)는 인스턴스변수 x와 y를 가지고 작업하기 때문에 static을 붙여서는 안 된다.

[6-10] 다음 중 생성자에 대한 설명으로 옳지 않은 것은? (모두 고르시오)

- a. 모든 생성자의 이름은 클래스의 이름과 동일해야한다.
- b. 생성자는 객체를 생성하기 위한 것이다.
- c. 클래스에는 생성자가 반드시 하나 이상 있어야 한다.
- d. 생성자가 없는 클래스는 컴파일러가 기본 생성자를 추가한다.
- e. 생성자는 오버로딩 할 수 없다.

[정답] b, e

[해설]

b. 생성자는 객체를 생성하기 위한 것이다.

→ 생성자가 객체를 생성할 때 사용되기는 하지만, 객체를 초기화할 목적으로 사용되는 것이다. 객체를 생성하는 것은 new연산자이다.

e. 생성자는 오버로딩 할 수 없다.

→ 생성자도 오버로딩이 가능해서 하나의 클래스에 여러 개의 생성자를 정의할 수 있다.

[6-11] 다음 중 this에 대한 설명으로 맞지 않은 것은? (모두 고르시오)

- a. 객체 자신을 가리키는 참조변수이다.
- b. 클래스 내에서라면 어디서든 사용할 수 있다.** → 인스턴스메서드에서만 사용가능
- c. 지역변수와 인스턴스변수를 구별할 때 사용한다.
- d. 클래스 메서드 내에서는 사용할 수 없다.

[정답] b

[해설]

b. 클래스 내에서라면 어디서든 사용할 수 있다.

→ 클래스 멤버(static이 붙은 변수나 메서드)에는 사용할 수 없다.

this는 인스턴스 자신의 주소를 저장하고 있으며, 모든 인스턴스메서드에 숨겨진 채로 존재하는 지역변수이다. 그래서 인스턴스메서드 내에서만 사용할 수 있다.

[6-12] 다음 중 오버로딩이 성립하기 위한 조건이 아닌 것은? (모두 고르시오)

- a. 메서드의 이름이 같아야 한다.
- b. 매개변수의 개수나 타입이 달라야 한다.
- c. 리턴타입이 달라야 한다.
- d. 매개변수의 이름이 달라야 한다.

[정답] c, d

[해설]

- c. 리턴타입이 달라야 한다.
→ 리턴타입은 오버로딩에 영향을 주지 못한다.
- d. 매개변수의 이름이 달라야 한다.
→ 리턴타입은 오버로딩에 영향을 주지 못한다.

<< 오버로딩의 조건 >>

1. 메서드 이름이 같아야 한다.
2. 매개변수의 개수 또는 타입이 달라야 한다.
3. 매개변수는 같고 리턴타입이 다른 경우는 오버로딩이 성립되지 않는다.
(리턴타입은 오버로딩을 구현하는데 아무런 영향을 주지 못한다.)

[6-13] 다음 중 아래의 add메서드를 올바르게 오버로딩 한 것은? (모두 고르시오)

```
long add(int a, int b) { return a+b; }
```

- a. long add(int x, int y) { return x+y; }
- b. long add(long a, long b) { return a+b; }
- c. int add(byte a, byte b) { return a+b; }
- d. int add(long a, int b) { return (int)(a+b); }

[정답] b, c, d

[해설] b, c, d는 모두 메서드의 이름이 add이고 매개변수의 타입이 다르므로 오버로딩이 성립한다. 오버로딩이 성립하기 위한 조건은 다음과 같다.

<< 오버로딩의 조건 >>

1. 메서드 이름이 같아야 한다.
2. 매개변수의 개수 또는 타입이 달라야 한다.
3. 매개변수는 같고 리턴타입이 다른 경우는 오버로딩이 성립되지 않는다.
(리턴타입은 오버로딩을 구현하는데 아무런 영향을 주지 못한다.)

[6-14] 다음 중 초기화에 대한 설명으로 옳지 않은 것은? (모두 고르시오)

- a. 멤버변수는 자동 초기화되므로 초기화하지 않고도 값을 참조할 수 있다.
- b. 지역변수는 사용하기 전에 반드시 초기화해야 한다.
- c. 초기화 블럭보다 생성자가 먼저 수행된다. → 초기화 블럭이 먼저 수행된다.
- d. 명시적 초기화를 제일 우선적으로 고려해야 한다.
- e. 클래스변수보다 인스턴스변수가 먼저 초기화된다. → 클래스변수가 먼저 초기화됨

[정답] c, e

[해설] 클래스변수는 클래스가 처음 메모리에 로딩될 때, 자동 초기화되므로 인스턴스 변수보다 먼저 초기화 된다. 그리고 생성자는 초기화 블럭이 수행된 다음에 수행된다.

[6-15] 다음중 인스턴스변수의 초기화 순서가 올바른 것은?

- a. 기본값-명시적초기화-초기화블럭-생성자
- b. 기본값-명시적초기화-생성자-초기화블럭
- c. 기본값-초기화블럭-명시적초기화-생성자
- d. 기본값-초기화블럭-생성자-명시적초기화

[정답] a

[해설] 변수의 초기화 순서는 다음과 같다.

클래스변수의 초기화시점 : 클래스가 처음 로딩될 때 단 한번 초기화 된다.

인스턴스변수의 초기화시점 : 인스턴스가 생성될 때마다 각 인스턴스별로 초기화가 이루어진다.

클래스변수의 초기화순서 : 기본값 → 명시적초기화 → 클래스 초기화 블럭

인스턴스변수의 초기화순서 : 기본값 → 명시적초기화 → 인스턴스 초기화 블럭 → 생성자

[6-16] 다음 중 지역변수에 대한 설명으로 옳지 않은 것은? (모두 고르시오)

- a. 자동 초기화되므로 별도의 초기화가 필요없다.
- b. 지역변수가 선언된 메서드가 종료되면 지역변수도 함께 소멸된다.
- c. 매서드의 매개변수로 선언된 변수도 지역변수이다.
- d. 클래스변수나 인스턴스변수보다 메모리 부담이 적다.
- e. 힙(heap)영역에 생성되며 가비지 컬렉터에 의해 소멸된다.

[정답] a, e

[해설] 지역변수는 자동 초기화 되지 않기 때문에 사용하기 전에 반드시 적절한 값으로 초기화를 해주어야한다. 지역변수는 자신이 선언된 블럭이나 메서드가 종료되면 소멸되므로 메모리 부담이 적다. 힙(heap)영역에는 인스턴스(인스턴스변수)가 생성되는 영역이며, 지역변수는 호출스택(call stack)에 생성된다.

[6-17] 호출스택이 다음과 같은 상황일 때 옳지 않은 설명은? (모두 고르시오)

println
method1
method2
main

- a. 제일 먼저 호출스택에 저장된 것은 main메서드이다.
- b. println메서드를 제외한 나머지 메서드들은 모두 종료된 상태이다.
- c. method2메서드를 호출한 것은 main메서드이다.
- d. println메서드가 종료되면 method1메서드가 수행을 재개한다.
- e. main-method2-method1-println의 순서로 호출되었다.
- f. 현재 실행중인 메서드는 println 뿐이다.

[정답] b

[해설] 호출스택의 제일 위에 있는 메서드가 현재 수행중인 메서드이며, 호출스택 안의 나머지 메서드들은 대기상태이다.

[6-18] 다음의 코드를 컴파일하면 에러가 발생한다. 컴파일 에러가 발생하는 라인과 그 이유를 설명하시오.

```

class MemberCall {
    int iv = 10;
    static int cv = 20;

    int iv2 = cv;
    static int cv2 = iv;           // 라인 A - 컴파일 에러

    static void staticMethod1() {
        System.out.println(cv);
        System.out.println(iv);   // 라인 B - 컴파일 에러
    }

    void instanceMethod1() {
        System.out.println(cv);
        System.out.println(iv);   // 라인 C
    }

    static void staticMethod2() {
        staticMethod1();
        instanceMethod1();        // 라인 D - 컴파일 에러
    }

    void instanceMethod2() {
        staticMethod1();          // 라인 E
        instanceMethod1();
    }
}

```

[정답] 라인 A, 라인 B, 라인 D

[해설] 라인 A - static변수의 초기화에 인스턴스변수를 사용할 수 없다.

꼭 사용해야한다면, 객체를 생성해야한다.

라인 B - static에서드에서는 인스턴스변수를 사용할 수 없다.

라인 D - static에서드에서는 인스턴스에서드를 사용할 수 없다.

[6-19] 다음 코드의 실행 결과를 예측하여 적으시오.

[연습문제]/ch6/Exercise6_19.java

```
class Exercise6_19
{
    public static void change(String str) {
        str += "456";
    }

    public static void main(String[] args)
    {
        String str = "ABC123";
        System.out.println(str);
        change(str);
        System.out.println("After change:"+str);
    }
}
```

[정답]

[실행결과]

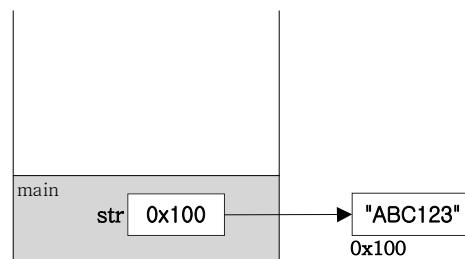
```
ABC123
After change:ABC123
```

[해설] change메서드의 매개변수가 참조형인데도 왜? main에서드의 문자열 str에 변경한 내용이 반영되지 않은 것일까? 많은 사람들이 매개변수가 참조형이라는 것만 보고 main에서드의 문자열 str이 변경될 것이라고 쉽게 생각한다. 누구라도 실수하기 쉬운 부분이므로 주의하길 바라는 마음에서 이 문제를 만들었다.

그림과 함께 단계 별로 설명하면 어렵지 않게 이해할 수 있을 것이다.

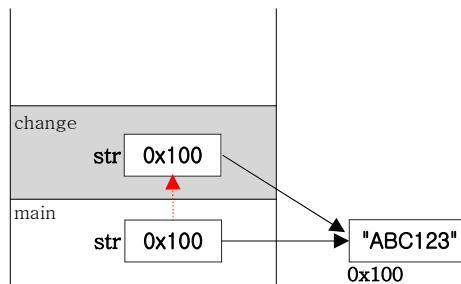
처음에 문자열을 참조변수 str에 저장하면 아래와 같은 그림이 된다.

```
String str = "ABC123";
```



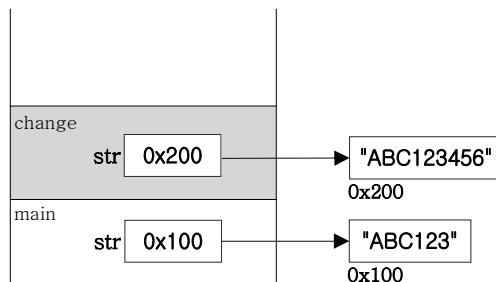
그 다음에 메서드 change를 호출하면서 참조변수 str을 넘겨주면, 메서드 change의 지역 변수 str에 주소값 0x100이 저장된다. 이제 메서드 change의 지역변수 str도 문자열 "ABC123"을 참조하게 된다. 이 두 참조변수는 이름은 같지만 분명히 다른 변수이다. 서로 다른 영역에 존재하기 때문에 이름이 같아도 상관없는 것이다.

```
change(str); // change를 호출하면서 문자열 str을 넘겨준다.
```



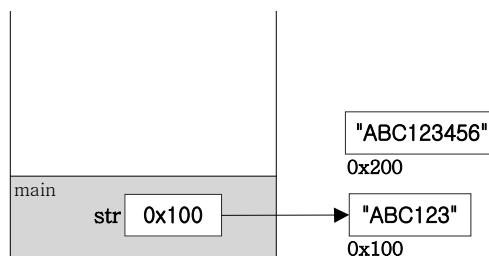
메서드 change에서는 넘겨받은 문자열의 뒤에 "456"을 붙인다. 문자열은 내용을 변경할 수 없기 때문에 덧셈연산을 하면 새로운 문자열이 생성되고 새로운 문자열의 주소가 변수 str에 저장된다.

```
public static void change(String str) {
    str += "456"; // 기존의 문자열에 "456"을 붙인다.
}
```



이제 change메서드는 종료되고, 작업에 사용하던 메모리를 반환하므로 change메서드의 지역변수인 str 역시 메모리에서 제거된다. 다시 main메서드로 돌아와서 문자열 str의 값을 출력하면 처음의 값과 변함없는 값이 출력된다. 문자열 "ABC123456"은 참조하는 변수가 하나도 없으므로 적절한 시기에 가비지컬렉터(garbage collector)에 의해 제거된다.

```
System.out.println("After change:"+str);
```



[6-20] 다음과 같이 정의된 메서드를 작성하고 테스트하시오.

[주의] `Math.random()`을 사용하는 경우 실행결과와 다를 수 있음.

메서드명 : `shuffle`

기 능 : 주어진 배열에 담긴 값의 위치를 바꾸는 작업을 반복하여 뒤섞이게 한다.
처리한 배열을 반환한다.

반환타입 : `int[]`

매개변수 : `int[] arr` - 정수값이 담긴 배열

[연습문제]/ch6/Exercise6_20.java

```
class Exercise6_20
{
    public static int[] shuffle(int[] arr) {
        if(arr==null || arr.length==0)
            return arr;

        for(int i=0; i< arr.length;i++) {
            int j = (int)(Math.random()*arr.length);

            // arr[i] 와 arr[j] 의 값을 서로 바꾼다.
            int tmp = arr[i];
            arr[i] = arr[j];
            arr[j] = tmp;
        }

        return arr;
    }

    public static void main(String[] args)
    {
        int[] original = {1,2,3,4,5,6,7,8,9};
        System.out.println(java.util.Arrays.toString(original));

        int[] result = shuffle(original);
        System.out.println(java.util.Arrays.toString(result));
    }
}
```

[실행결과]

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
[4, 6, 8, 3, 2, 9, 7, 1, 5]
```

[정답]

```
public static int[] shuffle(int[] arr) {
    if(arr==null || arr.length==0)
        return arr;

    for(int i=0; i< arr.length;i++) {
        int j = (int)(Math.random()*arr.length);

        // arr[i] 와 arr[j] 의 값을 서로 바꾼다.
        int tmp = arr[i];
        arr[i] = arr[j];
        arr[j] = tmp;
    }
}
```

```

        arr[j] = tmp;
    }

    return arr;
}

```

[해설] int 배열을 매개변수로 받아서 배열에 저장된 각 요소들의 위치를 여러번 바꿔서 섞은 다음 반복하는 메서드이다.

매개변수로 어떤 값이 넘어올지 모르기 때문에 작업을 시작하기 전에 값의 유효성체크는 반드시 해야 한다. 아래의 코드는 넘겨받은 배열이 null이거나 크기가 0이면 그대로 반환한다.

```

if(arr==null || arr.length==0)
    return arr;

```

반복문을 이용해서 반복적으로 배열의 임의의 두 요소의 값을 바꾼다.

```

for(int i=0; i< arr.length;i++) {
    int j = (int)(Math.random()*arr.length);

    // arr[i]와 arr[j]의 값을 서로 바꾼다.
    int tmp = arr[i];
    arr[i] = arr[j];
    arr[j] = tmp;
}

```

Math.random()을 사용하는 방법이나 두 변수의 값을 바꾸는 것에 대한 설명은 이전 문제들에서 했으므로 생략하겠다.

[6-21] Tv클래스를 주어진 로직대로 완성하시오. 완성한 후에 실행해서 주어진 실행결과와 일치하는지 확인하라.

[참고] 코드를 단순히 하기 위해서 유효성검사는 로직에서 제외했다.

[연습문제]/ch6/Exercise6_21.java

```
class MyTv {
    boolean isPowerOn;
    int channel;
    int volume;

    final int MAX_VOLUME = 100;
    final int MIN_VOLUME = 0;
    final int MAX_CHANNEL = 100;
    final int MIN_CHANNEL = 1;

    void turnOnOff() {
        // (1) isPowerOn의 값이 true면 false로, false면 true로 바꾼다.
        isPowerOn = !isPowerOn;
    }

    void volumeUp() {
        // (2) volume의 값이 MAX_VOLUME보다 작을 때만 값을 1증가시킨다.
        if(volume < MAX_VOLUME)
            volume++;
    }

    void volumeDown() {
        // (3) volume의 값이 MIN_VOLUME보다 클 때만 값을 1감소시킨다.
        if(volume > MIN_VOLUME)
            volume--;
    }

    void channelUp() {
        // (4) channel의 값을 1증가시킨다.
        // 만일 channel이 MAX_CHANNEL이면, channel의 값을 MIN_CHANNEL로 바꾼다.
        if(channel==MAX_CHANNEL) {
            channel = MIN_CHANNEL;
        } else {
            channel++;
        }
    }

    void channelDown() {
        // (5) channel의 값을 1감소시킨다.
        // 만일 channel이 MIN_CHANNEL이면, channel의 값을 MAX_CHANNEL로 바꾼다.
        if(channel==MIN_CHANNEL) {
            channel = MAX_CHANNEL;
        } else {
            channel--;
        }
    }
} // class MyTv

class Exercise6_21 {
    public static void main(String args[]) {
        MyTv t = new MyTv();
```

```
t.channel = 100;
t.volume = 0;
System.out.println("CH:"+t.channel+", VOL:"+ t.volume);

t.channelDown();
t.volumeDown();
System.out.println("CH:"+t.channel+", VOL:"+ t.volume);

t.volume = 100;
t.channelUp();
t.volumeUp();
System.out.println("CH:"+t.channel+", VOL:"+ t.volume);
}
}
```

【실행결과】

```
CH:100, VOL:0
CH:99, VOL:0
CH:100, VOL:100
```

[해설] 답을 보는 것만으로도 별도의 설명이 필요없을 것이라 생각한다. 혹시라도 질문이 있으면 <http://cafe.naver.com/javachobostudy.cafe>의 게시판에 올려주길 바란다.

[6-22] 다음과 같이 정의된 메서드를 작성하고 테스트하시오.

메서드명 : isNumber

기능 : 주어진 문자열이 모두 숫자로만 이루어져있는지 확인한다.

모두 숫자로만 이루어져 있으면 true를 반환하고,

그렇지 않으면 false를 반환한다.

만일 주어진 문자열이 null이거나 빈문자열 "" 이라면 false를 반환한다.

반환타입 : boolean

매개변수 : String str - 검사할 문자열

[Hint] String클래스의 charAt(int i)메서드를 사용하면 문자열의 i번째 위치한 문자를 얻을 수 있다.

[연습문제]/ch6/Exercise6_22.java

```
class Exercise6_22 {
    public static boolean isNumber(String str) {
        if(str==null || str.equals(""))
            return false;

        for(int i=0; i< str.length();i++) {
            char ch = str.charAt(i);

            if(ch < '0' || ch > '9') {
                return false;
            }
        } // for

        return true;
    }

    public static void main(String[] args) {
        String str = "123";
        System.out.println(str+"는 숫자입니까? "+isNumber(str));

        str = "1234o";
        System.out.println(str+"는 숫자입니까? "+isNumber(str));
    }
}
```

[실행결과]

123는 숫자입니까? true

1234o는 숫자입니까? false

[해설] 매개변수로 어떤 값이 넘어올지 모르기 때문에 값의 작업을 시작하기 전에 유효성체크는 반드시 해야 한다. 아래의 코드는 넘겨받은 문자열(str)이 null이거나 빈 문자열("")이면 false를 반환한다.

```
if(str==null || str.equals(""))
    return false;
```

반복문과 `charAt(int i)`을 이용해서 문자열에서 한 문자씩 차례대로 읽어와 `char` 타입의 변수 `ch`에 저장한다.

```
for(int i=0; i< str.length();i++) {  
    char ch = str.charAt(i);
```

읽어온 문자(`ch`)가 숫자가 아니면 `false`를 반환한다.

```
if(ch < '0' || ch > '9') { // if(!(0'<=ch && ch<='9'))와 같다.  
    return false;  
}
```

[6-23] 다음과 같이 정의된 메서드를 작성하고 테스트하시오.

메서드명 : max

기 능 : 주어진 int형 배열의 값 중에서 제일 큰 값을 반환한다.

만일 주어진 배열이 null이거나 크기가 0인 경우, -999999를 반환한다.

반환타입 : int

매개변수 : int[] arr - 최대값을 구할 배열

[연습문제]/ch6/Exercise6_23.java

```
class Exercise6_23{
    public static int max(int[] arr) {
        if(arr==null || arr.length==0)
            return -999999;

        int max = arr[0]; // 배열의 첫 번째 값으로 최대값을 초기화 한다.

        for(int i=1; i< arr.length; i++) { // 배열의 두 번째 값부터 비교한다.
            if(arr[i] > max)
                max = arr[i];
        }

        return max;
    }

    public static void main(String[] args)
    {
        int[] data = {3,2,9,4,7};
        System.out.println(java.util.Arrays.toString(data));
        System.out.println("최대값:"+max(data));
        System.out.println("최대값:"+max(null));
        System.out.println("최대값:"+max(new int[]{})); // 크기가 0인 배열
    }
}
```

[실행결과]

[3, 2, 9, 4, 7]

최대값:9

최대값:-999999

최대값:-999999

[해설] 매개변수로 넘겨받은 배열 arr이 null이거나 크기가 0이면 -999999를 반환한다.

```
if(arr==null || arr.length==0)
    return -999999;
```

배열의 첫 번째 요소(arr[0])로 최대값(max)을 초기화 한다.

```
int max = arr[0]; // 배열의 첫 번째 값으로 최대값을 초기화 한다.
```

최대값 max를 배열의 첫 번째 값으로 초기화 했으므로 첫 번째값은 비교할 필요가 없다. 그래서 두 번째 값(arr[1])부터 비교한다. 비교해서 최대값보다 크면 그 값을 변수 max에 저장한다.

```
for(int i=1; i< arr.length;i++) { // 배열의 두 번째 값부터 비교한다.  
    if(arr[i] > max) // 배열의 i번째 요소가 max보다 크면  
        max = arr[i];  
}
```

반복문을 다 돌고 나면, max에는 배열의 요소 중 가장 큰 값이 저장되어 있을 것이다. 이 값을 반환한다.

```
return max;
```

[6-24] 다음과 같이 정의된 메서드를 작성하고 테스트하시오.

메서드명 : abs
 기 능 : 주어진 값의 절대값을 반환한다.
 반환타입 : int
 매개변수 : int value

[연습문제]/ch6/Exercise6_24.java

```
class Exercise6_24
{
    public static int abs(int value) {
        return value >=0 ? value : -value;
    }

    public static void main(String[] args)
    {
        int value = 5;
        System.out.println(value+"의 절대값:"+abs(value));
        value = -10;
        System.out.println(value+"의 절대값:"+abs(value));
    }
}
```

[실행결과]

```
5의 절대값:5
-10의 절대값:10
```

[해설] value의 값이 양수이면 그대로 반환하고, 음수이면 부호를 바꿔서 반환하면 된다.

if문을 사용해도 되지만 삼항연산자를 이용하면 보다 간결한 코드를 얻을 수 있다. 참고로 if문을 사용한 코드는 다음과 같다.

```
public static int abs(int value) {
    if(value >=0) {
        return value;
    } else {
        return -value; // value가 음수인 경우, 부호를 변경한다.
    }
}
```