[연습문제 - 모범답안]

[3-1] 다음 연산의 결과를 적으시오.

[연습문제]/ch3/Exercise3_1.java

```
class  Exercise3_1 {
   public static void main(String[] args) {
        int x = 2;
        int y = 5;
        char c = 'A'; // 'A'의 문자코드는 65

        System.out.println(1 + x << 33);
        System.out.println(y >= 5 || x < 0 && x > 2);
        System.out.println(y += 10 - x++);
        System.out.println(x+=2);
        System.out.println( !('A' <= c && c <='Z') );
        System.out.println('C'-c);
        System.out.println('5'-'0');
        System.out.println(c+1);
        System.out.println(++c);
        System.out.println(c++);
        System.out.println(c);
   }
}
```

[정답]

[실행결과]

```
6
true
13
5
false
2
5
66
B
B
C
```

[해설]

```
        System.out.println(1 + x << 33);
```

'1 + x << 33'는 x의 값이 2이므로 '1 + 2 << 33'가 된다. 덧셈연산자(+)보다 쉬프트연산자(<<)가 우선순위가 낮으므로 '3 << 33'이 된다. int는 32 bit이므로 33번 쉬프트 하지 않고 1번만 쉬프트 한다. '3 << 1'은 3에 '2의 1제곱' 인 2를 곱하는 것과 같은 결과를 얻으므로 '3 * 2'가 되어 결국 6을 얻는다.

```
        System.out.println(y >= 5 || x < 0 && x > 2);
```

x의 값이 2이고, y의 값이 5이므로 위의 식은 'true || false && false'가 된다. 논리연

산자 'and(&&)'는 'or(||)'보다 우선순위가 높기 때문에 'false && false'가 먼저 연산되어 'true || false'가 되고 최종결과는 true가 된다.

```
System.out.println(y += 10 - x++);
```

'y += 10 - x++'를 풀어쓰면, 'y = y + (10 - x++)'이 된다. x++은 후위형이기 때문에 x의 값이 증가되지 않은 상태에서 (10 - x)는 계산되고 x의 값은 1증가된다.
그래서 (10 - 2)로 계산이 되고 x의 값은 1증가하여 3이 된다. y의 값은 5이므로 식은 'y = 5 + (10 - 2)'가 되어 y에 13이 저장된다.

```
System.out.println(x+=2);
```

'x+=2'는 'x=x+2'와 같다. 이전의 식에서 x의 값이 1증가하였으므로 이제 x의 값은 3이다. 3에 2를 더했으므로 결과는 5가 된다.

```
System.out.println( !('A' <= c && c <='Z') );
```

!('A' <= c && c <='Z')는 문자 c가 대문자가 아닌지를 확인하는 조건식이다. 먼저 괄호안의 'A' <= c && c <='Z'가 먼저 계산되고 마지막에 이 계산결과가 논리부정연산자(!)에 의해 반대(true ↔ false)로 바뀐다. c가 'A'이므로 'A' <= 'A' && 'A' <='Z'가 되고 양쪽의 조건식이 true이므로 'true && true'의 결과인 true를 얻게 된다. 이 결과에 논리부정연산(!)을 수행하니까 true가 false로 바뀌어 최종결과는 false가 된다.

```
System.out.println('C'-c);
```

이항연산자는 피연산자가 int보다 작은 타입(byte, short, char)인 경우 int로 변환한 다음에 연산을 수행한다. c의 값이 'A'이므로 'C'-c는 'C'-'A'가 되고 'C'와 'A'는 int로 변환되어 '67 - 65'가 되고 최종결과는 2가 된다.

```
System.out.println('5'-'0');
```

'5'-'0'도 위와 같은 이유로 '53 - 48'이 되어 5를 결과로 얻는다.

```
System.out.println(c+1);
```

c+1은 c의 값이 'A'이므로 'A'+1이 되고, 이항연산자의 성질(int보다 작은 타입은 int로 변환후 연산)때문에 'A'는 문자코드 값인 65로 변환되어 '65 + 1'을 수행하여 66을 결과로 얻는다. 단지 변수 c에 저장된 값을 읽어서 수식을 계산한 것이므로 변수 c의 저장된 값에는 아무런 변화가 없다.

```
System.out.println(++c);
```

단항연산자인'++'은 이항연산자와 달리 int보다 작은 타입도 형변환을 하지 않는다.(이항연산자는 연산을 위해 '피연산자 스택(operand stack)'을 사용하는데 이 과정에서 형변환이 발생하는 것이다. 반면에 단항연산자인 증가연산자'++'은 '피연산자 스택'을 사용하지 않으므로 형변환도 발생하지 않는다.) 그래서 println은 변수 c를 숫자(int)로 출력하는 것이 아니라 문자로 출력한다. 변수 c에 저장된 문자가 'A'(실제로 저장된 것은 'A'의 문자코드인 65)이므로 문자코드의 값이 1증가되어 66('B'의 문자코드)이 변수 c에 저장된다.

변수 c에 저장된 것은 문자코드, 즉 정수값이다. println은 이 값을 타입에 따라 어떻게 출력할지를 결정한다. 만일 문자타입이면 저장된 값(문자코드)에 해당하는 문자를 출력하고 숫자라면 숫자로 출력한다.

```
System.out.println(c++);
```

단항연산자'++'이 후위형인 경우에는 println()에 의해서 변수 c가 출력된 후에 c에 저장된 값이 증가하므로 문자 'B'가 출력된 후에 변수 c의 값이 1증가해서 문자 'C'가 저장된다.

[3-2] 아래의 코드는 사과를 담는데 필요한 바구니(버켓)의 수를 구하는 코드이다. 만일 사과의 수가 123개이고 하나의 바구니에는 10개의 사과를 담을 수 있다면, 13개의 바구니가 필요할 것이다. (1)에 알맞은 코드를 넣으시오.

[연습문제]/ch3/Exercise3_2.java

```
class  Exercise3_2 {
   public static void main(String[] args) {
        int numOfApples = 123;    // 사과의 개수
        int sizeOfBucket = 10;    // 바구니의 크기(바구니에 담을 수 있는 사과의 개수)
        int numOfBucket =
            numOfApples/sizeOfBucket + (numOfApples%sizeOfBucket > 0 ? 1 : 0) ;

        System.out.println("필요한 바구니의 수 :"+numOfBucket);
   }
}
```

[실행결과]

```
13
```

[정답] **numOfApples/sizeOfBucket + (numOfApples%sizeOfBucket > 0 ? 1 : 0)**

[해설] 사과의 개수(numOfApples)를 바구니의 크기(sizeOfBucket)으로 나눗셈연산(/)을 하면 사과를 담는데 필요한 바구니의 수(numOfBucket)를 구할 수 있다. 정수간의 나눗셈연산의 특징은 반올림을 하지 않고 버림을 한다는 것이다. 예를 들어 125/10의 결과는 13이 아니라 12가 된다. 게다가 int와 int간의 이항연산결과는 int이기 때문에, 12.5와 같은 실수값 결과가 나오지 않는다.

그리고 사과의 개수(numOfApples)를 바구니의 크기(sizeOfBucket)으로 나눴을 때 나머지가 있으면 하나의 바구니가 더 필요하다. 그래서 나머지 연산자(%)를 이용해서 나눗셈연산에서 나머지가 발생하는지 확인해서, 나머지가 발생하면 바구니의 개수(numOfBucket)에 1을 더해줘야 한다.

[3-3] 아래는 변수 num의 값에 따라 '양수', '음수', '0'을 출력하는 코드이다. 삼항 연산자를 이용해서 (1)에 알맞은 코드를 넣으시오.

[Hint] 삼항 연산자를 두 번 사용하라.

[연습문제]/ch3/Exercise3_3.java

```
class  Exercise3_3 {
   public static void main(String[] args) {
        int num = 10;
        System.out.println(num > 0 ? "양수":(num < 0 ? "음수" : "0"));
   }
}
```

[실행결과]

```
양수
```

[정답] `num > 0 ? "양수":(num < 0 ? "음수" : "0")`

[설명] 삼항연산자를 사용하면 2가지 경우의 수를 처리할 수 있다. 삼항연산자에 삼항연산자를 포함시키면 3가지 경우의 수를 처리할 수 있다. num의 값이 0보다 크면, '양수'를 출력하고 끝나지만, num의 값이 0보다 작거나 같으면 괄호안의 삼항연산자가 수행된다. 여기서 num의 값이 0보다 작으면 '음수'가 출력되고, 그렇지 않으면(num의 값이 0이면) '0'이 출력된다.

[3-4] 아래는 변수 num의 값 중에서 백의 자리 이하를 버리는 코드이다. 만일 변수 num의 값이 '456'이라면 '400'이 되고, '111'이라면 '100'이 된다. (1)에 알맞은 코드를 넣으시오.

[연습문제]/ch3/Exercise3_4.java

```
class  Exercise3_4 {
   public static void main(String[] args) {
        int num = 456;
        System.out.println(num/100*100);
   }
}
```

[실행결과]

```
400
```

[정답] num/100*100

[해설] 앞의 문제에서도 설명했듯이, 나눗셈연산자는 반올림을 하지 않고 버림을 한다. 이 성질을 이용한 문제이다.

[3-5] 아래는 변수 num의 값 중에서 일의 자리를 1로 바꾸는 코드이다. 만일 변수 num의 값이 333이라면 331이 되고, 777이라면 771이 된다. (1)에 알맞은 코드를 넣으시오.

[연습문제]/ch3/Exercise3_5.java

```
class  Exercise3_5 {
   public static void main(String[] args) {
        int num = 333;
        System.out.println(num/10*10+1);
   }
}
```

[실행결과]

```
331
```

[정답] num/10*10+1

[해설] 앞의 문제를 약간 더 응용한 문제이다. 이미 다 설명한 내용이므로 자세한 설명은 생략한다.

[3-6] 아래는 변수 num의 값보다 크면서도 가장 가까운 10의 배수에서 변수 num의 값을 뺀 나머지를 구하는 코드이다. 예를 들어, 24의 크면서도 가장 가까운 10의 배수는 30이다. 19의 경우 20이고, 81의 경우 90이 된다. 30에서 24를 뺀 나머지는 6이기 때문에 변수 num의 값이 24라면 6을 결과로 얻어야 한다. (1)에 알맞은 코드를 넣으시오.

[Hint] 나머지 연산자를 사용하라.

[연습문제]/ch3/Exercise3_6.java

```
class  Exercise3_6 {
   public static void main(String[] args) {
        int num = 24;
        System.out.println(10 - num%10);
   }
}
```

[실행결과]

```
6
```

[정답] `10 - num%10`

[해설] 고민을 좀 해봐야하는 문제인데 답을 보고나면 너무 간단해서 허탈할 지도 모르겠다. 이 문제를 풀기위해 여러 연산자를 놓고 고민을 해보는 것에 의미가 있다고 생각해서 넣게 되었다. 다른 정답으로는 '(num/10+1)*10 - num'이 있다.

[3-7] 아래는 화씨(Fahrenheit)를 섭씨(Celcius)로 변환하는 코드이다. 변환공식이 'C = 5/9 ×(F - 32)'라고 할 때, (1)에 알맞은 코드를 넣으시오. 단, 변환 결과값은 소수점 셋째자리에서 반올림해야한다.(Math.round()를 사용하지 않고 처리할 것)

[연습문제]/ch3/Exercise3_7.java

```
class  Exercise3_7 {
   public static void main(String[] args) {
        int fahrenheit = 100;
        float celcius = (int)((5/9f * (fahrenheit - 32))*100 + 0.5) / 100f;

        System.out.println("Fahrenheit:"+fahrenheit);
        System.out.println("Celcius:"+celcius);
   }
}
```

[실행결과]

```
Fahrenheit:100
Celcius:37.78
```

[정답] `(int)((5/9f * (fahrenheit - 32))*100 + 0.5) / 100f`

[해설] 먼저 화씨를 섭씨로 바꾸는 공식은 '5/9f * (fahrenheit - 32)'이다. 5/9의 결과는 0이기 때문에 두 피연산자 중 어느 한 쪽을 반드시 float나 double로 해야만 실수형태의 결과를 얻을 수 있다. 그래서 정수형 리터럴인 9대신 float타입의 리터럴인 9f를 사용하였다. 소수점 셋째자리에서 반올림을 하려면 다음의 과정을 거쳐야한다.

1. 값에 100을 곱한다.
   37.77778 * 100
2. 1의 결과에 0.5를 더한다.
   3777.778 + 0.5 → 3778.278
3. 2의 결과를 int타입으로 변환한다.
   (int)3778,278 → 3778
4. 3의 결과를 100f로 나눈다.(100으로 나누면 소수점 아래의 값을 잃는다.)
   3778 / 100f → 37.78

[3-8] 아래 코드의 문제점을 수정해서 실행결과와 같은 결과를 얻도록 하시오.

[연습문제]/ch3/Exercise3_8.java

```
class  Exercise3_8 {
   public static void main(String[] args) {
        byte a = 10;
        byte b = 20;
        byte c = (byte)(a + b);

        char ch = 'A';
        ch = (char)(ch + 2);

        float f = 3 / 2f;
        long l = 3000 * 3000 * 3000L;

        float f2 = 0.1f;
        double d = 0.1;

        boolean result = (float)d==f2;

        System.out.println("c="+c);
        System.out.println("ch="+ch);
        System.out.println("f="+f);
        System.out.println("l="+l);
        System.out.println("result="+result);
   }
}
```

[실행결과]

```
c=30
ch=C
f=1.5
l=27000000000
result=true
```

[정답]

```
byte c = a + b;  → byte c = (byte)(a + b);
ch = ch + 2;     → ch = (char)(ch + 2);
float f = 3 / 2; → float f = 3 / 2f;
long l = 3000 * 3000 * 3000; → long l = 3000 * 3000 * 3000L;
boolean result = d==f2; → boolean result = (float)d==f2;
```

[해설] 이항연산은 두 피연산자의 타입을 일치시킨 후 연산을 수행한다는 것,그리고 int보다 작은 타입은 int로 자동변환한다는 것은 반드시 기억하고 있어야하는 중요한 내용이다.

```
byte c = a + b;  → byte c = (byte)(a + b);
```

변수 a와 b는 모두 byte타입이므로 이항연산인 덧셈연산을 하기 전에 int타입으로 자동형변환된다. int와 int의 덧셈이므로 연산결과는 int가 된다. int타입의 값(4 byte)을 byte타입의 변수(1 byte)에 담아야하므로 형변환을 해주어야한다.

```
ch = ch + 2;     → ch = (char)(ch + 2);
```

이것도 이전의 경우와 마찬가지이다. char타입이 덧셈연산의 과정을 거치면서 int타입으로 변환되므로 char타입의 변수에 저장하기 위해서는 char타입으로 형변환을 해주어야한다.

```
float f = 3 / 2; → float f = 3 / 2f;
```

int와 int의 연산결과는 int이기 때문에 3/2의 결과는 1이 된다. 연산결과를 실수로 얻고 싶으면, 적어도 두 피연산자 중 한 쪽이 실수타입(float와 double중의 하나)이어야한다.

```
long l = 3000 * 3000 * 3000; → long l = 3000 * 3000 * 3000L;
```

int*int*int의 결과는 int이므로 int타입의 최대값인 약 2*10의 9제곱을 넘는 결과는 오버플로우가 발생하여 예상한 것과는 다른 값을 얻는다. 그래서 피연산자 중 적어도 한 값은 long타입이어야 최종결과를 long타입의 값으로 얻기 때문에 long타입의 접미사 'L'을 붙여서 long타입의 리터럴로 만들어줘야 한다.

```
boolean result = d==f2; → boolean result = (float)d==f2;
```

비교연산자도 이항연산자이므로 연산 시에 두 피연산자의 타입을 맞추기 위해 형변환이 발생한다. 그래서 double과 float의 연산은 double과 double의 연산으로 자동형변환 되는데, 실수는 정수와 달리 근사값으로 표현을 하기 때문에 float를 double로 형변환했을 때 오차가 발생할 수 있다.

그래서 float값을 double로 형변환하기 보다는 double값을 유효자리수가 적은 float로 형변환해서 비교하는 것이 정확한 결과를 얻는다.

[3-9] 다음은 문자형 변수 ch가 영문자(대문자 또는 소문자)이거나 숫자일 때만 변수 b의 값이 true가 되도록 하는 코드이다. (1)에 알맞은 코드를 넣으시오.

[연습문제]/ch3/Exercise3_9.java

```
class  Exercise3_9 {
   public static void main(String[] args) {
        char ch = 'z';
        boolean b = ('a' <= ch && ch <= 'z') || ('A' <= ch && ch <= 'Z') || ('0' <= ch && ch <= '9');

        System.out.println(b);
   }
}
```

[실행결과]

```
true
```

[정답] ('a' <= ch && ch <= 'z') || ('A' <= ch && ch <= 'Z') || ('0' <= ch && ch <= '9') 괄호는 생략해도 된다.

[3-10] 다음은 대문자를 소문자로 변경하는 코드인데, 문자 ch에 저장된 문자가 대문자인 경우에만 소문자로 변경한다. 문자코드는 소문자가 대문자보다 32만큼 더 크다. 예를 들어 'A'의 코드는 65이고 'a'의 코드는 97이다. (1)~(2)에 알맞은 코드를 넣으시오.

[연습문제]/ch3/Exercise3_10.java

```
class  Exercise3_10 {
   public static void main(String[] args) {
        char ch = 'A';

        char lowerCase = ('A' <= ch && ch <= 'Z') ? (char)(ch+32): ch;

        System.out.println("ch:"+ch);
        System.out.println("ch to lowerCase:"+lowerCase);
   }
}
```

[실행결과]

```
ch:A
ch to lowerCase:a
```

[정답] `('A' <= ch && ch <= 'Z'), (char)(ch+32)`

[해설] 대문자인 경우에만 문자코드의 값을 32만큼 증가시키면 소문자가 된다. 문자 ch가 대문자인지를 확인하는 조건식은 **'A' <= ch && ch <= 'Z'**이고, 문자 ch의 문자코드를 32증가시키기 위해서는 덧셈연산을 해야하는데, 이 때 덧셈연산의 결과가 int이므로 char타입으로의 형변환이 필요하다.

[연습문제 - 모범답안]

[4-1] 다음의 문장들을 조건식으로 표현하라.

1. int형 변수 x가 10보다 크고 20보다 작을 때 true인 조건식
2. char형 변수 ch가 공백이나 탭이 아닐 때 true인 조건식
3. char형 변수 ch가 'x' 또는 'X'일 때 true인 조건식
4. char형 변수 ch가 숫자('0'~'9')일 때 true인 조건식
5. char형 변수 ch가 영문자(대문자 또는 소문자)일 때 true인 조건식
6. int형 변수 year가 400으로 나눠떨어지거나 또는 4로 나눠떨어지고 100으로 나눠떨어지지 않을 때 true인 조건식
7. boolean형 변수 powerOn가 false일 때 true인 조건식
8. 문자열 참조변수 str이 "yes"일 때 true인 조건식

[정답 & 해설]

1. int형 변수 x가 10보다 크고 20보다 작을 때 true인 조건식

   `10 < x && x < 20`

2. char형 변수 ch가 공백이나 탭이 아닐 때 true인 조건식

   `!(ch == ' ' || ch =='₩t') 또는 ch!=' ' && ch !='₩t'`

3. char형 변수 ch가 'x' 또는 'X'일 때 true인 조건식

   `ch == 'x' || ch == 'X'`

4. char형 변수 ch가 숫자('0'~'9')일 때 true인 조건식

   `'0' <= ch && ch <='9'`

5. char형 변수 ch가 영문자(대문자 또는 소문자)일 때 true인 조건식

   `('a' <= ch && ch <= 'z') || ('A' <= ch && ch <= 'Z')`

6. int형 변수 year가 400으로 나눠떨어지거나 또는 4로 나눠떨어지고 100으로 나눠떨어지지 않을 때 true인 조건식

   `year%400==0 || year%4==0 && year%100!=0`

7. boolean형 변수 powerOn가 false일 때 true인 조건식

   `!powerOn 또는 powerOn==false`

8. 문자열 참조변수 str이 "yes"일 때 true인 조건식

   `str.equals("yes") 또는 "yes".equals(str)`

[4-2] 1부터 20까지의 정수 중에서 2 또는 3의 배수가 아닌 수의 총합을 구하시오.

[정답] 73

[해설]

[연습문제]/ch4/Exercise4_2.java

```
class  Exercise4_2 {
    public static void main(String[] args) {
        int sum = 0;

        for(int i=1; i <=20; i++) {
            if(i%2!=0 && i%3!=0) //i가 2또는 3의 배수가 아닐 때만 sum에 i를 더한다.
                sum +=i;
        }

        System.out.println("sum="+sum);
    } // main
}
```

[4-3] 1+(1+2)+(1+2+3)+(1+2+3+4)+...+(1+2+3+...+10)의 결과를 계산하시오.

[정답] 220

[해설]

[연습문제]/ch4/Exercise4_3.java

```
class  Exercise4_3 {
    public static void main(String[] args) {
        int sum = 0;
        int totalSum = 0;

        for(int i=1; i <=10; i++) {
            sum += i;
            totalSum += sum;
        }

        System.out.println("totalSum="+totalSum);
    } // main
}
```

i의 값이 1부터 10까지 1씩 증가하며 변하는 동안, i의 값을 누적해서 sum에 저장하면 sum의 값은 1,3,6,10,15,21,28,36,45,55의 순서로 변해간다. 즉, 1, 1+2, 1+2+3, 1+2+3+4, 1+2+3+4+5,...,1+2+3+4+5+6+7+8+9,1+2+3+4+5+6+7+8+9+10의 순서로 변해간다.

따라서 1+(1+2)+(1+2+3)+(1+2+3+4)+...+(1+2+3+...+10)의 결과를 계산하려면, sum의 값을 계속 더해나가면 된다. 그래서 totalSum이라는 변수를 두고, 이 변수에 sum의 값을 계속해서 누적하여 결과를 얻었다.

i	sum	totalSum
1	1	1
2	3 = 1+2	4 =1+3 = 1+(1+2)
3	6 = 1+2+3	10 =1+3+6 = 1+(1+2)+(1+2+3)
4	10= 1+2+3+4	20 =1+3+6+10
5	15 =1+2+3+4+5	35 =1+3+6+10+15
6	21 =1+2+3+4+5+6	56 =1+3+6+10+15+21
7	28 =1+2+3+4+5+6+7	84 =1+3+6+10+15+21+28
8	36 =1+2+3+4+5+6+7+8	120=1+3+6+10+15+21+28+36
9	45 =1+2+3+4+5+6+7+8+9	165=1+3+6+10+15+21+28+36+45
10	55 =1+2+3+4+5+6+7+8+9+10	220=1+3+6+10+15+21+28+36+45+55

[4-4] 1+(-2)+3+(-4)+... 과 같은 식으로 계속 더해나갔을 때, 몇까지 더해야 총합이 100이상이 되는지 구하시오.

[정답] 199

[해설]

[연습문제]/ch4/Exercise4_4.java

```
class  Exercise4_4 {
    public static void main(String[] args) {
        int sum = 0; // 총합을 저장할 변수
        int s = 1;   // 값의 부호를 바꿔주는데 사용할 변수
        int num = 0;

        // 조건식의 값이 true이므로 무한반복문이 된다.
        for(int i=1;true; i++, s=-s) { // 매 반복마다 s의 값은 1, -1, 1, -1...
            num = s * i; // i와 부호(s)를 곱해서 더할 값을 구한다.
            sum += num;

            if(sum >=100) // 총합이 100보다 같거나 크면 반복문을 빠져 나간다.
                break;
        }

        System.out.println("num="+num);
        System.out.println("sum="+sum);
    } // main
}
```

for문 대신 while문을 써도 좋고, for문을 보다 간략히 하면 다음과 같이 할 수 있다.

```
for(int i=1; sum < 100; i++, s=-s) {
    num = i * s;
    sum += num;
}
```

[4-5] 다음의 for문을 while문으로 변경하시오.

[연습문제]/ch4/Exercise4_5.java

```
public class Exercise4_5 {
     public static void main(String[] args) {
       for(int i=0; i<=10; i++) {
            for(int j=0; j<=i; j++)
                System.out.print("*");
            System.out.println();
       }
     } // end of main
} // end of class
```

[정답]

[연습문제]/ch4/Exercise4_5_2.java

```
public class Exercise4_5_2 {
     public static void main(String[] args) {
        int i=0;
        while( i<=10) {
            int j=0;
            while(j<=i) {
                System.out.print("*");
                j++;
            }
            System.out.println();
            i++;
        }
     } // end of main
} // end of class
```

[4-6] 두 개의 주사위를 던졌을 때, 눈의 합이 6이 되는 모든 경우의 수를 출력하는 프로그램을 작성하시오.

[정답]

1+5=6
2+4=6
3+3=6
4+2=6
5+1=6

[해설]

[연습문제]/ch4/Exercise4_6.java

```
class  Exercise4_6 {
    public static void main(String[] args) {
        for(int i=1;i<=6;i++)
            for(int j=1;j<=6;j++)
                if(i+j==6)
                    System.out.println(i+"+"+j+"="+(i+j));

    } // main
}
```

반복문을 중첩해서 1부터 6까지를 반복하면서 두 값의 합이 6인 경우를 화면에 출력하면 된다.

[4-7] Math.random()을 이용해서 1부터 6사이의 임의의 정수를 변수 value에 저장하는 코드를 완성하라. (1)에 알맞은 코드를 넣으시오.

[연습문제]/ch4/Exercise4_7.java

```
class  Exercise4_7 {
   public static void main(String[] args) {
        int value = (int)(Math.random()*6)+1;

        System.out.println("value:"+value);
   }
}
```

[정답] `(int)(Math.random()*6)+1`

[해설]

1. 각 변에 6을 곱한다.

```
0.0 * 6 <= Math.random() * 6 < 1.0 * 6
    0.0 <= Math.random() * 6 < 6.0
```

2. 각 변을 int형으로 변환한다.

```
(int)0.0 <= (int)(Math.random() * 6) < (int)6.0
       0 <= (int)(Math.random() * 6) < 6
```

지금까지는 0과 6사이의 정수 중 하나를 가질 수 있다. 0은 포함되고 6은 포함되지 않는다.

3. 각 변에 1을 더한다.

```
0 + 1 <= (int)(Math.random() * 6) + 1 < 6 + 1
    1 <= (int)(Math.random() * 6) + 1 < 7
```

자, 이제는 1과 7사이의 정수 중 하나를 얻을 수 있다. 단, 1은 범위에 포함되고 7은 포함되지 않는다.

[4-8] 방정식 2x+4y=10의 모든 해를 구하시오. 단, x와 y는 정수이고 각각의 범위는 0<=x<=10, 0<=y<=10 이다.

[실행결과]

```
x=1, y=2
x=3, y=1
x=5, y=0
```

[정답]

[연습문제]/ch4/Exercise4_8.java

```
class Exercise4_8
{
    public static void main(String[] args)
    {
        for(int x=0; x <=10;x++) {
            for(int y=0; y <=10;y++) {
                if(2*x+4*y==10) {
                    System.out.println("x="+x+", y="+y);
                }
            }
        }
    } // main
}
```

[해설] 문제4-6과 유사한 문제이다. 반복문을 중첩해서 0부터 10까지 1씩 증가시켜가면서 돌린다. 반복과정에서 방정식을 만족시키는 경우에만 x와 y의 값을 출력하면 된다.

[4-9] 숫자로 이루어진 문자열 str이 있을 때, 각 자리의 합을 더한 결과를 출력하는 코드를 완성하라. 만일 문자열이 "12345"라면, '1+2+3+4+5'의 결과인 15를 출력이 출력되어야 한다. (1)에 알맞은 코드를 넣으시오.

[Hint] String클래스의 charAt(int i)을 사용

[연습문제]/ch4/Exercise4_9.java

```
class  Exercise4_9 {
  public static void main(String[] args) {
      String str = "12345";
      int sum = 0;

      for(int i=0; i < str.length(); i++) {
          sum += str.charAt(i) - '0';
      }

      System.out.println("sum="+sum);
  }
}
```

[실행결과]

```
15
```

[정답] sum += str.charAt(i) - '0';

[해설] charAt(int i)메서드는 문자열에서 i번째 문자를 반환한다.(i의 값은 0부터 시작한다.) 예를 들어, "Hello"라는 문자열이 있을 때 "Hello".charAt(4)는 문자 'o'가 된다.

index	0	1	2	3	4
char	H	e	l	l	o

charAt(int i)을 이용해서 반복문으로 각 문자열의 문자를 하나씩 읽어서 숫자로 변환한 다음 sum에 계속 더하면 된다. 문자'3'을 숫자 3로 바꾸는 방법은 문자'3'에서 문자'0'을 빼주는 것이다. 알파벳이나 숫자는 문자코드가 연속적으로 할당되었기 때문에 이런 방법이 가능하다.

문자	문자코드
...	...
'0'	48
'1'	49
'2'	50
'3'	51
...	...

뺄셈과 같은 이항연산자는 int타입보다 작은 타입은 피연산자(byte, short, char)은 int로 변환한다. 그래서 '3'-'0'은 51 - 48으로 변환되고 그 결과는 숫자 3이 된다.

'3'-'0' → 51 - 48 → 3

[4-10] int타입의 변수 num 이 있을 때, 각 자리의 합을 더한 결과를 출력하는 코드를 완성하라. 만일 변수 num의 값이 12345라면, '1+2+3+4+5'의 결과인 15를 출력하라. (1)에 알맞은 코드를 넣으시오.

[주의] 문자열로 변환하지 말고 숫자로만 처리해야 한다.

[연습문제]/ch4/Exercise4_10.java

```
class  Exercise4_10 {
   public static void main(String[] args) {
        int num = 12345;
        int sum = 0;

        while(num > 0) {
            sum += num%10;
            num /= 10;
        }

        System.out.println("sum="+sum);
   }
}
```

[실행결과]

```
15
```

[정답]

```
        while(num > 0) {
            sum += num%10;
            num /= 10;
        }
```

[해설] 문제4-9에서처럼 문자열에서 charAt(int i)를 이용해서 문자를 숫자로 변환하는 것보다 숫자에서 각 자리수의 숫자를 하나씩 뽑아내는 것은 더 어렵다. 하지만, 숫자의 마지막 자리를 어떻게 뽑아내는지만 알아내면 나머지는 쉽게 해결된다.

방법은 의외로 간단하다 아래와 같이 숫자를 10으로 반복해서 나눠가면서, 10으로 나머지 연산을 하면 일의 자리를 얻어낼 수 있다.

num	num%10
12345	5
1234	4
123	3
12	2
1	1

이 값들을 더하기만하면 변수 num에 저장된 숫자의 각 자리수를 모두 더한 값을 구할 수 있다.

[4-11] 피보나치(Fibonnaci) 수열(數列)은 앞을 두 수를 더해서 다음 수를 만들어 나가는 수열이다. 예를 들어 앞의 두 수가 1과 1이라면 그 다음 수는 2가 되고 그 다음 수는 1과 2를 더해서 3이 되어서 1,1,2,3,5,8,13,21,... 과 같은 식으로 진행된다. 1과 1부터 시작하는 피보나치수열의 10번째 수는 무엇인지 계산하는 프로그램을 완성하시오.

[연습문제]/ch4/Exercise4_11.java

```
public class Exercise4_11 {
     public static void main(String[] args) {
          // Fibonnaci 수열의 시작의 첫 두 숫자를 1, 1로 한다.
          int num1 = 1;
          int num2 = 1;
          int num3 = 0;  // 세번째 값
          System.out.print(num1+","+num2);

          for (int i = 0 ; i < 8 ; i++ ) {
               num3 = num1 + num2; // 세번째 값은 첫번째와 두번째 값을 더해서 얻는다.
               System.out.print(","+num3);  // 세 번째 수열 출력

               num1 = num2; // 두 번째 수열을 첫 번째 값으로 한다.
               num2 = num3; // 세 번째 수열을 두 번째 값으로 한다.
          }
     } // end of main
} // end of class
```

[실행결과]

```
1,1,2,3,5,8,13,21,34,55
```

[정답]

```
num3 = num1 + num2; // 세 번째 값은 첫 번째와 두 번째 값을 더해서 얻는다.
System.out.print(","+num3);  // 세 번째 수열 출력

num1 = num2; // 두 번째 수열을 첫 번째 값으로 한다.
num2 = num3; // 세 번째 수열을 두 번째 값으로 한다.
```

[해설] 워낙 간단한 문제이니 정답을 보는 것만으로도 다른 설명이 필요하지는 않을 것이라 생각한다. 혹시라도 질문이 있으면 http://cafe.naver.com/javachobostudy.cafe의 게시판에 올려주길 바란다.

[4-12] 구구단의 일부분을 다음과 같이 출력하시오.

[실행결과]

```
2*1=2     3*1=3     4*1=4
2*2=4     3*2=6     4*2=8
2*3=6     3*3=9     4*3=12

5*1=5     6*1=6     7*1=7
5*2=10    6*2=12    7*2=14
5*3=15    6*3=18    7*3=21

8*1=8     9*1=9
8*2=16    9*2=18
8*3=24    9*3=27
```

[정답]

[연습문제]/ch4/Exercise4_12.java

```
public class Exercise4_12 {
   public static void main(String[] args) {
       for (int i = 1 ; i <= 9 ; i++) {
           for (int j = 1; j <= 3; j++) {
               int x = j+1+(i-1)/3*3;
               int y = i%3==0? 3 : i%3 ;

               if(x > 9) // 9단까지만 출력한다. 이 코드가 없으면 10단까지 출력된다.
                   break;

               System.out.print(x+"*"+y+"="+x*y+"\t"); //println이 아님에 주의
           }
           System.out.println();
           if(i%3==0) System.out.println(); //
       }
   } // end of main
} // end of class
```

[해설] 이 문제를 푸는 방법은 여러 가지가 있겠지만, 반복문을 2번 사용해서 작성해 보았다. 가로로 출력을 했기 때문에 좀 복잡하긴 한데, 일단 세로로 출력해본 다음에 가로로 출력되도록 변경하면 문제풀기가 수월해진다. 그래서 실행결과와 같은 결과를 얻기 위해서는 먼저 아래와 같이 출력할 수 있어야 한다.

```
2*1=2
3*1=3
4*1=4
2*2=4
3*2=6
4*2=8
2*3=6
3*3=9
4*3=12
5*1=5
```

```
6*1=6
7*1=7
5*2=10
6*2=12
7*2=14
5*3=15
6*3=18
7*3=21
8*1=8
9*1=9
10*1=10
8*2=16
9*2=18
10*2=20
8*3=24
9*3=27
10*3=30
```

원래는 2단부터 9단까지 출력하는 것이지만, 문제를 쉽게 하기 위해서 10단까지 출력하는 것이 좋다. 구구단을 곱하기 3까지만 출력하므로 전체 반복회수는 3 * 9 = 27번이 된다. 3번 반복하는 반복문과 9번 반복하는 반복문을 2중으로 사용하면 되는 것이다.

```
public class Exercise4_12 {
   public static void main(String[] args) {
        for (int i = 1 ; i <= 9 ; i++) {
            for (int j = 1; j <= 3; j++) {
                // 이 블럭{}안의 코드가 9 * 3 = 27번 반복된다.
            }
        }
   } // end of main
} // end of class
```

단을 의미하는 변수 x와 곱하는 숫자를 의미하는 변수 y를 정의하고 카운터 i와 j를 이용해서 x와 y의 값을 적절히 계산해 주는 것이 이 문제의 핵심이다.

```
public class Exercise4_12 {
   public static void main(String[] args) {
        for (int i = 1 ; i <= 9 ; i++) {
            for (int j = 1; j <= 3; j++) {
                int x = 0; // i와 j값을 이용해서 적절히 계산해야 한다.
                int y = 0; // i와 j값을 이용해서 적절히 계산해야 한다.
                System.out.println(x+"*"+y+"="+x*y);
            }
        }
   } // end of main
} // end of class
```

i와 j의 값의 변화와 그에 따른 x와 y값의 변화를 표로 적어보았다. 어떻게 하면 i와 y의 값을 이용해서 x와 y의 값을 구해낼 수 있을까?

먼저 i의 값과 y의 값을 보면 유사한 점이 많다. y의 값은 i의 값만 잘 이용하면 어떻게 될 것 같다는 생각이 든다. i가 1~9까지 1씩 증가하는데 반해 y의 값은 1,2,3이 계속해서

반복된다. 그래서 i를 3으로 나머지 연산을 해봤더니, 우리가 원하는 y의 값과 매우 유사해 졌다. 다만 i%3의 결과가 0일 때, 3으로 바꾸면 된다. 그래서 다음과 같은 식이 만들어지게 된 것이다.

```
public class Exercise4_12 {
   public static void main(String[] args) {
        for (int i = 1 ; i <= 9 ; i++) {
            for (int j = 1; j <= 3; j++) {
                int x = 0; // i와 j값을 이용해서 적절히 계산해야 한다.
                int y = i%3==0? 3 : i%3 ;
                System.out.println(x+"*"+y+"="+x*y);
            }
        }
   } // end of main
} // end of class
```

i	j	x	y	i%3	j+1	x-(j+1)
1	1	2	1	1	2	0
1	2	3	1	1	3	0
1	3	4	1	1	4	0
2	1	2	2	2	2	0
2	2	3	2	2	3	0
2	3	4	2	2	4	0
3	1	2	3	0	2	0
3	2	3	3	0	3	0
3	3	4	3	0	4	0
4	1	5	1	1	2	3
4	2	6	1	1	3	3
4	3	7	1	1	4	3
5	1	5	2	2	2	3
5	2	6	2	2	3	3
5	3	7	2	2	4	3
6	1	5	3	0	2	3
6	2	6	3	0	3	3
6	3	7	3	0	4	3
7	1	8	1	1	2	6
7	2	9	1	1	3	6
7	3	10	1	1	4	6
8	1	8	2	2	2	6
8	2	9	2	2	3	6
8	3	10	2	2	4	6
9	1	8	3	0	2	6
9	2	9	3	0	3	6
9	3	10	3	0	4	6

이제 x의 값만 구하면 되는데, x의 값은 j의 값과 좀 유사하다. 다만 2단부터 시작하기 때문에 x의 값은 1이 아닌 2부터 시작한다. 그래서 j의 값에 1을 더해서 j+1의 값을 적어보니 x의 값과 3 또는 6의 차이가 있다. 이 값의 차이를 보정해주기 위해서는 어떻게 식을 만들어 가야 할까? i와 j중에서 i는 증가하는 성질의 값이고 j는 반복되는 성질의 값이므로 j보다는 i가 적절할 것 같다. i의 값을 가지고 어떻게 하면 원하는 값을 얻을 수 있을까 고민해보자.

```
public class Exercise4_12 {
   public static void main(String[] args) {
        for (int i = 1 ; i <= 9 ; i++) {
            for (int j = 1; j <= 3; j++) {
                int x = (j+1) + (???); // (???)안에 적절한 수식이 들어가야 한다.
                int y = i%3==0? 3 : i%3 ;
                System.out.println(x+"*"+y+"="+x*y);
            }
        }
   } // end of main
} // end of class
```

i	j	x	y	j+1	i/3	(i-1)/3	(i-1)/3*3	j+1+(i-1)/3*3
1	1	2	1	2	0	0	0	2
1	2	3	1	3	0	0	0	3
1	3	4	1	4	0	0	0	4
2	1	2	2	2	0	0	0	2
2	2	3	2	3	0	0	0	3
2	3	4	2	4	0	0	0	4
3	1	2	3	2	1	0	0	2
3	2	3	3	3	1	0	0	3
3	3	4	3	4	1	0	0	4
4	1	5	1	2	1	1	3	5
4	2	6	1	3	1	1	3	6
4	3	7	1	4	1	1	3	7
5	1	5	2	2	1	1	3	5
5	2	6	2	3	1	1	3	6
5	3	7	2	4	1	1	3	7
6	1	5	3	2	2	1	3	5
6	2	6	3	3	2	1	3	6
6	3	7	3	4	2	1	3	7
7	1	8	1	2	2	2	6	8
7	2	9	1	3	2	2	6	9
7	3	10	1	4	2	2	6	10
8	1	8	2	2	2	2	6	8
8	2	9	2	3	2	2	6	9
8	3	10	2	4	2	2	6	10
9	1	8	3	2	3	2	6	8
9	2	9	3	3	3	2	6	9
9	3	10	3	4	3	2	6	10

위의 표와 같은 과정을 거쳐 i의 값을 연산했더니 우리가 원했던 값을 얻을 수 있었다.

혼자서 문제를 풀 때는 어려웠는데 별 것 아니었다고 느끼는가? 아니면 여전히 어려운가? 어렵다고 느끼는 이유는 이러한 문제풀이과정이 익숙하지 않기 때문일 뿐이다.

이러한 문제들을 통해 여러분들이 얻어야 하는 것은 문제를 풀어가는 과정이라고 생각한다. 코드만 이리저리 주무르다가 답을 얻는 것이 아니라 논리적인 과정을 하나하나 거쳐 나가면서 답을 만들어 가는 것이 여러분들이 진정으로 배워야하는 것이다.

나머지 과정은 설명하지 않아도 이해할 수 있으리라 생각하고 생략하겠다. 이와 같은 과정을 거쳐 구구단의 일부가 아닌 전체를 출력하는 코드를 작성해보자.

[4-13] 다음은 주어진 문자열(value)이 숫자인지를 판별하는 프로그램이다. (1)에 알맞은 코드를 넣어서 프로그램을 완성하시오.

[연습문제]/ch4/Exercise4_13.java

```
class Exercise4_13
{
     public static void main(String[] args)
     {
          String value = "12o34";
          char ch = ' ';
          boolean isNumber = true;

          // 반복문과 charAt(int i)를 이용해서 문자열의 문자를
          // 하나씩 읽어서 검사한다.
          for(int i=0; i < value.length() ;i++) {
               ch = value.charAt(i);

               if(!('0'<=ch && ch<='9')) {
                    isNumber = false;
                    break;
               }
          }

          if (isNumber) {
               System.out.println(value+"는 숫자입니다.");
          } else {
               System.out.println(value+"는 숫자가 아닙니다.");
          }
     }  // end of main
}  // end of class
```

[실행결과]

```
12o34는 숫자가 아닙니다.
```

[정답]

```
ch = value.charAt(i);

if(!('0'<=ch && ch<='9')) {
    isNumber = false;
    break;
}
```

[해설] charAt(int i)메서드는 문자열에서 i번째 문자를 반환한다.(i의 값은 0부터 시작한다.) "12o34"라는 문자열이 있을 때 "12o34".charAt(2)는 문자 'o'가 된다.

index	0	1	**2**	3	4
char	1	2	**o**	3	4

조건식 '0'<=ch && ch<='9'는 문자 ch가 숫자('0'~'9'사이의 문자)이면 참(true)이 된다. 이 조건식 전체에 논리부정 연산자 '!'를 붙였으니, 문자 ch가 숫자가 아니어야 참(true)인 조건식이 된다. 이 조건식을 만족하는 경우(문자열 중의 어느 한 문자라도 숫자가 아닌 경우)에만 isNumber의 값을 false로 바꾸고 break문을 수행해서 반복문을 빠져나온다.

[4-14] 다음은 숫자맞추기 게임을 작성한 것이다. 1과 100사이의 값을 반복적으로 입력해서 컴퓨터가 생각한 값을 맞추면 게임이 끝난다. 사용자가 값을 입력하면, 컴퓨터는 자신이 생각한 값과 비교해서 결과를 알려준다. 사용자가 컴퓨터가 생각한 숫자를 맞추면 게임이 끝나고 몇 번 만에 숫자를 맞췄는지 알려준다. (1)~(2)에 알맞은 코드를 넣어 프로그램을 완성하시오.

[연습문제]/ch4/Exercise4_14.java

```
class Exercise4_14
{
     public static void main(String[] args)
     {
       // 1~100사이의 임의의 값을 얻어서 answer에 저장한다.
       int answer = (int)(Math.random() * 100) + 1;
       int input = 0;           // 사용자입력을 저장할 공간
       int count = 0;           // 시도횟수를 세기위한 변수

       // 화면으로 부터 사용자입력을 받기 위해서 Scanner클래스 사용
       java.util.Scanner s = new java.util.Scanner(System.in);

       do {
            count++;
            System.out.print("1과 100사이의 값을 입력하세요 :");
            input = s.nextInt(); // 입력받은 값을 변수 input에 저장한다.

            if(answer > input) {
                System.out.println("더 큰 수를 입력하세요.");
            } else if(answer < input) {
                System.out.println("더 작은 수를 입력하세요.");
            } else {
                System.out.println("맞췄습니다.");
                System.out.println("시도횟수는 "+count+"번입니다.");
                break;           // do-while문을 벗어난다
            }
       } while(true); // 무한반복문
     }      // end of main
}      // end of class HighLow
```

[실행결과]

```
1과 100사이의 값을 입력하세요 :50
더 큰 수를 입력하세요.
1과 100사이의 값을 입력하세요 :75
더 큰 수를 입력하세요.
1과 100사이의 값을 입력하세요 :87
더 작은 수를 입력하세요.
1과 100사이의 값을 입력하세요 :80
더 작은 수를 입력하세요.
1과 100사이의 값을 입력하세요 :77
더 작은 수를 입력하세요.
1과 100사이의 값을 입력하세요 :76
맞췄습니다.
시도횟수는 6번입니다.
```

[정답] **(1) (int)(Math.random() * 100) + 1;**

```
(2)
    if(answer > input) {
        System.out.println("더 큰 수를 입력하세요.");
    } else if(answer < input) {
        System.out.println("더 작은 수를 입력하세요.");
    } else {
        System.out.println("맞췄습니다.");
        System.out.println("시도횟수는 "+count+"번입니다.");
        break;            // do-while문을 벗어난다
    }
```

[해설] (1)에는 1과 100사이의 임의의 정수를 구하는 코드가 들어가야 하며 다음과 같은 과정을 통해 만들어낼 수 있다.

1. 각 변에 100을 곱한다.

```
0.0 * 100 <= Math.random() * 100 < 1.0 * 100
    0.0 <= Math.random() * 100 < 100.0
```

2. 각 변을 int형으로 변환한다.

```
(int)0.0 <= (int)(Math.random() * 100) < (int)100.0
       0 <= (int)(Math.random() * 100) < 100
```

지금까지는 0과 100사이의 정수 중 하나를 가질 수 있다. 0은 포함되고 100은 포함되지 않는다.

3. 각 변에 1을 더한다.

```
0 + 1 <= (int)(Math.random() * 100) + 1 < 100 + 1
    1 <= (int)(Math.random() * 100) + 1 < 101
```

자, 이제는 1과 101사이의 정수 중 하나를 얻을 수 있다. 1은 포함되고 101은 포함되지 않는다.

(2)에 들어갈 코드는 입력받은 값(input)이 컴퓨터가 생각한 값(answer)과 큰 지, 작은 지, 같은지 모두 세가지 경우에 대해 처리해야하므로 if-else if문을 사용했다. 변수 input의 값과 answer의 값이 같은 경우(else블럭)에는 break문을 만나 do-while문을 빠져나오게 된다.

[4-15] 다음은 회문수를 구하는 프로그램이다. 회문수(palindrome)란, 숫자를 거꾸로 읽어도 앞으로 읽는 것과 같은 수를 말한다. 예를 들면 '12321'이나 '13531'같은 수를 말한다. (1)에 알맞은 코드를 넣어서 프로그램을 완성하시오.

[Hint] 나머지 연산자를 이용하시오.

[연습문제]/ch4/Exercise4_15.java

```
class Exercise4_15
{
   public static void main(String[] args)
   {
        int number = 12321;
        int tmp = number;

        int result =0;  // 변수 number를 거꾸로 변환해서 담을 변수

        while(tmp !=0) {
            result = result*10 + tmp % 10; // 기존 결과에 10을 곱해서 더한다.
            tmp /= 10;
        }

        if(number == result)
            System.out.println( number + "는 회문수 입니다.");
        else
            System.out.println( number + "는 회문수가 아닙니다.");
   }  // main
}
```

[실행결과]

```
12321는 회문수 입니다.
```

[정답]

```
result = result*10 + tmp % 10;
tmp /= 10;
```

[해설] 숫자를 역순으로 바꾼 후 원래의 숫자와 비교해서 같으면 회문수이다.예를 들어 원래의 숫자(number)의 값이 12345라면, 역순으로 바꾸면 54321이 될 것이다. 어떻게 하면 12345를 54321로 바꿀 수 있을까? 각 자리수의 값을 더하는 문제4-10과 같은 방식, 10으로 나눠가면서 10으로 나머지 연산을 하는 방식으로 각 자리수를 얻을 수 있다. 다만 그냥 더하는 게 아니라 10을 곱해가면서 더하면 숫자를 역순으로 바꿀 수 있다.

result	result*10	tmp	tmp%10
0	0	12345	5
5	50	1234	4
54	540	123	3
543	5430	12	2
5432	54320	1	1
54321	-	0	-

Chapter 5

배 열

Array

[연습문제 - 모범답안]

[5-1] 다음은 배열을 선언하거나 초기화 한 것이다. 잘못된 것을 고르고 그 이유를 설명하시오.

a. int[] arr[];
b. int[] arr = {1,2,3,}; // 마지막의 쉼표 ',' 는 있어도 상관없음.
c. int[] arr = new int[5];
d. int[] arr = new int[5]{1,2,3,4,5}; // 두 번째 대괄호[]에 숫자 넣으면 안됨.
e. int arr[5]; // 배열을 선언할 때는 배열의 크기를 지정할 수 없음.
f. int[] arr[] = new int[3][];

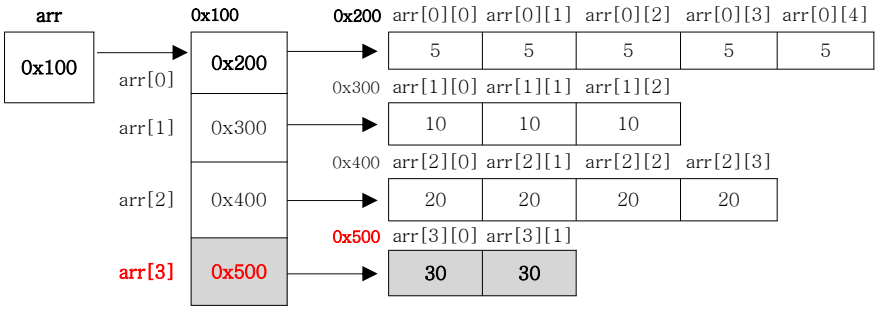
[정답] d, e
[해설] d. int[] arr = new int[]{1,2,3,4,5}에서는 대괄호[]안에 배열의 크기를 지정할 수 없다. 괄호{}안의 데이터의 개수에 따라 자동적으로 결정되기 때문이다.

[5-2] 다음과 같은 배열이 있을 때, arr[3].length의 값은 얼마인가?

```
int[][] arr = {
        {  5, 5, 5, 5, 5},
        { 10, 10, 10},
        { 20, 20, 20, 20},
        { 30, 30}
};
```

[정답] 2

[해설] 위와 같은 코드가 실행되면 다음과 같은 그림의 배열이 생성된다.



arr[3].length는 arr[3]이 가리키는 배열의 크기를 의미한다. 위의 그림에서 arr[3]이 가리키는 배열은 0x500번지에 있는 배열이며 크기는 2이다. 그래서 arr[3].length의 값은 2가 된다. 참고로 arr.length의 값은 4이고, arr[0].length의 값은 5, arr[1].length의 값은 3, arr[2].length의 값은 4이다.

[5-3] 배열 arr에 담긴 모든 값을 더하는 프로그램을 완성하시오.

[연습문제]/ch5/Exercise5_3.java

```
class Exercise5_3
{
     public static void main(String[] args)
     {
       int[] arr = {10, 20, 30, 40, 50};
       int sum = 0;

       for(int i=0;i<arr.length;i++) {
           sum += arr[i];
       }

       System.out.println("sum="+sum);
   }
}
```

[실행결과]

```
sum=150
```

[정답]

```
for(int i=0;i<arr.length;i++) {
    sum += arr[i];
}
```

[해설] 간단한 문제라서 별도의 설명은 생략함.

[5-4] 2차원 배열 arr에 담긴 모든 값의 총합과 평균을 구하는 프로그램을 완성하시오.

[연습문제]/ch5/Exercise5_4.java

```
class Exercise5_4
{
      public static void main(String[] args)
      {
        int[][] arr = {
             { 5, 5, 5, 5, 5},
             {10,10,10,10,10},
             {20,20,20,20,20},
             {30,30,30,30,30}
        };

        int total = 0;
        float average = 0;

        for(int i=0; i < arr.length;i++) {
             for(int j=0; j < arr[i].length;j++) {
                  total += arr[i][j];
             }
        }

        average = total /(float)(arr.length * arr[0].length);
        System.out.println("total="+total);
        System.out.println("average="+average);
    } // end of main
} // end of class
```

[실행결과]

```
total=325
average=16.25
```

[해설] 이번에도 배열과 반복문을 이용하는 문제인데, 2차원 배열이라 2중 for문을 사용해야한다는 것을 제외하고는 이전 문제와 다르지 않다.

평균을 구할 때는 배열의 모든 요소의 총합을 개수로 나누면 되는데, int로 나누면 int 나누기 int이기 때문에 결과를 int로 얻으므로 소수점 이하의 값을 얻을 수 없다. 그래서 나누는 값을 float로 형변환 해주었다. 만일 float로 형변환을 해주지 않으면 avaerage는 16.25가 아닌 16.00이 될 것이다.(average의 타입이 float이므로 16을 저장하면 16.00이 된다.)

1. int형(4 byte)보다 크기가 작은 자료형은 int형으로 형변환 후에 연산을 수행한다.
   byte / short → int / int → int
2. 두 개의 피연산자 중 자료형의 표현범위가 큰 쪽에 맞춰서 형변환 된 후 연산을 수행한다.
   int / float → float / float → float
3. 정수형 간의 나눗셈에서 0 으로 나누는 것은 금지되어 있다.

[5-5] 다음은 1과 9사이의 중복되지 않은 숫자로 이루어진 3자리 숫자를 만들어내는 프로그램이다. (1)~(2)에 알맞은 코드를 넣어서 프로그램을 완성하시오.

[참고] Math.random()을 사용했기 때문에 실행결과와 다를 수 있다.

[연습문제]/ch5/Exercise5_5.java

```
class Exercise5_5 {
   public static void main(String[] args) {
        int[] ballArr = {1,2,3,4,5,6,7,8,9};
        int[] ball3   = new int[3];

        // 배열 ballArr의 임의의 요소를 골라서 위치를 바꾼다.
        for(int i=0; i< ballArr.length;i++) {
            int j = (int)(Math.random() * ballArr.length);
            int tmp = 0;

            tmp = ballArr[i];
            ballArr[i] = ballArr[j];
            ballArr[j] = tmp;
        }

        // 배열 ballArr의 앞에서 3개의 수를 배열 ball3로 복사한다.
        System.arraycopy(ballArr,0, ball3,0,3);

        for(int i=0;i<ball3.length;i++) {
            System.out.print(ball3[i]);
        }
        System.out.println();
   } // end of main
} // end of class
```

[실행결과]

```
486
```

[정답]

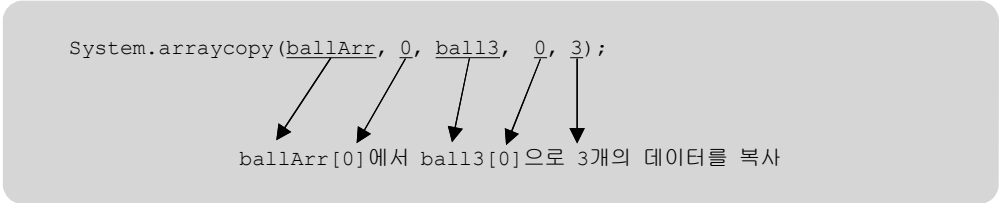
(1)

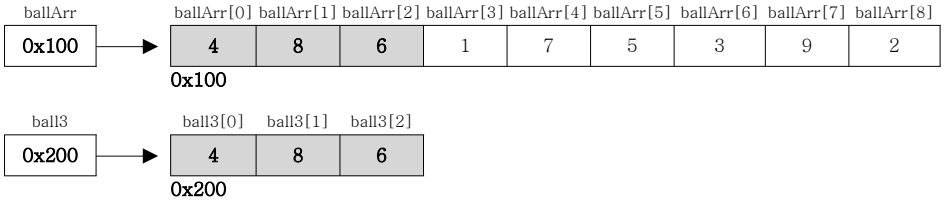
```
tmp = ballArr[i];
ballArr[i] = ballArr[j];
ballArr[j] = tmp;
```

(2)

```
System.arraycopy(ballArr,0, ball3,0,3);
```

[해설] 1~9의 숫자를 배열에 순서대로 담고, 반복해서 위치를 서로 바꿈으로써 숫자를 섞는다. 그 다음에 배열의 세 요소를 차례대로 가져오면 중복되지 않은 세 개의 정수를 얻을 수 있다. 그 다음엔 배열 ballArr에서 세 개의 값을 배열 ball3으로 복사한다. 편의상 맨 앞의 세 값을 ball3로 복사하기로 하자.




ballArr
0x100
ballArr[0] ballArr[1] ballArr[2] ballArr[3] ballArr[4] ballArr[5] ballArr[6] ballArr[7] ballArr[8]
4 8 6 1 7 5 3 9 2
0x100
ball3
0x200
ball3[0] ball3[1] ball3[2]
4 8 6
0x200

[5-6] 다음은 거스름돈을 몇 개의 동전으로 지불할 수 있는지를 계산하는 문제이다. 변수 money의 금액을 동전으로 바꾸었을 때 각각 몇 개의 동전이 필요한지 계산해서 출력하라. 단, 가능한 한 적은 수의 동전으로 거슬러 주어야한다. (1)에 알맞은 코드를 넣어서 프로그램을 완성하시오.

[Hint] 나눗셈 연산자와 나머지 연산자를 사용해야 한다.

[연습문제]/ch5/Exercise5_6.java

```
class Exercise5_6 {
   public static void main(String args[]) {
        // 큰 금액의 동전을 우선적으로 거슬러 줘야한다.
        int[] coinUnit = {500, 100, 50, 10};

        int money = 2680;
        System.out.println("money="+money);

        for(int i=0;i<coinUnit.length;i++) {
            System.out.println(coinUnit[i]+"원: "+money/coinUnit[i]);
            money = money%coinUnit[i];
        }
   } // main
}
```

[실행결과]

```
money=2680
500원: 5
100원: 1
50원: 1
10원: 3
```

[정답]

```
System.out.println(coinUnit[i]+"원: "+money/coinUnit[i]);
money = money%coinUnit[i];
```

[해설] 동전의 단위를 내림차순으로 배열에 초기화한다. 금액이 큰 동전을 우선적으로 지불해야 가장 적은 동전의 개수로 거스름돈을 줄 수 있기 때문이다. 그렇지 않으면, 모든 거스름돈을 10원짜리로만 주게 될 수도 있다.

변수 money를 coinUnit[i]로 나누면 지불할 동전의 개수가 되고, 나머지 연산을 하면 coinUnit[i]로 지불하고 남은 금액이 된다. 동전단위(coinUnit배열)의 개수만큼 이 과정을 반복하면 된다.

money	coinUnit[i]	money/coinUnit[i]	money%coinUnit[i]
2680	500	5	180
180	100	1	80
80	50	1	30
30	10	3	0

[5-7] 문제 5-6에 동전의 개수를 추가한 프로그램이다. 커맨드라인으로부터 거슬러 줄 금액을 입력받아 계산한다. 보유한 동전의 개수로 거스름돈을 지불할 수 없으면, '거스름돈이 부족합니다.'라고 출력하고 종료한다. 지불할 돈이 충분히 있으면, 거스름돈을 지불한 만큼 가진 돈에서 빼고 남은 동전의 개수를 화면에 출력한다. (1)에 알맞은 코드를 넣어서 프로그램을 완성하시오.

[연습문제]/ch5/Exercise5_7.java

```
class Exercise5_7
{
   public static void main(String args[])
  {
       if(args.length!=1) {
           System.out.println("USAGE: java Exercise5_7 3120");
           System.exit(0);
       }

      // 문자열을 숫자로 변환한다. 입력한 값이 숫자가 아닐 경우 예외가 발생한다.
       int money = Integer.parseInt(args[0]);

       System.out.println("money="+money);

       int[] coinUnit = {500, 100, 50, 10 }; // 동전의 단위
       int[] coin     = {5, 5, 5, 5};        // 단위별 동전의 개수


       for(int i=0;i<coinUnit.length;i++) {
           int coinNum = 0;

           // 1. 금액(money)을 동전단위로 나눠서 필요한 동전의 개수(coinNum)를 구한다.
           coinNum = money/coinUnit[i];

           // 2. 배열 coin에서 coinNum만큼의 동전을 뺀다.
           //   (만일 충분한 동전이 없다면 배열 coin에 있는 만큼만 뺀다.)
           if(coin[i] >= coinNum) {
               coin[i] -= coinNum;
           } else {
               coinNum = coin[i];
               coin[i] = 0;
           }

           // 3. 금액에서 동전의 개수(coinNum)와 동전단위를 곱한 값을 뺀다.
           money -= coinNum*coinUnit[i];

           System.out.println(coinUnit[i]+"원: "+coinNum);
       }

       if(money > 0) {
           System.out.println("거스름돈이 부족합니다.");
           System.exit(0);  // 프로그램을 종료한다.
       }

       System.out.println("=남은 동전의 개수 =");

       for(int i=0;i<coinUnit.length;i++) {
```

```
            System.out.println(coinUnit[i]+"원:"+coin[i]);
        }
    } // main
}
```

[실행결과]

```
C:\jdk1.8\work\ch5>java Exercise5_7
USAGE: java Exercise5_7 3120

C:\jdk1.8\work\ch5>java Exercise5_7 3170
money=3170
500원: 5
100원: 5
50원: 3
10원: 2
=남은 동전의 개수 =
500원:0
100원:0
50원:2
10원:3

C:\jdk1.8\work\ch5>java Exercise5_7 3510
money=3510
500원: 5
100원: 5
50원: 5
10원: 5
거스름돈이 부족합니다.
```

[정답]

```
// 1. 금액(money)을 동전단위로 나눠서 필요한 동전의 개수(coinNum)를 구한다.
coinNum = money/coinUnit[i];

// 2. 배열 coin에서 coinNum만큼의 동전을 뺀다.
//   (만일 충분한 동전이 없다면 배열 coin에 있는 만큼만 뺀다.)
if(coin[i] >= coinNum) {
    coin[i] -= coinNum;
} else {
    coinNum = coin[i];
    coin[i] = 0;
}

// 3. 금액에서 동전의 개수(coinNum)와 동전단위를 곱한 값을 뺀다.
money -= coinNum*coinUnit[i];
```

[해설] 주어진 로직대로만 작성하면 별 어려움 없이 풀 수 있었을 것이라 생각한다. 문제 5-6을 이해했다면 이 문제도 쉽게 이해될 것이므로 자세한 설명은 생략한다.

이 예제를 발전시켜 자판기 프로그램을 작성해보면 좋은 공부가 될 것이다.

[5-8] 다음은 배열 answer에 담긴 데이터를 읽고 각 숫자의 개수를 세어서 개수만큼 '*'을 찍어서 그래프를 그리는 프로그램이다. (1)~(2)에 알맞은 코드를 넣어서 완성하시오.

[연습문제]/ch5/Exercise5_8.java

```
class Exercise5_8 {
     public static void main(String[] args) {
           int[] answer = { 1,4,4,3,1,4,4,2,1,3,2 };
           int[] counter = new int[4];

           for(int i=0; i < answer.length;i++) {
               counter[answer[i]-1]++;
           }

           for(int i=0; i < counter.length;i++) {
               System.out.print(counter[i]);

               for(int j=0; j < counter[i];j++) {
                   System.out.print("*"); // counter[i]의 값 만큼 '*'을 찍는다.
               }

               System.out.println();
           }
     }      // end of main
}      // end of class
```

[실행결과]

```
3***
2**
2**
4****
```

[정답]

(1) `counter[answer[i]-1]++;`

(2)

```
System.out.print(counter[i]);

for(int j=0; j < counter[i];j++) {
    System.out.print("*");
}
```

[해설] 배열을 이용해서 데이터의 개수를 세는 문제이다. 1~4범위의 데이터를 집계할 것이기 때문에 크기가 4인 배열(counter)을 생성하였다. 크기가 4이지만 배열의 index는 0~3이기 때문에 answer[i]에서 1을 빼주어야 한다.

```
counter[answer[i]-1]++;
```

Java의 정석 3판의 예제5-11과 거의 동일하므로 자세한 설명은 생략한다. 다만, Java의 정석 소스 압축파일에 있는 플래시 동영상'/flash/Array.swf'을 보면 단계별로 자세히 설명되어 있으니 참고하기 바란다.

Java의 정석 소스압축파일은 http://cafe.naver.com/javachobostudy 에서 얻을 수 있다.

[5-9] 주어진 배열을 시계방향으로 90도 회전시켜서 출력하는 프로그램을 완성하시오.

[연습문제]/ch5/Exercise5_9.java

```
class Exercise5_9 {
     public static void main(String[] args) {
       char[][] star = {
           {'*','*',' ',' ',' '},
           {'*','*',' ',' ',' '},
           {'*','*','*','*','*'},
           {'*','*','*','*','*'}
       };

       char[][] result = new char[star[0].length][star.length];

       for(int i=0; i < star.length;i++) {
           for(int j=0; j < star[i].length;j++) {
               System.out.print(star[i][j]);
           }
           System.out.println();
       }

       System.out.println();

       for(int i=0; i < star.length;i++) {
           for(int j=0; j < star[i].length;j++) {
               int x = j;
               int y = star.length-1-i;

               result[x][y]=star[i][j];
           }
       }

       for(int i=0; i < result.length;i++) {
           for(int j=0; j < result[i].length;j++) {
               System.out.print(result[i][j]);
           }
           System.out.println();
       }
     } // end of main
} // end of class
```

[실행결과]

```
**
**
*****
*****

****
****
**
**
**
```

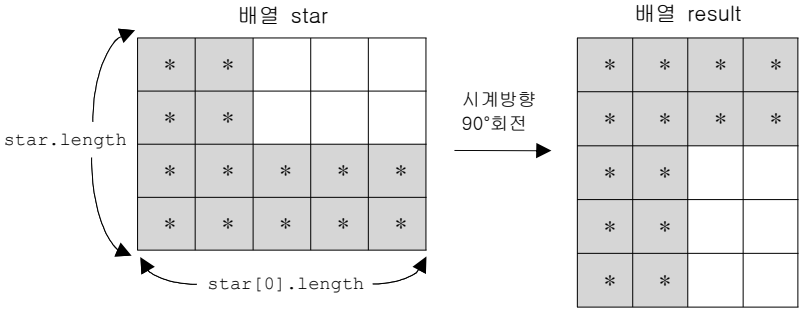
[정답]

```
int x = j;
int y = star.length-1-i;

result[x][y]=star[i][j];
```

[해설] 테트리스의 도형을 회전시키듯이 배열을 회전시키는 문제이다. 배열 star가 4×5의 2차원 배열이므로 이 배열을 90도 회전시키면 5×4의 2차원 배열이 되어야한다.

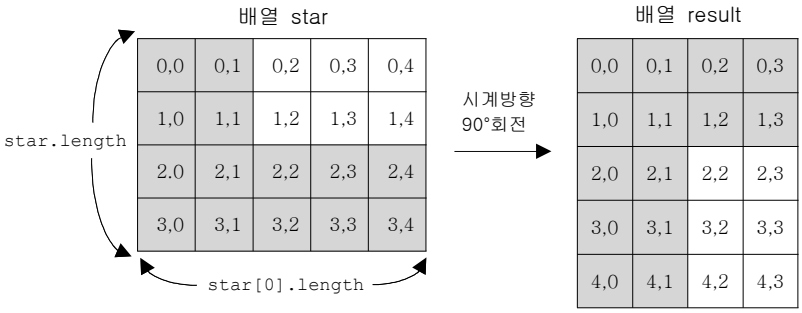
```
char[][] result = new char[star[0].length][star.length];
```

배열 star를 시계방향으로 회전시켜서 만든 배열 result를 그림으로 그려보면 다음과 같다.



result배열을 생성했으니, 이제 배열 star의 요소들을 배열 result의 적절한 위치로 옮기면 된다. 원래의 위치를 (i,j), 옮길 위치를 (x, y)라고 할 때 i와 j의 값을 이용해서 x와 y의 값을 어떻게 계산해 낼 것인지를 고민해보자.

```
for(int i=0; i < star.length;i++) {
    for(int j=0; j < star[i].length;j++) {
        int x = ???;
        int y = ???;

        result[x][y]=star[i][j]; // (i,j) → (x,y)
    }
}
```



위 그림은 배열의 인덱스를 표시한 것인데, star[0][0]은 result[0][3]으로, star[0][1]은 result[1][3]으로 이동해야 함을 알 수 있다. 이 것을 표로 정리하면 다음과 같다.

i	j
0	0
0	1
0	2
0	3
1	0
1	1
…	…
3	2
3	3
3	4

→

x	y
0	3
1	3
2	3
3	3
0	2
1	2
…	…
2	0
3	0
4	0

위의 표를 보면, x의 값은 j의 값과 정확히 일치함을 알 수 있다. 그래서 x의 값은 j의 값을 그대로 가져다 쓰면 된다. y의 값을 보면, i와 y의 합이 일정함을 알 수 있다. i와 y의 합은 항상 3이다. 3은 star.length-1과 동일한 값이다.

'i+y = star.length-1'이니까, 'y = star.length-1-i'라고 할 수 있다.

```
for(int i=0; i < star.length;i++) {
    for(int j=0; j < star[i].length;j++) {
        int x = j;
        int y = star.length-1-i;

        result[x][y]=star[i][j]; // (i,j) → (x,y)
    }
}
```

익숙하지 않겠지만 이렇게 그림을 그리고 표를 그리면 쉽게 해결된다. 코딩을 하기 전에 종이에 그림을 그리고 로직을 정리하는 것은 모니터를 보는 시간을 줄여서 눈의 피로를 적게 하고, 논리적 오류를 줄여줄 수 있다는 장점이 있다.

[5-10] 다음은 알파벳과 숫자를 아래에 주어진 암호표로 암호화하는 프로그램이다. (1)에 알맞은 코드를 넣어서 완성하시오.

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t
`	~	!	@	#	$	%	^	&	*	(	)	-	_	+	=	|	[	]	{

u	v	w	x	y	z
}	;	:	,	.	/

0	1	2	3	4	5	6	7	8	9
q	w	e	r	t	y	u	i	o	p

[연습문제]/ch5/Exercise5_10.java

```
class Exercise5_10 {
   public static void main(String[] args)     {
        char[] abcCode =
           { '`','~','!','@','#','$','%','^','&','*',
             '(',')','-','_','+','=','|','[',']','{',
             '}',';',':',',','.','/'};
                        // 0   1   2   3   4   5   6   7   8   9
        char[] numCode = {'q','w','e','r','t','y','u','i','o','p'};

        String src = "abc123";
        String result = "";

        // 문자열 src의 문자를 charAt()으로 하나씩 읽어서 변환 후 result에 저장
        for(int i=0; i < src.length();i++) {
            char ch = src.charAt(i);

            if('a' <= ch && ch <='z') {
                result += abcCode[ch-'a'];
            } else if('0' <= ch && ch <='9') {
                result += numCode[ch-'0'];
            }
        }

        System.out.println("src:"+src);
        System.out.println("result:"+result);

   } // end of main
} // end of class
```

[실행결과]

```
src:abc123
result:`~!wer
```

[정답]

```
if('a' <= ch && ch <='z') {
    result += abcCode[ch-'a'];
} else if('0' <= ch && ch <='9') {
    result += numCode[ch-'0'];
}
```

[해설] 문자열을 반복문과 chatAt(int i)을 이용해서, 한 문자씩 배열에 있는 암호코드로 변경해서 암호화하는 문제이다.

암호코드는 영어소문자와 숫자로 나뉘어져 있는데, 영어소문자인 경우 배열 abcCode에서 해당 암호코드를 얻고, 숫자인 경우에는 배열 numCode에서 암호코드를 얻도록 되어 있다.

그래서 조건문으로 문자가 영어소문자인 경우와 숫자인 경우를 나누어서 처리했다.

```
if('a' <= ch && ch <='z') {
    result += abcCode[ch-'a'];
} else if('0' <= ch && ch <='9') {
    result += numCode[ch-'0'];
}
```

암호코드배열 abcCode에는 문자 'a'시작해서 문자 'z'까지의 암호코드가 순서대로 저장되어 있기 때문에 문자 'a'의 암호코드는 abcCode[0]이고, 문자 'c'의 암호코드는 abcCode[2]이다. 즉, 영어소문자 ch의 암호코드는 abcCode[ch-'a']로 표현할 수 있는 것이다.

만일 문자 ch가 'c'였다면, 조건식 'a' <= ch && ch <='z'가 true가 되어 result+= abcCode[ch-'a';가 수행된다. 이 문장은 아래와 같은 순서로 연산이 진행된다.

```
result+= abcCode[ch-'a']; → result+= abcCode['c'-'a'];
→ result+= abcCode[2]; → result+= '!';
```

알파벳이나 숫자는 문자코드가 연속적으로 할당되어 있기 때문에, 'c'에서 'a'를 빼면 2를 결과로 얻는다.

```
'c'-'a' → 99 - 97 → 2
```

뺄셈과 같은 이항연산자는 int타입보다 작은 타입은 피연산자(byte, short, char)은 int로 변환한다. 그래서 'c'-'a'은 99 - 97로 변환되고 그 결과는 숫자 2가 된다. 참고로 문자 'a'와 'c'의 코드는 아래와 같다.

문자	문자코드
...	...
'a'	**97**
'b'	98
'c'	**99**
'd'	100
...	...

[5-11] 주어진 2차원 배열의 데이터보다 가로와 세로로 1이 더 큰 배열을 생성해서 배열의 행과 열의 마지막 요소에 각 열과 행의 총합을 저장하고 출력하는 프로그램이다. (1)에 알맞은 코드를 넣어서 완성하시오.

[연습문제]/ch5/Exercise5_11.java

```
class Exercise5_11
{
   public static void main(String[] args)
   {
       int[][] score = {
                  {100, 100, 100}
                , {20, 20, 20}
                , {30, 30, 30}
                , {40, 40, 40}
                , {50, 50, 50}
            };

       int[][] result = new int[score.length+1][score[0].length+1];

       for(int i=0; i < score.length;i++) {
           for(int j=0; j < score[i].length;j++) {
               result[i][j] = score[i][j];
               result[i][score[0].length] += result[i][j];
               result[score.length][j] += result[i][j];
               result[score.length][score[0].length] += result[i][j];
           }
       }

       for(int i=0; i < result.length;i++) {
           for(int j=0; j < result[i].length;j++) {
               System.out.printf("%4d",result[i][j]);
           }
           System.out.println();
       }
   } // main
}
```

[실행결과]

```
 100 100 100 300
  20  20  20  60
  30  30  30  90
  40  40  40 120
  50  50  50 150
 240 240 240 720
```

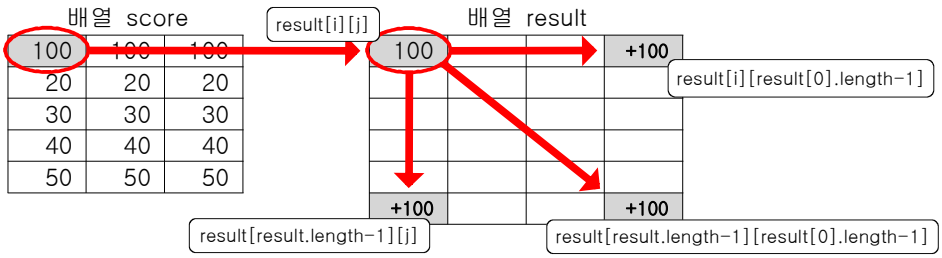
[정답]

```
result[i][j] = score[i][j];
result[i][score[0].length] += result[i][j];
result[score.length][j] += result[i][j];
result[score.length][score[0].length] += result[i][j];
```

[해설] 2차원 배열의 복사를 조금 응용한 문제이다. 2차원 배열 score에 담긴 값들을 2차원 배열 result에 복사하되 배열 result의 맨 오른쪽 열에는 각 행의 총합이, 그리고 맨

마지막 행에는 각 열의 총합이, 마지막으로 맨 오른쪽 행의 마지막 열에는 전체 총합이 저장되어야 한다. 언뜻 복잡해 보이지만 그림을 그려보면 간단명료해 진다.

먼저 score[i][j]를 result[i][j]에 저장하고, result[i][j]는 아래 그림에 표시한 세 곳에 누적해서 더해주면 된다.



result.length-1과 result[0].length-1은 score.length와 score[0].length와 각각 동일하므로 보다 식을 간단히 하기 위해 대체해서 사용했는데, 반드시 그렇게 해야 하는 것은 아니다.

```
result.length-1    → score.length
result[0].length-1 → score[0].length
```

score[i][j]의 값을 result[i][j]에 저장하기 때문에, result[i][j] 대신 score[i][j]를 사용해도 같은 결과를 얻는다.

```
result[i][score[0].length] += score[i][j];
result[score.length][j] += score[i][j];
result[score.length][score[0].length] += score[i][j];
```

[5-12] 예제5-23을 변경하여, 아래와 같은 결과가 나오도록 하시오.

[실행결과]

```
Q1. chair의 뜻은? dmlwk
틀렸습니다. 정답은 의자입니다

Q2. computer의 뜻은? 컴퓨터
정답입니다.

Q3. integer의 뜻은? 정수
정답입니다.

전체 3문제 중 2문제 맞추셨습니다.
```

[정답]

[연습문제]/ch5/Exercise5_12.java

```
import java.util.*;

class  Exercise5_12 {
   public static void main(String[] args) {
        String[][] words = {
            {"chair","의자"},          // words[0][0], words[0][1]
            {"computer","컴퓨터"},     // words[1][0], words[1][1]
            {"integer","정수"}         // words[2][0], words[2][1]
        };

        int score = 0; // 맞춘 문제의 수를 저장하기 위한 변수

        Scanner scanner = new Scanner(System.in);

        for(int i=0;i<words.length;i++) {
            System.out.printf("Q%d. %s의 뜻은?", i+1, words[i][0]);

            String tmp = scanner.nextLine();

            if(tmp.equals(words[i][1])) {
                System.out.printf("정답입니다.%n%n");
                score++;
            } else {
              System.out.printf("틀렸습니다. 정답은 %s입니다.%n%n",words[i][1]);
            }
        } // for

        System.out.printf("전체 %d문제 중 %d문제 맞추셨습니다.%n",
                                                    words.length, score);
   } // main의 끝
}
```

[해설] 맞춘 문제의 수를 저장하기 위한 변수 하나를 추가하고, 정답을 맞추면 이 변수의 값을 증가시키기만 하면 된다.

[5-13] 단어의 글자위치를 섞어서 보여주고 원래의 단어를 맞추는 예제이다. 실행결과와 같이 동작하도록 예제의 빈 곳을 채우시오.

[연습문제5-13]/ch5/Excercise5_13.java

```
import java.util.Scanner;

class Exercise5_13 {
   public static void main(String args[]) {
        String[] words = { "television", "computer", "mouse", "phone" };

        Scanner scanner = new Scanner(System.in);

        for(int i=0;i<words.length;i++) {
            char[] question = words[i].toCharArray(); // String을 char[]로 변환

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

                char tmp = question[i];
                question[i] = question[idx];
                question[idx] = tmp;
            }

            System.out.printf("Q%d. %s의 정답을 입력하세요.>",
                                                    i+1, new String(question));
            String answer = scanner.nextLine();

            // trim()으로 answer의 좌우 공백을 제거한 후, equals로 word[i]와 비교
            if(words[i].equals(answer.trim()))
                System.out.printf("맞았습니다.%n%n");
            else
                System.out.printf("틀렸습니다.%n%n");
        }
   } // main의 끝
}
```

[실행결과]

```
Q1. lvtsieeoin의 정답을 입력하세요.>television
맞았습니다.

Q2. otepcumr의 정답을 입력하세요.>computer
맞았습니다.

Q3. usemo의 정답을 입력하세요.>asdf
틀렸습니다.

Q4. ohpne의 정답을 입력하세요.>phone
맞았습니다.
```

[정답]

```
for(int j=0;j<question.length;j++) {
    int idx = (int)(Math.random() * question.length);
```

```
            char tmp = question[i];
            question[i] = question[idx];
            question[idx] = tmp;
        }
```

[해설] 예제5-8을 응용한 문제이다. 간단한 문제이므로 설명은 생략한다.

[연습문제 - 모범답안]

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

타 입	변수명	설 명
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 실행결과와 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 정의해야한다.

```
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 isKwan g)
{
    this.num = num;
    this.isKwang = isKwang;
}
```

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

SutdaCard(int num, boolean isKwan g)
{
    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 card인스턴스의 정보를 문자열로 반환하기 위한 것이다. card인스턴스의 멤버변수 num와 isKwang의 값을 문자열로 만들어서 반환하면 된다. isKwang의 값이 true인 경우에는 숫자 뒤에 "K"를 붙이도록 삼항연산자를 사용했다.

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

변수 num은 타입이 int이지만 문자열과 덧셈연산을 하기 때문에 최종적으로는 문자열을 반환하게 된다.

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

타 입	변수명	설 명
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 값들을 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 Math클래스의 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 MyPoint클래스의 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, y를 사용해서 작업하므로 매개변수로 x1과 y1만을 제공받으면 된다. 인스턴스와 관계가 있으므로(인스턴스 변수를 사용했으니까) static을 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 지역변수이다.

```
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 방어력은 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 값이어야 하고, 병사들마다 부상의 정도가 다를 것이므로 병사들의 체력(hp) 역시 개별적인 값이어야 한다. 그러나 모든 병사들의 공격력과 방어력은 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.

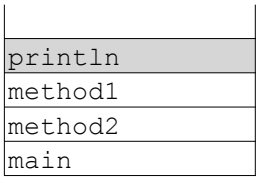
d. 클래스변수나 인스턴스변수보다 메모리 부담이 적다.

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)



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 - static메서드에서는 인스턴스변수를 사용할 수 없다.
라인 D - static메서드에서는 인스턴스메서드를 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]

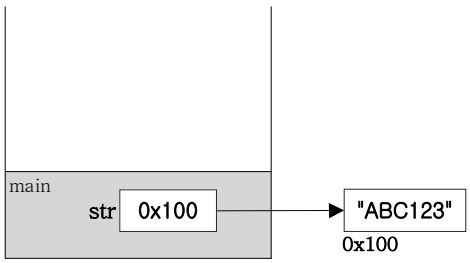
```
ABC123
After change:ABC123
```

[해설] change메서드의 매개변수가 참조형인데도 왜? main메서드의 문자열 str에 변경한 내용이 반영되지 않은 것일까? 많은 사람들이 매개변수가 참조형이라는 것만 보고 main메서드의 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.

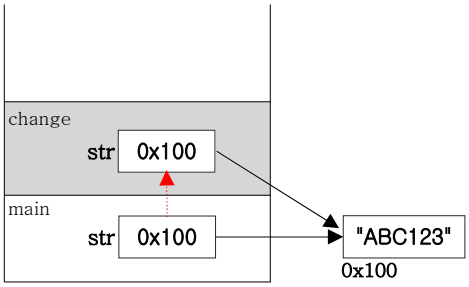
처음에 문자열을 참조변수 str에 저장하면 아래와 같은 그림이 된다.

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



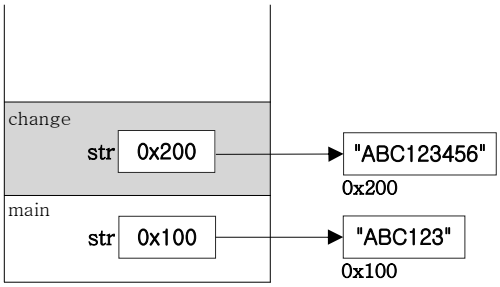
그 다음에 메서드 change를 호출하면서 참조변수 str을 넘겨주면, 메서드 change의 지역변수 str에 주소값 0x100이 저장된다. 이제 메서드 change의 지역변수 str도 문자열 "ABC123"을 참조하게 된다. 이 두 참조변수는 이름은 같지만 분명히 다른 변수 이다. 서로 다른 영역에 존재하기 때문에 이름이 같아도 상관없는 것이다.

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



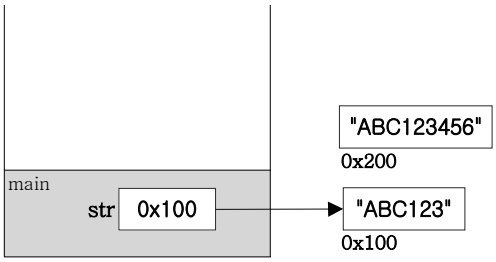
메서드 change에서는 넘겨받은 문자열의 뒤에 "456"을 붙인다. 문자열은 내용을 변경할 수 없기 때문에 덧셈연산을 하면 새로운 문자열이 생성되고 새로운 문자열의 주소가 변수 str에 저장된다.

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



이제 change메서드는 종료되고, 작업에 사용하던 메모리를 반환하므로 change메서드의 지역변수인 str역시 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;
        }

        return arr;
    }
```

[해설] int배열을 매개변수로 받아서 배열에 저장된 각 요소들의 위치를 여러번 바꿔서 섞은 다음 반환하는 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.cafe의 게시판에 올려주길 바란다.

[6-22] 다음과 같이 정의된 메서드를 작성하고 테스트하시오.

메서드명 : isNumber
기　　능 : 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 유효성 체크는 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 char타입의 변수 ch에 저장한다.

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

읽어온 문자(ch)가 숫자가 아니면 false를 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 -999999을 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 i번 째 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.
    }
}
```

[연습문제 - 모범답안]

[7-1] 섯다카드 20장을 포함하는 섯다카드 한 벌(SutdaDeck클래스)을 정의한 것이다. 섯다카드 20장을 담는 SutdaCard배열을 초기화하시오. 단, 섯다카드는 1부터 10까지의 숫자가 적힌 카드가 한 쌍씩 있고, 숫자가 1, 3, 8인 경우에는 둘 중의 한 장은 광(Kwang)이어야 한다. 즉, SutdaCard의 인스턴스변수 isKwang의 값이 true이어야 한다.

[연습문제]/ch7/Exercise7_1.java

```
class SutdaDeck {
   final int CARD_NUM = 20;
   SutdaCard[] cards = new SutdaCard[CARD_NUM];

   SutdaDeck() {
       for(int i=0;i < cards.length;i++) {
           int num = i%10+1;
           boolean isKwang = (i < 10)&&(num==1||num==3||num==8);

           cards[i] = new SutdaCard(num,isKwang);
       }
   }
}

class SutdaCard {
   int num;
   boolean isKwang;

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

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

   // info()대신 Object클래스의 toString()을 오버라이딩했다.
   public String toString() {
       return num + ( isKwang ? "K":"");
   }
}

class Exercise7_1 {
   public static void main(String args[]) {
       SutdaDeck deck = new SutdaDeck();

       for(int i=0; i < deck.cards.length;i++)
           System.out.print(deck.cards[i]+",");
   }
}
```

[실행결과]

```
1K,2,3K,4,5,6,7,8K,9,10,1,2,3,4,5,6,7,8,9,10,
```

[해설] SutdaDeck클래스에 cards라는 SutdaCard배열이 정의되어 있다. 이 배열을 생성했다고 해서 SutdarCard인스턴스가 생성된 것은 아니다. 그저 SutdaCard인스턴스를 저장하기 위한 공간을 생성한 것일 뿐이다. 객체배열을 생성할 때, 배열만 생성해 놓고 객체를 생성하지 않는 실수를 하지 않도록 주의하자.

```
SutdaCard[] cards = new SutdaCard[CARD_NUM];
```

생성자를 통해 객체배열 SutdaCard에 SutdaCard인스턴스를 생성해서 저장할 차례다. 아래와 같이 반복문을 이용해서 배열의 크기만큼 SutdaCard인스턴스를 생성하면 되는데, 이때 num의 값과 isKwang의 값을 어떻게 계산해 낼 것인지를 고민해야한다.

```
SutdaDeck() {
    for(int i=0;i < cards.length;i++) {
        int     num = ???;
        boolean isKwang = ???;

        cards[i] = new SutdaCard(num,isKwang);
    }
}
```

아래의 표에서 볼 수 있는 것 처럼, i의 값이 0~19까지 변하는 동안 우리가 원하는 num의 값을 얻기 위해서는 i%10+1과 같은 계산식을 사용하면 된다.

i	i%10	i%10+1
0	0	1
1	1	2
2	2	3
3	3	4
4	4	5
5	5	6
6	6	7
7	7	8
8	8	9
9	9	10
10	0	1
11	1	2
12	2	3
13	3	4
14	4	5
15	5	6
16	6	7
17	7	8
18	8	9
19	9	10

그리고 num의 값이 1,3,8일 때, 한 쌍의 카드 중에서 하나는 광(kwang)이어야하므로 아래와 같은 조건식이 필요하다. AND(&&)가 OR(||)보다 우선순위가 높기 때문에 괄호를 꼭 사용해야한다.

```
boolean isKwang = (i < 10)&&(num==1||num==3||num==8);
```

만일 i의 값이 20이고 num의 값이 30이라면 위의 조건식은 다음과 같은 계산과정을 거쳐서 isKwang에는 true가 저장된다.

```
    boolean isKwang = (2 < 10)&&(3==1||3==3||3==8);
→   boolean isKwang = (true)&&(false||true||false);
→   boolean isKwang = (true)&&(true||false);
→   boolean isKwang = (true)&&(true);
→   boolean isKwang = true;
```

[7-2] 문제7-1의 SutdaDeck클래스에 다음에 정의된 새로운 메서드를 추가하고 테스트 하시오.

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

1. 메서드명 : shuffle
   기　능 : 배열 cards에 담긴 카드의 위치를 뒤섞는다.(Math.random()사용)
   반환타입 : 없음
   매개변수 : 없음

2. 메서드명 : pick
   기　능 : 배열 cards에서 지정된 위치의 SutdaCard를 반환한다.
   반환타입 : SutdaCard
   매개변수 : int index - 위치

3. 메서드명 : pick
   기　능 : 배열 cards에서 임의의 위치의 SutdaCard를 반환한다.(Math.random()사용)
   반환타입 : SutdaCard
   매개변수 : 없음

[연습문제]/ch7/Exercise7_2.java

```
class SutdaDeck {
   final int CARD_NUM = 20;
   SutdaCard[] cards = new SutdaCard[CARD_NUM];

   SutdaDeck() {
       for(int i=0;i < cards.length;i++) {
           int num = i%10+1;
           boolean isKwang = (i < 10)&&(num==1||num==3||num==8);

           cards[i] = new SutdaCard(num,isKwang);
       }
   }

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

           // cards[i]와 cards[j]의 값을 서로 바꾼다.
           SutdaCard tmp  = cards[i];
           cards[i] = cards[j];
           cards[j] = tmp;
       }
   }

   SutdaCard pick(int index) {
       if(index < 0 || index >= CARD_NUM)  // index의 유효성을 검사한다.
           return null;
       return cards[index];
   }
```

```
    SutdaCard pick() {
        int index = (int)(Math.random()*cards.length);
        return pick(index); // pick(int index)를 호출한다.
    }
} // SutdaDeck

class SutdaCard {
    int num;
    boolean isKwang;

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

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

    public String toString() {
        return num + ( isKwang ? "K":"");
    }
}

class Exercise7_2 {
    public static void main(String args[]) {
        SutdaDeck deck = new SutdaDeck();

        System.out.println(deck.pick(0));
        System.out.println(deck.pick());
        deck.shuffle();

        for(int i=0; i < deck.cards.length;i++)
            System.out.print(deck.cards[i]+",");

        System.out.println();
        System.out.println(deck.pick(0));
    }
}
```

[실행결과]

```
1K
7
2,6,10,1K,7,3,10,5,7,8,5,1,2,9,6,9,4,8K,4,3K,
2
```

[해설] shuffle메서드에 대한 것은 이미 문제6-20에서 이미 설명했으므로 설명을 생략하겠다. pick(int index)메서드는 매개변수 index에 대한 유효성검사가 필요하다. 그렇지 않으면 배열의 index범위를 넘어서서 ArrayIndexOutOfBounndsException이 발생할 수 있다. 매개변수가 있는 메서드는 반드시 작업 전에 유효성검사를 해야 한다는 것을 기억하자.

```
SutdaCard pick(int index) {
        if(index < 0 || index >= CARD_NUM)  // index의 유효성을 검사한다.
            return null;
        return cards[index];
    }

    SutdaCard pick() {
        int index = (int)(Math.random()*cards.length);
        return pick(index); // pick(int index)를 호출한다.
    }
```

pick()메서드의 경우, cards배열에 있는 임의의 카드를 꺼내야하므로 Math.random()을 이용해서 유효한 index범위 내의 한 값을 얻어서 다시 pick(int index)메서드를 호출한다. 다소 비효율적이지만 코드의 중복을 제거하고 재사용성을 높이기 위해 이처럼 하는 것이다. 그러나 너무 객체지향적인 측면에 얽매여서 코드를 짤 필요는 없다고 생각한다. 상황에 맞는 적절한 코드를 작성하면 그것으로 좋지 않을까.

```
    SutdaCard pick() {
        int index = (int)(Math.random()*cards.length);
        return cards[index];
    }
```

[7-3] 오버라이딩의 정의와 필요성에 대해서 설명하시오.

[정답] 오버라이딩(overriding)이란, '조상 클래스로부터 상속받은 메서드를 자손 클래스에 맞게 재정의 하는 것'을 말한다.
조상 클래스로부터 상속받은 메서드를 자손 클래스에서 그대로 사용할 수 없는 경우가 많기 때문에 오버라이딩이 필요하다.

[7-4] 다음 중 오버라이딩의 조건으로 옳지 않은 것은? (모두 고르시오)

a. 조상의 메서드와 이름이 같아야 한다.

b. 매개변수의 수와 타입이 모두 같아야 한다.

c. 접근 제어자는 조상의 메서드보다 좁은 범위로만 변경할 수 있다.

d. 조상의 메서드보다 더 많은 수의 예외를 선언할 수 있다.

[정답] c, d

[해설]

자손 클래스에서 오버라이딩하는 메서드는 조상 클래스의 메서드와

- **이름이 같아야 한다.**
- **매개변수가 같아야 한다.**
- **리턴타입이 같아야 한다.**

[참고] JDK1.5부터 '공변 반환타입(covariant return type)'이 추가되어, 반환타입을 자손 클래스의 타입으로 변경하는 것은 가능하도록 조건이 완화되었다. p.457

조상 클래스의 메서드를 자손 클래스에서 오버라이딩할 때

1. **접근 제어자를 조상 클래스의 메서드보다 좁은 범위로 변경할 수 없다.**
2. **예외는 조상 클래스의 메서드보다 많이 선언할 수 없다.**
3. **인스턴스메서드를 static메서드로 또는 그 반대로 변경할 수 없다.**

[7-5] 다음의 코드는 컴파일하면 에러가 발생한다. 그 이유를 설명하고 에러를 수정하기 위해서는 코드를 어떻게 바꾸어야 하는가?

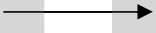
[연습문제]/ch7/Exercise7_5.java

```
class Product
{
   int price;        // 제품의 가격
   int bonusPoint;   // 제품구매 시 제공하는 보너스점수

   Product() {}

   Product(int price) {
        this.price = price;
        bonusPoint =(int)(price/10.0);
   }
}

class Tv extends Product {
   Tv() {}

   public String toString() {
        return "Tv";
   }
}

class Exercise7_5 {
   public static void main(String[] args) {
        Tv t = new Tv();
   }
}
```

[정답] Product클래스에 기본 생성자 Product()가 없기 때문에 에러가 발생한다. Product클래스에 기본 생성자 Product() {}를 추가해 줘야한다.

[해설] Tv클래스의 인스턴스를 생성할 때, 생성자 Tv()가 호출되고 Tv()는 조상 생성자 super()를 호출한다. 실제 코드에서는 super()를 호출하는 곳이 없지만 컴파일러가 자동적으로 추가해 준다. 그래서 컴파일을 하고 나면 아래의 오른쪽 코드와 같이 변경된다.

```
Tv() {}
```



```
Tv() {
    super(); //Product()를 호출
}
```

추가된 super()는 조상클래스인 Product의 기본 생성자 Product()를 호출하는 것인데, Product클래스에는 기본 생성자 Product()가 정의되어 있지 않다. 정의되어 있지 않은 생성자를 호출하니까 에러가 발생하는 것이다. Product클래스에는 이미 Product(int price)라는 생성자가 정의되어 있기 때문에 컴파일러가 자동적으로 추가해 주지도 않으므로 직접 Product클래스에 Product(){}를 넣어주면 문제가 해결된다.

[7-6] 자손 클래스의 생성자에서 조상 클래스의 생성자를 호출해야하는 이유는 무엇인가?

[정답] 조상에 정의된 인스턴스 변수들이 초기화되도록 하기 위해서.
[해설] 자손클래스의 인스턴스를 생성하면 조상으로부터 상속받은 인스턴스변수들도 생성되는데, 이 상속받은 인스턴스변수들 역시 적절히 초기되어야 한다. 상속받은 조상의 인스턴스변수들을 자손의 생성자에서 직접 초기화하기보다는 조상의 생성자를 호출함으로써 초기화되도록 하는 것이 바람직하다.

각 클래스의 생성자는 해당 클래스에 선언된 인스턴스변수의 초기화만을 담당하고, 조상클래스로부터 상속받은 인스턴스변수의 초기화는 조상클래스의 생성자가 처리하도록 해야하는 것이다.

[7-7] 다음 코드의 실행했을 때 호출되는 생성자의 순서와 실행결과를 적으시오.

[연습문제]/ch7/Exercise7_7.java

```
class Parent {
   int x=100;

   Parent() {
       this(200);   // Parent(int x)를 호출
   }

   Parent(int x) {
       this.x = x;
   }

   int getX() {
       return x;
   }
}

class Child extends Parent {
   int x = 3000;

   Child() {
       this(1000); // Child(int x)를 호출
   }

   Child(int x) {
       this.x = x;
   }
}

class Exercise7_7 {
   public static void main(String[] args) {
       Child c = new Child();

       System.out.println("x="+c.getX());
   }
}
```

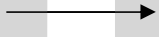
[정답] Child() → Child(int x) → Parent() → Parent(int x) → Object()의 순서로 호출된다.

[실행결과]

```
x=200
```

[해설] 컴파일러는 생성자의 첫 줄에 다른 생성자를 호출하지 않으면 조상의 기본 생성자를 호출하는 코드'super();'를 넣는다. 그래서 왼쪽의 코드는 컴파일 후 오른쪽과 같은 코드로 바뀐다. Child클래스의 조상은 Parent이므로 super()는 Parent()를 의미한다.

```
Child(int x) {
   this.x = x;
}
```



```
Child(int x) {
   super(); // Parent()를 호출
   this.x = x;
}
```

마찬가지로 Parent(int x) 역시 컴파일러가 Parent의 조상인 Object클래스의 기본 생성자를 호출하는 코드'super();'를 넣는다.

```
Parent(int x) {
   this.x = x;
}
```



```
Parent(int x) {
   super(); // Object()를 호출
   this.x = x;
}
```

Child() → Child(int x) → Parent() → Parent(int x) → Object()의 순서로 호출되니까, Child클래스의 인스턴스변수 x는 1000이 되고, Parent클래스의 인스턴스변수 x는 200이 된다. getX()는 조상인 Parent클래스에 정의된 것이라서, getX()에서 x는 Parent클래스의 인스턴스변수 x를 의미한다. 그래서 x=200이 출력된다.

[7-8] 다음 중 접근제어자를 접근범위가 넓은 것에서 좁은 것의 순으로 바르게 나열한 것은?

a. public-protected-(default)-private
b. public-(default)-protected-private
c. (default)-public-protected-private
d. private-protected-(default)-public

[정답] a
[해설]

접근 제어자가 사용될 수 있는 곳 - 클래스, 멤버변수, 메서드, 생성자

private - **같은 클래스** 내에서만 접근이 가능하다.
default - **같은 패키지** 내에서만 접근이 가능하다.
protected - 같은 패키지 내에서, 그리고 다른 패키지의 자손클래스에서 접근이 가능하다.
public - 접근 제한이 전혀 없다.

접근 범위가 넓은 쪽에서 좁은 쪽의 순으로 왼쪽부터 나열하면 다음과 같다.

public > protected > default > private

제어자	같은 클래스	같은 패키지	자손클래스	전 체
public				
protected				
default				
private				

[7-9] 다음 중 제어자 final을 붙일 수 있는 대상과 붙였을 때 그 의미를 적은 것이다. 옳지 않은 것은? (모두 고르시오)

a. 지역변수 - 값을 변경할 수 없다.

b. 클래스 - 상속을 통해 클래스에 새로운 멤버를 추가할 수 없다.

c. 메서드 - 오버로딩을 할 수 없다. ← 오버라이딩(overriding)을 할 수 없다.

d. 멤버변수 - 값을 변경할 수 없다.

[정답] c

[해설] 제어자 final은 '마지막의' 또는 '변경될 수 없는'의 의미를 가지고 있으며 거의 모든 대상에 사용될 수 있다.

제어자	대상	의 미
final	클래스	변경될 수 없는 클래스, 확장될 수 없는 클래스가 된다. 그래서 final로 지정된 클래스는 다른 클래스의 조상이 될 수 없다.
	메서드	변경될 수 없는 메서드, final로 지정된 메서드는 오버라이딩을 통해 재정의 될 수 없다.
	멤버변수	변수 앞에 final이 붙으면, 값을 변경할 수 없는 상수가 된다.
	지역변수	

[7-10] MyTv2클래스의 멤버변수 isPowerOn, channel, volume을 클래스 외부에서 접근할 수 없도록 제어자를 붙이고 대신 이 멤버변수들의 값을 어디서나 읽고 변경할 수 있도록 getter와 setter메서드를 추가하라.

[연습문제]/ch7/Exercise7_10.java

```
class MyTv2 {
    private boolean isPowerOn;
    private int     channel;
    private int     volume;

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

    public void setVolume(int volume){
        if(volume > MAX_VOLUME || volume < MIN_VOLUME)
            return;

        this.volume = volume;
    }

    public int getVolume(){
        return volume;
    }

    public void setChannel(int channel){
        if(channel > MAX_CHANNEL || channel < MIN_CHANNEL)
            return;

        this.channel = channel;
    }

    public int getChannel(){
        return channel;
    }
}

class Exercise7_10 {
    public static void main(String args[]) {
        MyTv2 t = new MyTv2();

        t.setChannel(10);
        System.out.println("CH:"+t.getChannel());
        t.setVolume(20);
        System.out.println("VOL:"+t.getVolume());
    }
}
```

[실행결과]

```
CH:10
VOL:20
```

[해설] 별로 어렵지 않은 문제라 별도의 설명이 필요없을 것이다. 다만 매개변수가 있는 메서드는 반드시 작업 전에 넘겨받은 값의 유효성검사를 해야 한다는 것을 잊지 말자.

[7-11] 문제7-10에서 작성한 MyTv2클래스에 이전 채널(previous channel)로 이동하는 기능의 메서드를 추가해서 실행결과와 같은 결과를 얻도록 하시오.

[Hint] 이전 채널의 값을 저장할 멤버변수를 정의하라.

메서드명 : gotoPrevChannel
기　　능 : 현재 채널을 이전 채널로 변경한다.
반환타입 : 없음
매개변수 : 없음

[연습문제]/ch7/Exercise7_11.java

```
class MyTv2 {
   private boolean isPowerOn;
   private int     channel;
   private int     volume;
   private int     prevChannel;   // 이전 채널(previous channel)

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

   public void setVolume(int volume){
       if(volume > MAX_VOLUME || volume < MIN_VOLUME)
           return;

       this.volume = volume;
   }

   public int getVolume(){
       return volume;
   }

   public void setChannel(int channel){
       if(channel > MAX_CHANNEL || channel < MIN_CHANNEL)
           return;

       prevChannel = this.channel;  // 현재 채널을 이전 채널에 저장한다.
       this.channel = channel;
   }

   public int getChannel(){
       return channel;
   }

   public void gotoPrevChannel() {
       setChannel(prevChannel); // 현재 채널을 이전 채널로 변경한다.
   }
}

class Exercise7_11 {
   public static void main(String args[]) {
       MyTv2 t = new MyTv2();
```

```
        t.setChannel(10);
        System.out.println("CH:"+t.getChannel());
        t.setChannel(20);
        System.out.println("CH:"+t.getChannel());
        t.gotoPrevChannel();
        System.out.println("CH:"+t.getChannel());
        t.gotoPrevChannel();
        System.out.println("CH:"+t.getChannel());
    }
}
```

[실행결과]

```
CH:10
CH:20
CH:10
CH:20
```

[해설] 먼저 이전 채널을 저장할 변수(prevChannel)를 하나 추가해야 한다. 그리고 채널이 바뀔 때마다 이 변수에 바뀌기 전의 채널을 저장해야 한다. 문제7-10의 코드에 아래의 붉은 색 코드를 추가했다.

```
public void setChannel(int channel){
    if(channel > MAX_CHANNEL || channel < MIN_CHANNEL)
        return;

    prevChannel = this.channel;  // 현재 채널을 이전 채널에 저장한다.
    this.channel = channel;
}
```

이제 gotoPrevChannel()에서는 setChannel()을 호출해주기만 하면 된다.

```
public void gotoPrevChannel() {
    setChannel(prevChannel); // 현재 체널을 이전 채널로 변경한다.
}
```

[7-12] 다음 중 접근 제어자에 대한 설명으로 옳지 않은 것은? (모두 고르시오)

a. public은 접근제한이 전혀 없는 접근 제어자이다.
b. (default)가 붙으면, 같은 패키지 내에서만 접근이 가능하다.
c. 지역변수에도 접근 제어자를 사용할 수 있다.
d. protected가 붙으면, 같은 패키지 내에서도 접근이 가능하다.
e. protected가 붙으면, 다른 패키지의 자손 클래스에서 접근이 가능하다.

[정답] c
[해설]

접근 제어자가 사용될 수 있는 곳 - 클래스, 멤버변수, 메서드, 생성자

private - **같은 클래스** 내에서만 접근이 가능하다.
default - **같은 패키지** 내에서만 접근이 가능하다.
protected - 같은 패키지 내에서, 그리고 다른 패키지의 자손클래스에서 접근이 가능하다.
public - 접근 제한이 전혀 없다.

제어자	같은 클래스	같은 패키지	자손클래스	전 체
public				
protected				
default				
private				

[7-13] Math클래스의 생성자는 접근 제어자가 private이다. 그 이유는 무엇인가?
[정답] Math클래스의 모든 메서드가 static메서드이고 인스턴스변수가 존재하지 않기 때문에 객체를 생성할 필요가 없기 때문

[해설] Math클래스는 몇 개의 상수와 static메서드만으로 구성되어 있기 때문에 인스턴스를 생성할 필요가 없다. 그래서 외부로부터의 불필요한 접근을 막기 위해 다음과 같이 생성자의 접근 제어자를 private으로 지정하였다.

```
public final class Math {
     private Math() {}
     //...
}
```

[7-14] 문제7-1에 나오는 섯다카드의 숫자와 종류(isKwang)는 사실 한번 값이 지정되면 변경되어서는 안 되는 값이다. 카드의 숫자가 한번 잘못 바뀌면 똑같은 카드가 두 장이 될 수 도 있기 때문이다. 이러한 문제점이 발생하지 않도록 아래의 SutdaCard를 수정하시오.

[연습문제]/ch7/Exercise7_14.java

```
class SutdaCard {
   final int NUM;
   final boolean IS_KWANG;

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

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

   public String toString() {
       return NUM + ( IS_KWANG ? "K":"");
   }
}

class Exercise7_14 {
   public static void main(String args[]) {
       SutdaCard card = new SutdaCard(1, true);
   }
}
```

[해설] 원래 변수 앞에 final을 붙일 때는 선언과 초기화를 동시에 해야 한다.

```
final int MAX_VOLUME  = 100;
```

그러나 인스턴스변수의 경우, 선언시에 초기화 하지 않고 생성자에서 초기화할 수 있다. 생성할 때 지정된 값이 변하지 않도록 할 수 있는 것이다. 상수이므로 한번 초기화한 이후로는 값을 바꿀 수 없다.

```
final int NUM;
final boolean IS_KWANG;

SutdaCard(int num, boolean isKwang) {
    this.NUM = num;           // 생성자에서 단 한 번의 초기화만 가능
    this.IS_KWANG = isKwang;  // 생성자에서 단 한 번의 초기화만 가능
}
```

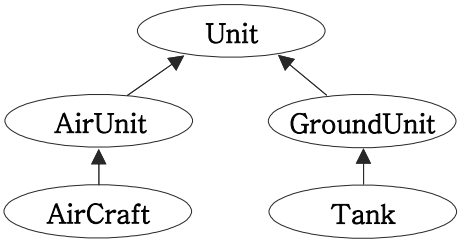
카드게임에서 카드의 숫자와 무늬가 게임도중에 변경되는 것이 가능하다면, 실수로 같은 카드가 두 장이 되는 일이 일어날 수 있기 때문에 이를 방지하기 위해서 숫자와 무늬는 한번 지정되면 변경할 수 없도록 하는 것이 바람직하다.

[7-15] 클래스가 다음과 같이 정의되어 있을 때, 형변환을 올바르게 하지 않은 것은? (모두 고르시오.)

```
class Unit {}
class AirUnit extends Unit {}
class GroundUnit extends Unit {}
class Tank extends GroundUnit {}
class AirCraft extends AirUnit {}

Unit u = new GroundUnit();
Tank t = new Tank();
AirCraft ac = new AirCraft();
```

a. u = (Unit)ac;
b. u = ac;
c. GroundUnit gu = (GroundUnit)u;
d. AirUnit au = ac;
e. t = (Tank)u; ← 조상타입의 인스턴스를 자손타입으로 형변환 할 수 없다.
f. GroundUnit gu = t;

[정답] e
[해설] 클래스간의 상속관계를 그림으로 그려보면 쉽게 알 수 있다.



Unit클래스는 나머지 네 개 클래스의 조상이므로 형변환이 가능하며, 심지어는 생략할 수도 있다.

```
AirCraft ac = new AirCraft();
u = (Unit)ac; // u는 AirCraft의 조상인 Unit타입이므로 형변환이 가능하다.
u = ac;       // 업캐스팅(자손→조상)이므로 형변환을 생략할 수 있다.
```

조상타입의 참조변수로 자손타입의 인스턴스를 참조하는 것이 가능하기 때문에 아래의 코드는 모두 가능하다.

```
Unit u = new GroundUnit();
GroundUnit gu = (GroundUnit)u; // u가 참조하는 객체가 GroundUnit이므로 OK
GroundUnit gu = (GroundUnit)new GroundUnit(); // 위의 두 줄을 한 줄로 합침

AirCraft ac = new AirCraft();
AirUnit au = ac; // AirCraft가 AirUnit의 자손이므로 가능. 형변환 생략됨
AirUnit au = new AirCraft(); // 위의 두 줄을 한 줄로 합치면 이렇게 쓸 수 있음

Tank t = new Tank();
GroundUnit gu = t;    // 조상타입의 참조변수로 자손타입의 인스턴스를 참조. OK
GroundUnit gu = new Tank(); // 위의 두 줄을 한 줄로 합치면 이렇게 쓸 수 있음
```

그러나 조상인스턴스를 자손타입으로 형변환하는 것은 허용하지 않는다. 참조변수 u는 실제로 GroundUnit인스턴스를 참조하고 있다. (Tank)u는 GroundUnit인스턴스를 자손타입인 Tank로 형변환하는 것인데, 자손타입으로 형변환은 허용되지 않으므로 실행시 에러가 발생한다.

[참고] 컴파일 시에는 타입만을 체크하기 때문에 에러가 발생하지 않을 수도 있지만, 실행시에 에러가 발생한다.

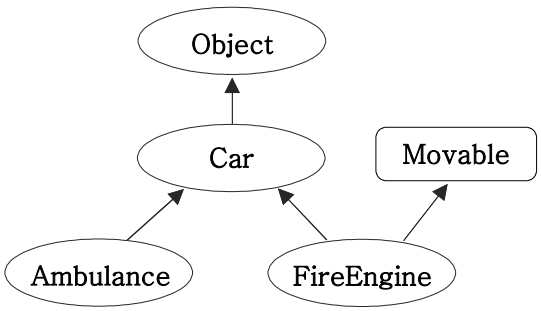
```
Unit u = new GroundUnit();
Tank t = new Tank();

t = (Tank)u; // 조상인스턴스(GroundUnit)를 자손(Tank)으로 형변환할 수 없다.
Tank t = (Tank)new GroundUnit; // 허용되지 않음
```

[7-16] 다음 중 연산결과가 true가 아닌 것은? (모두 고르시오)

```
class Car {}
class FireEngine extends Car implements Movable {}
class Ambulance extends Car {}

FireEngine fe = new FireEngine();
```

a. fe instanceof FireEngine
b. fe instanceof Movable
c. fe instanceof Object
d. fe instanceof Car
e. fe instanceof Ambulance

[정답] e
[해설] instanceof연산자는 실제 인스턴스의 모든 조상이나 구현한 인터페이스에 대해 true를 반환한다. 그래서, 아래 그림에서 알 수 있듯이 FireEngine인스턴스는 Object, Car, Movable, FireEngine타입에 대해 instanceof연산을 하면 결과로 true를 얻는다. 어떤 타입에 대해 instanceof연산결과가 true라는 것은 그 타입으로 형변환이 가능하다는 것을 뜻한다. 참조변수의 형변환을 하기에 앞서 instanceof연산자로 형변환이 가능한지 미리 확인해 보는 것이 좋다.



[7-17] 아래 세 개의 클래스로부터 공통부분을 뽑아서 Unit이라는 클래스를 만들고, 이 클래스를 상속받도록 코드를 변경하시오.

```
class Marine {    // 보병
    int x, y;     // 현재 위치
    void move(int x, int y) { /* 지정된 위치로 이동 */ }
    void stop()             { /* 현재 위치에 정지 */ }
    void stimPack()         { /* 스팀팩을 사용한다.*/}
}

class Tank {      // 탱크
    int x, y;     // 현재 위치
    void move(int x, int y) { /* 지정된 위치로 이동 */ }
    void stop()             { /* 현재 위치에 정지 */ }
    void changeMode()       { /* 공격모드를 변환한다. */}
}

class Dropship {  // 수송선
    int x, y;     // 현재 위치
    void move(int x, int y) { /* 지정된 위치로 이동 */ }
    void stop()             { /* 현재 위치에 정지 */ }
    void load()             { /* 선택된 대상을 태운다.*/ }
    void unload()           { /* 선택된 대상을 내린다.*/ }
}
```

[정답] 각 클래스의 공통부분을 뽑아서 Unit클래스를 생성하면 된다. 클래스마다 이동하는 방법이 다르므로 move메서드는 추상메서드로 정의하였다. 책에도 같은 내용이 있기 때문에 자세한 설명은 생략하겠다.

```
abstract class Unit {
    int x, y;
    abstract void move(int x, int y); // 추상클래스
    void stop() {      /* 현재 위치에 정지 */ }
}

class Marine extends Unit {  // 보병
    void move(int x, int y) { /* 지정된 위치로 이동 */ }
    void stimPack()         { /* 스팀팩을 사용한다.*/}
}

class Tank extends Unit {    // 탱크
    void move(int x, int y) { /* 지정된 위치로 이동 */ }
    void changeMode()       { /* 공격모드를 변환한다. */}
}

class Dropship extends Unit { // 수송선
    void move(int x, int y) { /* 지정된 위치로 이동 */ }
    void load()             { /* 선택된 대상을 태운다.*/ }
    void unload()           { /* 선택된 대상을 내린다.*/ }
}
```

[7-18] 다음과 같은 실행결과를 얻도록 코드를 완성하시오.

[Hint] instanceof연산자를 사용해서 형변환한다.

메서드명 : action
기　　능 : 주어진 객체의 메서드를 호출한다.
　　DanceRobot인 경우, dance()를 호출하고,
　　SingRobot인 경우, sing()을 호출하고,
　　DrawRobot인 경우, draw()를 호출한다.
반환타입 : 없음
매개변수 : Robot r - Robot인스턴스 또는 Robot의 자손 인스턴스

[연습문제]/ch7/Exercise7_18.java

```
class Exercise7_18 {
   public static void action(Robot r) {
        if(r instanceof DanceRobot) {
            DanceRobot dr = (DanceRobot)r;
            dr.dance();
        } else if(r instanceof SingRobot) {
            SingRobot sr = (SingRobot)r;
            sr.sing();
        } else if(r instanceof DrawRobot) {
            DrawRobot dr = (DrawRobot)r;
            dr.draw();
        }
   }

   public static void main(String[] args) {
        Robot[] arr = { new DanceRobot(), new SingRobot(), new DrawRobot()};

        for(int i=0; i< arr.length;i++)
            action(arr[i]);
   } // main
}

class Robot {}

class DanceRobot extends Robot {
   void dance() {
        System.out.println("춤을 춥니다.");
   }
}

class SingRobot extends Robot {
   void sing() {
        System.out.println("노래를 합니다.");
   }
}

class DrawRobot extends Robot {
   void draw() {
        System.out.println("그림을 그립니다.");
   }
}
```

[실행결과]

```
춤을 춥니다.
노래를 합니다.
그림을 그립니다.
```

[해설] action메서드의 매개변수가 Robot타입이므로 Robot클래스의 자손클래스인 DanceRobot, SingRobot, DrawRobot의 인스턴스는 모두 매개변수로 가능하다.

```
Robot[] arr = { new DanceRobot(), new SingRobot(), new DrawRobot()};

for(int i=0; i< arr.length;i++)
    action(arr[i]);
```

action메서드 내에서는 실제로 받아온 인스턴스가 어떤 것인지 알 수 없다. 단지 Robot클래스 또는 그 자손클래스의 인스턴스일 것이라는 것만 알 수 있다. 그래서 instanceof연산자를 이용해야만 실제 인스턴스의 타입을 확인할 수 있다.

```
public static void action(Robot r) {
    if(r instanceof DanceRobot) {
        DanceRobot dr = (DanceRobot)r;
        dr.dance();
    } else if(r instanceof SingRobot) {
        SingRobot sr = (SingRobot)r;
        sr.sing();
    } else if(r instanceof DrawRobot) {
        DrawRobot dr = (DrawRobot)r;
        dr.draw();
    }
}
```

[7-19] 다음은 물건을 구입하는 사람을 정의한 Buyer클래스이다. 이 클래스는 멤버변수로 돈(money)과 장바구니(cart)를 가지고 있다. 제품을 구입하는 기능의 buy메서드와 장바구니에 구입한 물건을 추가하는 add메서드, 구입한 물건의 목록과 사용금액, 그리고 남은 금액을 출력하는 summary메서드를 완성하시오.

1. 메서드명 : buy
   기　　능 : 지정된 물건을 구입한다. 가진 돈(money)에서 물건의 가격을 빼고,
   장바구니(cart)에 담는다.
   만일 가진 돈이 물건의 가격보다 적다면 바로 종료한다.
   반환타입 : 없음
   매개변수 : Product p - 구입할 물건

2. 메서드명 : add
   기　　능 : 지정된 물건을 장바구니에 담는다.
   만일 장바구니에 담을 공간이 없으면, 장바구니의 크기를 2배로 늘린 다음에 담는다.
   반환타입 : 없음
   매개변수 : Product p - 구입할 물건

3. 메서드명 : summary
   기　　능 : 구입한 물건의 목록과 사용금액, 남은 금액을 출력한다.
   반환타입 : 없음
   매개변수 : 없음

[연습문제]/ch7/Exercise7_19.java

```
class Exercise7_19 {
   public static void main(String args[]) {
        Buyer b = new Buyer();
        b.buy(new Tv());
        b.buy(new Computer());
        b.buy(new Tv());
        b.buy(new Audio());
        b.buy(new Computer());
        b.buy(new Computer());
        b.buy(new Computer());

        b.summary();
   }
}

class Buyer {
   int money = 1000;
   Product[] cart = new Product[3];    // 구입한 제품을 저장하기 위한 배열
   int i = 0;                          // Product배열 cart에 사용될 index

   void buy(Product p) {
//      1.1 가진 돈과 물건의 가격을 비교해서 가진 돈이 적으면 메서드를 종료한다.
        if(money < p.price) {
            System.out.println("잔액이 부족하여 "+ p +"을/를 살수 없습니다.");
            return;
        }
```

```
//      1.2 가진 돈이 충분하면, 제품의 가격을 가진 돈에서 빼고
        money -= p.price;
//      1.3 장바구니에 구입한 물건을 담는다.(add메서드 호출)
        add(p);
    }

    void add(Product p) {
//      1.1  i의 값이 장바구니의 크기보다 같거나 크면
        if(i >= cart.length) {
//          1.1.1 기존의 장바구니보다 2배 큰 새로운 배열을 생성한다.
            Product[] tmp = new Product[cart.length*2];
//          1.1.2 기존의 장바구니의 내용을 새로운 배열에 복사한다.
            System.arraycopy(cart,0,tmp,0,cart.length);
//          1.1.3 새로운 장바구니와 기존의 장바구니를 바꾼다.
            cart = tmp;
        }
//      1.2 물건을 장바구니(cart)에 저장한다. 그리고 i의 값을 1 증가시킨다.
        cart[i++]=p;
    } // add(Product p)

    void summary() {
        String itemList = "";
        int sum = 0;

        for(int i=0; i < cart.length;i++) {
            if(cart[i]==null)
                break;
//          1.1 장바구니에 담긴 물건들의 목록을 만들어 출력한다.
            itemList += cart[i] + ",";
//          1.2 장바구니에 담긴 물건들의 가격을 모두 더해서 출력한다.
            sum += cart[i].price;
        }

//      1.3 물건을 사고 남은 금액(money)를 출력한다.
        System.out.println("구입한 물건:"+itemList);
        System.out.println("사용한 금액:"+sum);
        System.out.println("남은 금액:"+money);
    } // summary()
}

class Product {
    int price;              // 제품의 가격

    Product(int price) {
        this.price = price;
    }
}

class Tv extends Product {
    Tv() { super(100); }

    public String toString() { return "Tv"; }
}

class Computer extends Product {
    Computer() { super(200); }
```

```
    public String toString() { return "Computer";}
}

class Audio extends Product {
    Audio() { super(50); }

    public String toString() { return "Audio"; }
}
```

[실행결과]

```
잔액이 부족하여 Computer을/를 살수 없습니다.
구입한 물건:Tv,Computer,Tv,Audio,Computer,Computer,
사용한 금액:850
남은 금액:150
```

[해설] 자신이 스스로 로직을 작성할 수 있으면 가장 좋겠지만, 적어도 주어진 로직대로 코드를 구현할 수 있는 능력은 갖추어야 한다. 그런 능력을 향상시키기 위한 문제이다. 이 문제가 쉽게 느껴지는 사람은 로직(주석)을 안보고 코드를 다시 작성해보기 바란다.

책에 있는 내용을 복습하는 문제이기 때문에 자세한 설명은 생략하겠다. 책을 참고하길 바란다.

[7-20] 다음의 코드를 실행한 결과를 적으시오.

[연습문제]/ch7/Exercise7_20.java

```
class Exercise7_20 {
   public static void main(String[] args) {
        Parent p = new Child();
        Child  c = new Child();

        System.out.println("p.x = " + p.x);
        p.method();

        System.out.println("c.x = " + c.x);
        c.method();
   }
}

class Parent {
   int x = 100;

   void method() {
        System.out.println("Parent Method");
   }
}

class Child extends Parent {
   int x = 200;

   void method() {
        System.out.println("Child Method");
   }
}
```

[정답]

[실행결과]

```
p.x = 100
Child Method
c.x = 200
Child Method
```

[해설] 조상 클래스에 선언된 멤버변수와 같은 이름의 인스턴스변수를 자손 클래스에 중복으로 정의했을 때, 조상타입의 참조변수로 자손 인스턴스를 참조하는 경우와 자손타입의 참조변수로 자손 인스턴스를 참조하는 경우는 서로 다른 결과를 얻는다.

메서드의 경우 조상 클래스의 메서드를 자손의 클래스에서 오버라이딩한 경우에도 참조변수의 타입에 관계없이 항상 실제 인스턴스의 메서드(오버라이딩된 메서드)가 호출되지만, 멤버변수의 경우 참조변수의 타입에 따라 달라진다.

타입은 다르지만, 참조변수 p, c모두 Child인스턴스를 참조하고 있다.

```
Parent p = new Child();
Child  c = new Child();
```

그리고, Parent클래스와 Child클래스는 서로 같은 멤버들을 정의하고 있다.

```
class Parent {
   int x = 100;
   ...
}

class Child extends Parent {
   int x = 200;
   ...
}
```

이 때 조상타입의 참조변수 p로 Child인스턴스의 멤버들을 사용하는 것과 자손타입의 참조변수 c로 Child인스턴스의 멤버들을 사용하는 것의 차이를 알 수 있다.

메서드인 method()의 경우 참조변수의 타입에 관계없이 항상 실제 인스턴스의 타입인 Child클래스에 정의된 메서드가 호출되지만, 인스턴스변수인 x는 참조변수의 타입에 따라서 달라진다.

[7-21] 다음과 같이 attack메서드가 정의되어 있을 때, 이 메서드의 매개변수로 가능한 것 두 가지를 적으시오.

```
interface Movable {
   void move(int x, int y);
}

void attack(Movable f) {
   /* 내용 생략 */
}
```

[정답] null, Movable인터페이스를 구현한 클래스 또는 그 자손의 인스턴스
[해설] 매개변수의 다형성을 잘 이해하고 있는지를 확인하는 문제이다. 매개변수의 타입이 인터페이스라는 것은 어떤 의미일지 이해하지 못하는 경우가 많은데, 이것을 이해하는 것은 매우 중요하다.

언제라도 누가 'Movable인터페이스타입의 매개변수로 가능한 것이 무엇이냐?'고 물었을 때, 주저 없이 얘기할 수 있도록 완전히 외우고 있어야 한다.

[7-22] 아래는 도형을 정의한 Shape클래스이다. 이 클래스를 조상으로 하는 Circle클래스와 Rectangle클래스를 작성하시오. 이 때, 생성자도 각 클래스에 맞게 적절히 추가해야 한다.

(1) 클래스명 : Circle
조상클래스 : Shape
멤버변수 : double r - 반지름

(2) 클래스명 : Rectangle
조상클래스 : Shape
멤버변수 : double width - 폭
double height - 높이
메서드 :
1. 메서드명 : isSquare
기 능 : 정사각형인지 아닌지를 알려준다.
반환타입 : boolean
매개변수 : 없음

[연습문제]/ch7/Exercise7_22.java

```
abstract class Shape {
    Point p;

    Shape() {
        this(new Point(0,0));
    }

    Shape(Point p) {
        this.p = p;
    }


    abstract double calcArea();  // 도형의 면적을 계산해서 반환하는 메서드

    Point getPosition() {
        return p;
    }

    void setPosition(Point p) {
        this.p = p;
    }
}

class Rect extends Shape {
    double width;
    double height;

    Rect(double width, double height) {
        this(new Point(0,0), width, height);
    }

    Rect(Point p, double width, double height) {
        super(p);    // 조상의 멤버는 조상의 생성자가 초기화하도록 한다.
        this.width = width;
```

```
        this.height = height;
    }

    boolean isSquare() {
        // width나 height가 0이 아니고 width와 height가 같으면 true를 반환한다.
        return width*height!=0 && width==height;
    }

    double calcArea() {
        return width * height;
    }
}

class Circle extends Shape {
    double r;    // 반지름

    Circle(double r) {
        this(new Point(0,0),r); // Circle(Point p, double r)를 호출
    }

    Circle(Point p, double r) {
        super(p);    // 조상의 멤버는 조상의 생성자가 초기화하도록 한다.
        this.r = r;
    }

    double calcArea() {
        return Math.PI * r * r;
    }
}

class Point {
    int x;
    int y;

    Point() {
        this(0,0);
    }

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

    public String toString() {
        return "["+x+","+y+"]";
    }
}
```

[7-23] 문제7-22에서 정의한 클래스들의 면적을 구하는 메서드를 작성하고 테스트 하시오.

1. 메서드명 : sumArea
   기　　능 : 주어진 배열에 담긴 도형들의 넓이를 모두 더해서 반환한다.
   반환타입 : double
   매개변수 : Shape[] arr

[연습문제]/ch7/Exercise7_23.java

```
class Exercise7_23
{
   static double sumArea(Shape[] arr) {
        double sum = 0;

        for(int i=0; i < arr.length;i++)
            sum+= arr[i].calcArea();

        return sum;
   }

   public static void main(String[] args)
   {
        Shape[] arr = {new Circle(5.0), new Rectangle(3,4), new Circle(1)};
        System.out.println("면적의 합:"+sumArea(arr));
   }
}
```

[실행결과]

```
면적의 합:93.68140899333463
```

[해설] 반복문으로 넘겨받은 객체배열(arr)의 객체들에 대해 calcArea()를 호출하여 면적을 구하고 누적해서 반환하도록 작성하면 된다.

Shape타입의 배열에는 Shape의 자손 인스턴스가 들어있기 때문에, Shape클래스의 추상메서드 calcArea()를 호출해도 실제로는 각 인스턴스에 완전히 구현된 calcArea()가 호출된다.

[7-24] 다음 중 인터페이스의 장점이 아닌 것은?

a. 표준화를 가능하게 해준다.
b. 서로 관계없는 클래스들에게 관계를 맺어 줄 수 있다.
c. 독립적인 프로그래밍이 가능하다.
d. 다중상속을 가능하게 해준다.
e. 패키지간의 연결을 도와준다.

[정답] e
[해설] 인터페이스를 사용하는 이유와 그 장점을 정리해 보면 다음과 같다.

1. 개발시간을 단축시킬 수 있다.
일단 인터페이스가 작성되면, 이를 사용해서 프로그램을 작성하는 것이 가능하다. 메서드를 호출하는 쪽에서는 메서드의 내용에 관계없이 선언부만 알면 되기 때문이다.
그리고 동시에 다른 한 쪽에서는 인터페이스를 구현하는 클래스를 작성하도록 하여, 인터페이스를 구현하는 클래스가 작성될 때까지 기다리지 않고도 양쪽에서 동시에 개발을 진행할 수 있다.

2. 표준화가 가능하다.
프로젝트에 사용되는 기본 틀을 인터페이스로 작성한 다음, 개발자들에게 인터페이스를 구현하여 프로그램을 작성하도록 함으로써 보다 일관되고 정형화된 프로그램의 개발이 가능하다.

3. 서로 관계없는 클래스들에게 관계를 맺어 줄 수 있다.
서로 상속관계에 있지도 않고, 같은 조상클래스를 가지고 있지 않은 서로 아무런 관계도 없는 클래스들에게 하나의 인터페이스를 공통적으로 구현하도록 함으로써 관계를 맺어 줄 수 있다.

4. 독립적인 프로그래밍이 가능하다.
인터페이스를 이용하면 클래스의 선언과 구현을 분리시킬 수 있기 때문에 실제구현에 독립적인 프로그램을 작성하는 것이 가능하다. 클래스와 클래스간의 직접적인 관계를 인터페이스를 이용해서 간접적인 관계로 변경하면, 한 클래스의 변경이 관련된 다른 클래스에 영향을 미치지 않는 독립적인 프로그래밍이 가능하다.

[7-25] Outer클래스의 내부 클래스 Inner의 멤버변수 iv의 값을 출력하시오.

[연습문제]/ch10/Exercise7_25.java

```
class Outer {       // 외부 클래스
   class Inner {    // 내부 클래스(인스턴스 클래스)
       int iv=100;
   }
}

class Exercise7_25 {
   public static void main(String[] args) {
       Outer o = new Outer();
       Outer.Inner ii = o.new Inner();
       System.out.println(ii.iv);
   }
}
```

[실행결과]

```
100
```

[해설] 내부 클래스(인스턴스 클래스)의 인스턴스를 생성하기 위해서는 먼저 외부클래스의 인스턴스를 생성해야한다. 왜냐하면 '인스턴스 클래스'는 외부 클래스의 '인스턴스 변수'처럼 외부 클래스의 인스턴스가 생성되어야 쓸 수 있기 때문이다.

[7-26] Outer클래스의 내부 클래스 Inner의 멤버변수 iv의 값을 출력하시오.

[연습문제]/ch10/Exercise7_26.java

```
class Outer {              // 외부 클래스
   static class Inner {    // 내부 클래스(static클래스)
       int iv=200;
   }
}

class Exercise7_26 {
   public static void main(String[] args) {
       Outer.Inner ii = new Outer.Inner();
       System.out.println(ii.iv);
   }
}
```

[실행결과]

```
200
```

[해설] 스태틱 클래스(static inner class)는 인스턴스 클래스와 달리 외부 클래스의 인스턴스를 생성하지 않고도 사용할 수 있다. 마치 static멤버를 인스턴스 생성없이 사용할 수 있는 것처럼.

[7-27] 다음과 같은 실행결과를 얻도록 (1)~(4)의 코드를 완성하시오.

[연습문제]/ch10/Exercise7_27.java

```
class Outer {
   int value=10;          // Outer.this.value

   class Inner {  // 인스턴스 클래스(instance inner class)
        int value=20;      // this.value

        void method1() {
            int value=30; // value

            System.out.println(           value);
            System.out.println(      this.value);
            System.out.println(Outer.this.value);
        }
   } // Inner클래스의 끝
} // Outer클래스의 끝

class Exercise7_27 {
   public static void main(String args[]) {
        Outer outer = new Outer();
        Outer.Inner inner = outer.new Inner();

        inner.method1();
   }
}
```

[실행결과]

```
30
20
10
```

[해설] 외부 클래스와 내부 클래스에 같은 이름의 인스턴스 변수(value)가 선언되었을 때 어떻게 구별하는가에 대한 문제이다. 외부 클래스의 인스턴스 변수는 내부 클래스에서 '외부클래스이름.this.변수이름' 로 접근할 수 있다.

내부 클래스의 종류가 인스턴스 클래스이기 때문에 외부 클래스의 인스턴스를 생성한 다음에야 내부 클래스의 인스턴스를 생성할 수 있다.

[7-28] 아래의 EventHandler를 익명 클래스(anonymous class)로 변경하시오.

[연습문제]/ch7/Exercise7_28.java

```
import java.awt.*;
import java.awt.event.*;

class Exercise7_28
{
   public static void main(String[] args)
   {
        Frame f = new Frame();
        f.addWindowListener(new EventHandler());
   }
}

class EventHandler extends WindowAdapter
{
   public void windowClosing(WindowEvent e) {
        e.getWindow().setVisible(false);
        e.getWindow().dispose();
        System.exit(0);
   }
}
```

[정답]

[연습문제]/ch10/Exercise7_28_2.java

```
import java.awt.*;
import java.awt.event.*;

class Exercise7_28_2
{
   public static void main(String[] args)
   {
        Frame f = new Frame();
        f.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                e.getWindow().setVisible(false);
                e.getWindow().dispose();
                System.exit(0);
            }
        });
   } // main
}
```

[7-29] 지역 클래스에서 외부 클래스의 인스턴스멤버와 static멤버에 모두 접근할 수 있지만, 지역변수는 final이 붙은 상수만 접근할 수 있는 이유 무엇인가?

[정답] 메서드가 수행을 마쳐서 지역변수가 소멸된 시점에도, 지역 클래스의 인스턴스가 소멸된 지역변수를 참조하려는 경우가 발생할 수 있기 때문이다.

[해설] 아직 쓰레드를 배우지 않았지만, 쓰레드를 사용해서 상황을 만들어 보았다.

[연습문제]/ch10/Exercise7_29.java

```
import java.awt.*;
import java.awt.event.*;

class Exercise7_29
{
  public static void main(String[] args)
  {
      final int VALUE = 10; // 외부 클래스의 지역변수

      Thread t = new Thread(new Runnable() { // 익명 클래스(내부 클래스)
          public void run() {
              for(int i=0; i < 10;i++) { // 10번 반복한다.
                  try {
                      Thread.sleep(1*1000); // 1초간 멈춘다.
                  } catch(InterruptedException e) {}

                  System.out.println(VALUE); // 외부 클래스의 지역변수를 사용
              }
          } // run()
      });

      t.start(); // 쓰레드를 시작한다.
      System.out.println("main() - 종료.");
  } // main
}
```

[실행결과]

```
main() - 종료.
10
10
10
10
10
10
10
10
10
10
```

실행결과를 보면 main메서드가 종료된 후에도 지역변수 VALUE의 값을 사용하고 있다는 것을 알 수 있다. 지역변수는 메서드가 종료되면 함께 사라지지만, 상수의 경우 이미 컨스턴트 풀(constant pool, 상수를 따로 모아서 저장해 놓는 곳)에 저장되어 있기 때문에 사용할 수 있는 것이다.

[연습문제 - 모범답안]

[8-1] 예외처리의 정의와 목적에 대해서 설명하시오.

[정답]

정의 - 프로그램 실행 시 발생할 수 있는 예외의 발생에 대비한 코드를 작성하는 것
목적 - 프로그램의 비정상 종료를 막고, 정상적인 실행상태를 유지하는 것

[해설] 프로그램의 실행도중에 발생하는 에러는 어쩔 수 없지만, 예외는 프로그래머가 이에 대한 처리를 미리 해주어야 한다.

에러(error) - 프로그램 코드에 의해서 수습될 수 없는 심각한 오류
예외(exception) - 프로그램 코드에 의해서 수습될 수 있는 다소 미약한 오류

예외처리(exception handling)란, 프로그램 실행 시 발생할 수 있는 예기치 못한 예외의 발생에 대비한 코드를 작성하는 것이며, 예외처리의 목적은 예외의 발생으로 인한 실행 중인 프로그램의 갑작스런 비정상 종료를 막고, 정상적인 실행상태를 유지할 수 있도록 하는 것이다.

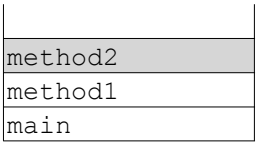
예외처리(exception handling)의
정의 - 프로그램 실행 시 발생할 수 있는 예외의 발생에 대비한 코드를 작성하는 것
목적 - 프로그램의 비정상 종료를 막고, 정상적인 실행상태를 유지하는 것

[8-2] 다음은 실행도중 예외가 발생하여 화면에 출력된 내용이다. 이에 대한 설명 중 옳지 않은 것은?

```
java.lang.ArithmeticException : / by zero
     at ExceptionEx18.method2(ExceptionEx18.java:12)
     at ExceptionEx18.method1(ExceptionEx18.java:8)
     at ExceptionEx18.main(ExceptionEx18.java:4)
```

a. 위의 내용으로 예외가 발생했을 당시 호출스택에 존재했던 메서드를 알 수 있다.
b. 예외가 발생한 위치는 method2 메서드이며, ExceptionEx18.java파일의 12번째 줄이다.
c. 발생한 예외는 ArithmeticException이며, 0으로 나누어서 예외가 발생했다.
d. method2메서드가 method1메서드를 호출하였고 그 위치는 ExceptionEx18.java파일의 8번째 줄이다.

[정답] d
[해설] 예외의 종류는 ArithmeticException이고 0으로 나눠서 발생하였다. 예외가 발생한 곳은 method2이고 ExceptionEx18.java의 12번째 줄이다. 예외가 발생했을 당시의 호출스택을 보면 아래의 그림과 같다. 호출스택은 맨 위에 있는 메서드가 현재 실행 중인 메서드이고 아래 있는 메서드가 바로 위의 메서드를 호출한 것이다. 그래서 main → method1 → method2의 순서로 호출되었음을 알 수 있다.



괄호안의 내용은 예외가 발생한 소스와 라인인데, method1()의 경우 예외가 발생한 곳이 method2()호출한 라인이고 main의 경우 method1()을 호출한 라인이다.
method1()에서 봤을 때는 method2()를 호출한 곳에서 예외가 발생한 것이기 때문이다. main메서드 역시 마찬가지.

[8-3] 다음 중 오버라이딩이 잘못된 것은? (모두 고르시오)

```
void add(int a, int b)
     throws InvalidNumberException, NotANumberException {}

class NumberException extends Exception {}
class InvalidNumberException extends NumberException {}
class NotANumberException extends NumberException {}
```

a. void add(int a, int b) throws InvalidNumberException, NotANumberException {}
b. void add(int a, int b) throws InvalidNumberException {}
c. void add(int a, int b) throws NotANumberException {}
d. void add(int a, int b) throws Exception {}
e. void add(int a, int b) throws NumberException {}

[정답] d, e
[해설] 오버라이딩(overriding)을 할 때, 조상 클래스의 메서드보다 많은 수의 예외를 선언할 수 없다.

- 아래의 코드를 보면 Child클래스의 parentMethod()에 선언된 예외의 개수가 조상인 Parent클래스의 parentMethod()에 선언된 예외의 개수보다 적으므로 바르게 오버라이딩 되었다.

```
Class Parent {
     void parentMethod() throws IOException, SQLException {
          //..
     }
}

Class Child extends Parent {
     void parentMethod() throws IOException {
          //..
     }
     //..
}
```

여기서 주의해야할 점은 단순히 선언된 예외의 개수의 문제가 아니라는 것이다.

```
Class Child extends Parent {
     void parentMethod() throws Exception {
          //..
    }
    //..
}
```

만일 위와 같이 오버라이딩을 하였다면, 분명히 조상클래스에 정의된 메서드보다 적은 개수의 예외를 선언한 것처럼 보이지만 Exception은 모든 예외의 최고 조상이므로 가장 많

은 개수의 예외를 던질 수 있도록 선언한 것이다.
그래서 예외의 개수는 적거나 같아야 한다는 조건을 만족시키지 못하는 잘못된 오버라이딩인 것이다.

아래의 코드로 이 문제를 직접 테스트할 수 있다.

```
class NumberException extends Exception {}
class InvalidNumberException extends NumberException {}
class NotANumberException extends NumberException {}

class Parent {
   int a;
   int b;

   Parent() {
       this(0,0);
   }

   Parent(int a, int b) {
       this.a = a;
       this.b = b;
   }

   void add(int a, int b)
         throws InvalidNumberException, NotANumberException {}

}

class Child extends Parent {
   Child() {}
   Child(int a, int b) {
       super(a,b);
   }

   void add(int a, int b)
         throws InvalidNumberException, NotANumberException {}
}
```

[8-4] 다음과 같은 메서드가 있을 때, 예외를 잘못 처리한 것은? (모두 고르시오)

```
void method() throws InvalidNumberException, NotANumberException {}

class NumberException extends RuntimeException {}
class InvalidNumberException extends NumberException {}
class NotANumberException extends NumberException {}
```

a. try {method();} catch(Exception e) {}
b. try {method();} catch(NumberException e) {} catch(Exception e) {}
c. try {method();} catch(Exception e) {} catch(NumberException e) {}
d. try {method();} catch(InvalidNumberException e) {
} catch(NotANumberException e) {}
e. try {method();} catch(NumberException e) {}
f. try {method();} catch(RuntimeException e) {}

[정답] c
[해설] try블럭 내에서 예외가 발생하면, catch블럭 중에서 예외를 처리할 수 있는 것을 을 차례대로 찾아 내려간다. 발생한 예외의 종류와 일치하는 catch블럭이 있으면 그 블럭의 문장들을 수행하고 try-catch문을 빠져나간다. 일치하는 catch블럭이 없으면 예외는 처리되지 않는다.

발생한 예외의 종류와 일치하는 catch블럭을 찾을 때, instanceof로 검사를 하기 때문에 모든 예외의 최고조상인 Exception이 선언된 catch블럭은 모든 예외를 다 처리할 수 있다. 한 가지 주의할 점은 Exception을 처리하는 catch블럭은 모든 catch블럭 중 제일 마지막에 있어야 한다는 것이다.

```
try {
    method();
} catch(Exception e) {   // 컴파일 에러 발생!!!

} catch(NumberException e) {

}
```

위의 코드에서는 Exception을 선언한 catch블럭이 마지막 catch블럭이 아니기 때문에 컴파일 에러가 발생한다.

[8-5] 아래의 코드가 수행되었을 때의 실행결과를 적으시오.

[연습문제]/ch8/Exercise8_5.java

```
class Exercise8_5 {
   static void method(boolean b) {
        try {
            System.out.println(1);
            if(b) throw new ArithmeticException();
            System.out.println(2); // 예외가 발생하면 실행되지 않는 문장
        } catch(RuntimeException r) {
            System.out.println(3);
            return; // 메서드를 빠져나간다.(finally블럭을 수행한 후에)
        } catch(Exception e) {
            System.out.println(4);
            return;
        } finally {
            System.out.println(5); // 예외발생여부에 관계없이 항상 실행되는 문장
        }

        System.out.println(6);
   }

   public static void main(String[] args) {
        method(true);
        method(false);
   } // main
}
```

[정답]

[실행결과]

```
1
3
5
1
2
5
6
```

[해설] 예외가 발생하면 1,3,5가 출력되고 예외가 발생하지 않으면, 1,2,5,6이 출력된다. ArithmeticException은 RuntimeException의 자손이므로 RuntimeException이 정의된 catch블럭에서 처리된다. 이 catch블럭에 return문이 있으므로 메서드를 종료하고 빠져나가게 되는데, 이 때도 finally블럭이 수행된다.

[8-6] 아래의 코드가 수행되었을 때의 실행결과를 적으시오.

[연습문제]/ch8/Exercise8_6.java

```
class Exercise8_6 {
    public static void main(String[] args) {
        try {
            method1();
        } catch(Exception e) {
            System.out.println(5);
        }
    }

    static void method1() {
        try {
            method2();
            System.out.println(1);
        } catch(ArithmeticException e) {
            System.out.println(2);
        } finally {
            System.out.println(3);
        }

        System.out.println(4);
    }  // method1()

    static void method2() {
        throw new NullPointerException();
    }
}
```

[정답]

[실행결과]

```
3
5
```

[해설] main메서드가 method1()을 호출하고, method1()은 method2()를 호출한다.

method2()에서 NullPointerException이 발생했는데, 이 예외를 처리해줄 try-catch블럭이 없으므로 method2()는 종료되고, 이를 호출한 method1()으로 되돌아갔는데 여기에는 try-catch블럭이 있긴 하지만 NullPointerException을 처리해줄 catch블럭이 없으므로 method1()도 종료되고, 이를 호출한 main메서드로 돌아간다. 이 때 finally블럭이 수행되어 '3'이 출력된다.

main메서드에서는 모든 예외를 처리할 수 있는 Exception이 선언된 catch블럭이 있으므로 예외가 처리되고 '5'가 출력된다.

[8-7] 아래의 코드가 수행되었을 때의 실행결과를 적으시오.

[연습문제]/ch8/Exercise8_7.java

```
class Exercise8_7 {
   static void method(boolean b) {
        try {
            System.out.println(1);
            if(b) System.exit(0);
            System.out.println(2);
        } catch(RuntimeException r) {
            System.out.println(3);
            return;
        } catch(Exception e) {
            System.out.println(4);
            return;
        } finally {
            System.out.println(5);
        }

        System.out.println(6);
   }

   public static void main(String[] args) {
        method(true);
        method(false);
   } // main
}
```

[정답]

[실행결과]

```
1
```

[해설] 변수 b의 값이 true이므로 System.exit(0);이 수행되어 프로그램이 즉시 종료된다. 이럴 때는 finally블럭이 수행되지 않는다.

[8-8] 다음은 1~100사이의 숫자를 맞추는 게임을 실행하던 도중에 숫자가 아닌 영문자를 넣어서 발생한 예외이다. 예외처리를 해서 숫자가 아닌 값을 입력했을 때는 다시 입력을 받도록 보완하라.

```
1과 100사이의 값을 입력하세요 :50
더 작은 수를 입력하세요.
1과 100사이의 값을 입력하세요 :asdf
Exception in thread "main" java.util.InputMismatchException
       at java.util.Scanner.throwFor(Scanner.java:819)
       at java.util.Scanner.next(Scanner.java:1431)
       at java.util.Scanner.nextInt(Scanner.java:2040)
       at java.util.Scanner.nextInt(Scanner.java:2000)
       at Exercise8_8.main(Exercise8_8.java:16)
```

[연습문제]/ch8/Exercise8_8.java

```
import java.util.*;

class Exercise8_8
{
     public static void main(String[] args)
     {
          // 1~100사이의 임의의 값을 얻어서 answer에 저장한다.
          int answer = (int)(Math.random() * 100) + 1;
          int input = 0;  // 사용자입력을 저장할 공간
          int count = 0;  // 시도횟수를 세기 위한 변수

          do {
               count++;
               System.out.print("1과 100사이의 값을 입력하세요 :");

//             input = new Scanner(System.in).nextInt();
               try {
                    input = new Scanner(System.in).nextInt();
               } catch(Exception e) {
                    System.out.println("유효하지 않은 값입니다. "
                                                    +"다시 값을 입력해주세요.");
                    continue;
               }

               if(answer > input) {
                    System.out.println("더 큰 수를 입력하세요.");
               } else if(answer < input) {
                    System.out.println("더 작은 수를 입력하세요.");
               } else {
                    System.out.println("맞췄습니다.");
                    System.out.println("시도횟수는 "+count+"번입니다.");
                    break;  // do-while문을 벗어난다
               }
           } while(true); // 무한반복문
     } // end of main
}  // end of class HighLow
```

[실행결과]

```
1과 100사이의 값을 입력하세요 :50
더 작은 수를 입력하세요.
1과 100사이의 값을 입력하세요 :asdf
유효하지 않은 값입니다. 다시 값을 입력해주세요.
1과 100사이의 값을 입력하세요 :25
더 큰 수를 입력하세요.
1과 100사이의 값을 입력하세요 :38
더 큰 수를 입력하세요.
1과 100사이의 값을 입력하세요 :44
맞췄습니다.
시도횟수는 5번입니다.
```

[해설] 사용자로부터 값을 입력받는 경우에는 유효성검사를 철저하게 해야 한다. 사용자가 어떤 값을 입력할지 모르기 때문이다.

여기서는 간단하게 화면으로부터 값을 입력받는 부분에 try-catch구문으로 예외처리를 해주기만 하면 된다. 값을 입력받을 때 예외가 발생하면, 값을 다시 입력하라는 메세지를 보여주고 다시 입력 받으면 된다.

```
input = new Scanner(System.in).nextInt();
```

```
try {
   input = new Scanner(System.in).nextInt();
} catch(Exception e) {
   System.out.println("유효하지 않은 값입니다. 다시 값을 입력해주세요.");
   continue;
}
```

[8-9] 다음과 같은 조건의 예외클래스를 작성하고 테스트하시오.

[참고] 생성자는 실행결과를 보고 알맞게 작성해야한다.

* 클래스명 : UnsupportedFuctionException
* 조상클래스명 : RuntimeException
* 멤버변수 :
  - 이 름 : ERR_CODE
  - 저장값 : 에러코드
  - 타 입 : int
  - 기본값 : 100
  - 제어자 : final private
* 메서드 :
  1. 메서드명 : getErrorCode
     - 기 능 : 에러코드(ERR_CODE)를 반환한다.
     - 반환타입 : int
     - 매개변수 : 없음
     - 제어자 : public
  2. 메서드명 : getMessage
     - 기 능 : 메세지의 내용을 반환한다.(Exception클래스의 getMessage()를 오버라이딩)
     - 반환타입 : String
     - 매개변수 : 없음
     - 제어자 : public

[연습문제]/ch8/Exercise8_9.java

```
class UnsupportedFuctionException extends RuntimeException {
    private final int ERR_CODE;

    UnsupportedFuctionException(String msg, int errCode) {  // 생성자
        super(msg);
        ERR_CODE = errCode;
    }

    UnsupportedFuctionException(String msg) { // 생성자
        this(msg, 100);         // ERR_CODE를 100(기본값)으로 초기화한다.
    }

    public int getErrCode() {  // 에러 코드를 얻을 수 있는 메서드도 추가했다.
        return ERR_CODE;      // 이 메서드는 주로 getMessage()와 함께 사용될 것이다.
    }

    public String getMessage() { // Exception의 getMeesage()를 오버라이딩한다.
        return "["+getErrCode()+"]" + super.getMessage();
    }
}

class Exercise8_9 {
    public static void main(String[] args) throws Exception
    {
        throw new UnsupportedFuctionException("지원하지 않는 기능입니다.",100);
    }
}
```

[실행결과]

```
Exception in thread "main" UnsupportedFuctionException: [100]지원하지 않는 기능입니다.
        at Exercise8_9.main(Exercise8_9.java:5)
```

[해설] 에러메시지를 저장하는 인스턴스변수 msg는 상속받은 것이므로 조상의 생성자를 호출해서 초기화되도록 해야 한다. ERR_CODE는 한 번 값이 지정되면 바뀌는 값이 아니라서 final을 붙여서 상수로 했다. 그리고 생성자를 통해 초기화하였다.

```
UnsupportedFuctionException(String msg, int errCode) {  // 생성자
        super(msg);      // 조상의 생성자 RuntimeException(String msg)를 호출
        ERR_CODE = errCode;
}
```

getMessage() 역시 조상으로부터 상속받은 것이며, ERR_CODE도 같이 출력되도록 하기 위해 오버라이딩했다. 조상의 메서드를 오버라이딩할 때는, 가능하다면 조상의 메서드를 재활용하는 것이 좋다.

```
public String getMessage() { // Exception의 getMeesage()를 오버라이딩한다.
   return "["+getErrCode()+"]" + super.getMessage();
}
```

[8-10] 아래의 코드가 수행되었을 때의 실행결과를 적으시오.

[연습문제]/ch8/Exercise8_10.java

```
class Exercise8_10 {
    public static void main(String[] args) {
        try {
            method1();  // 예외 발생!!!
            System.out.println(6); // 예외가 발생해서 실행되지 않는다.
        } catch(Exception e) {
            System.out.println(7);
        }
    }

    static void method1() throws Exception {
        try {
            method2();
            System.out.println(1);
        } catch(NullPointerException e) {
            System.out.println(2);
            throw e;     // 예외를 다시 발생시킨다. 예외 되던지기(re-throwing)
        } catch(Exception e) {
            System.out.println(3);
        } finally {
            System.out.println(4);
        }

        System.out.println(5);
    }  // method1()

    static void method2() {
        throw new NullPointerException(); // NullPointerException을 발생시킨다.
    }
}
```

[정답]

[실행결과]

```
2
4
7
```

[해설] method2()에서 발생한 예외를 method1()의 try-catch문에서 처리했다가 다시 발생시킨다.

```
        } catch(NullPointerException e) {
            System.out.println(2);
            throw e;     // 예외를 다시 발생시킨다. 예외 되던지기(re-throwing)
        } catch(Exception e) {
```

예외가 발생한 catch블럭 내에 이 예외(NullPointerException)를 처리할 try-catch블럭이 없기 때문에 method1()이 종료되면서 main메서드에 예외가 전달된다. 이 때 예외가 처리되진 않았지만, finally블럭의 문장이 수행되어 4가 출력된다.

main메서드의 try-catch블럭은 method1()으로부터 전달된 예외를 처리할 catch블럭이 있으므로 해당 catch블럭이 수행되어 7을 출력하고 try-catch블럭을 벗어난다. 그리고 더 이상 수행할 코드가 없으므로 프로그램이 종료된다.

```
try {
    method1();  // NullPointerException 발생!!!
    System.out.println(6); // 예외가 발생해서 실행되지 않는다.
} catch(Exception e) {      // 모든 종류의 예외를 처리할 수 있다.
    System.out.println(7);
}
```

Chapter 9

java.lang패키지

java.lang package

[연습문제 - 모범답안]

[9-1] 다음과 같은 실행결과를 얻도록 SutdaCard클래스의 equals()를 멤버변수인 num, isKwang의 값을 비교하도록 오버라이딩하고 테스트 하시오.

[연습문제]/ch9/Exercise9_1.java

```
class Exercise9_1 {
   public static void main(String[] args) {
        SutdaCard c1 = new SutdaCard(3,true);
        SutdaCard c2 = new SutdaCard(3,true);

        System.out.println("c1="+c1);
        System.out.println("c2="+c2);
        System.out.println("c1.equals(c2):"+c1.equals(c2));
   }
}

class SutdaCard {
   int num;
   boolean isKwang;

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

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

   public boolean equals(Object obj) {
        if(obj instanceof SutdaCard) {
            SutdaCard c = (SutdaCard)obj;
            return num==c.num && isKwang==c.isKwang;
        }

        return false;
   }

   public String toString() {
        return num + ( isKwang ? "K":"");
   }
}
```

[실행결과]

```
c1=3K
c2=3K
c1.equals(c2):true
```

[해설] 매개변수가 Object타입이므로 어떤 타입의 인스턴스도 매개변수로 가능하다.
그래서 반드시 instanceof로 확인한 후에 형변환해서 멤버변수 num과 isKwang의 값을 비교해야한다. 만일 instanceof의 결과가 false라면 멤버변수의 값을 비교할 필요도 없이 그냥 false만 반환하면 된다.

```
public boolean equals(Object obj) {
    if(obj instanceof SutdaCard) {
        SutdaCard c = (SutdaCard)obj;
        return num==c.num && isKwang==c.isKwang;
    }

    return false;
}
```

[9-2] 다음과 같은 실행결과를 얻도록 Point3D클래스의 equals()를 멤버변수인 x, y, z의 값을 비교하도록 오버라이딩하고, toString()은 실행결과를 참고해서 적절히 오버라이딩하시오.

[연습문제]/ch9/Exercise9_2.java

```
class Exercise9_2 {
   public static void main(String[] args) {
        Point3D p1 = new Point3D(1,2,3);
        Point3D p2 = new Point3D(1,2,3);

        System.out.println(p1);
        System.out.println(p2);
        System.out.println("p1==p2?"+(p1==p2));
        System.out.println("p1.equals(p2)?"+(p1.equals(p2)));
   }
}

class Point3D {
   int x,y,z;

   Point3D(int x, int y, int z) {
        this.x=x;
        this.y=y;
        this.z=z;
   }

   Point3D() {
        this(0,0,0);
   }

   public boolean equals(Object obj) {
        if(obj instanceof Point3D) {
            Point3D p =(Point3D)obj;
            return x==p.x && y==p.y && z==p.z;
        }

        return false;
   }

   public String toString() {
        return "["+x+","+y+","+z+"]";
   }
}
```

[실행결과]

```
[1,2,3]
[1,2,3]
p1==p2?false
p1.equals(p2)?true
```

[해설] 문제9-1과 유사한 문제이므로 설명을 생략하겠다.

[9-3] 다음과 같은 실행결과가 나오도록 코드를 완성하시오.

[연습문제]/ch9/Exercise9_3.java

```
class Exercise9_3 {
   public static void main(String[] args) {
        String fullPath = "c:\\jdk1.8\\work\\PathSeparateTest.java";
        String path = "";
        String fileName = "";

        int pos = fullPath.lastIndexOf("\\");

        if(pos!=-1) {
            path = fullPath.substring(0, pos);
            fileName = fullPath.substring(pos+1);
        }

        System.out.println("fullPath:"+fullPath);
        System.out.println("path:"+path);
        System.out.println("fileName:"+fileName);
   }
}
```

[실행결과]

```
fullPath:c:\jdk1.8\work\PathSeparateTest.java
path:c:\jdk1.8\work
fileName:PathSeparateTest.java
```

[해설] lastIndexOf()와 substring()을 사용해서 문자열을 나누는 문제다. 마지막 경로구분자를 찾아야하기 때문에, indexOf()보다는 lastIndexOf()가 적합하다.

lastIndexOf()는 찾는 문자열이 없으면 -1을 반환하기 때문에 조건문을 사용해서 결과가 -1인 경우에는 substring()을 호출하지 않아야 한다.(pos의 값이 음수이면, substring()에서 예외가 발생한다.)

그래서 if문으로 처리를 해주었는데, try-catch로 처리해도 좋다.(if문 없이 문제를 풀었어도 충분히 좋은 답안이다.)

[참고] 아래의 두 문장은 서로 같은 의미이다.

```
fileName = fullPath.substring(pos+1); // 지정된 위치부터 끝까지 잘라낸다.
fileName = fullPath.substring(pos+1, fullPath.length()); // 위 문장과 동일
```

[9-4] 다음과 같이 정의된 메서드를 작성하고 테스트하시오.

메서드명 : printGraph
기　　능 : 주어진 배열에 담긴 값만큼 주어진 문자를 가로로 출력한 후, 값을 출력한다.
반환타입 : 없음
매개변수 : int[] dataArr - 출력할 그래프의 데이터
　　char ch - 그래프로 출력할 문자.

[연습문제]/ch9/Exercise9_4.java

```
class Exercise9_4 {
   static void printGraph(int[] dataArr, char ch) {
        for(int i=0; i < dataArr.length;i++) {
            for(int j=0; j < dataArr[i];j++) {
                System.out.print(ch);
            }
            System.out.println(dataArr[i]);
        }
   }

   public static void main(String[] args) {
        printGraph(new int[]{3,7,1,4},'*');
   }
}
```

[실행결과]

```
***3
*******7
*1
****4
```

[해설] 반복문으로 배열의 저장된 숫자를 읽어서, 그 숫자만큼 별을 출력하면 된다.

[9-5] 다음과 같이 정의된 메서드를 작성하고 테스트하시오.

메서드명 : count
기　　능 : 주어진 문자열(src)에 찾으려는 문자열(target)이 몇 번 나오는지 세어서 반환한다.
반환타입 : int
매개변수 : String src
　　String target

[Hint] String클래스의 indexOf(String str, int fromIndex)를 사용할 것

[연습문제]/ch9/Exercise9_5.java

```
class Exercise9_5 {
   public static int count(String src, String target) {
        int count = 0; // 찾은 횟수
        int pos = 0;   // 찾기 시작할 위치

//      (1) 반복문을 사용해서 아래의 과정을 반복한다.
        while(true) {
//          1. src에서 target을 pos의 위치부터 찾는다.
            pos = src.indexOf(target,pos);

//          2. 찾으면 count의 값을 1 증가 시키고,
//            pos의 값을 target.length만큼 증가시킨다.
            if(pos!=-1) {
                count++;
                pos += target.length();  // pos를 찾은 단어 이후로 옮긴다.
            } else {
//          3. indexOf의 결과가 -1이면 반복문을 빠져나가서 count를 반환한다.
                break;
            }
        }

        return count;
   }

   public static void main(String[] args) {
        System.out.println(count("12345AB12AB345AB","AB"));
        System.out.println(count("12345","AB"));
   }
}
```

[실행결과]

```
3
0
```

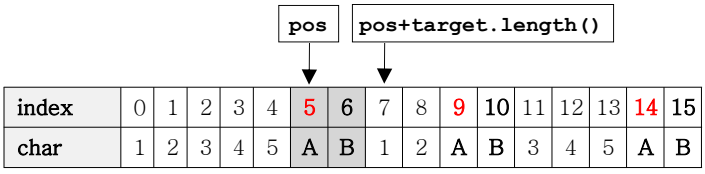
[해설] indexOf()는 지정된 문자열을 찾아서 그 위치를 알려준다. 만일 찾지 못한다면 -1을 반환한다. 문자열 "12345AB12AB345AB"에서 문자열 "AB"를 indexOf()로 찾으면 그 첫 번째 위치는 아래의 표에서 알 수 있듯이 5이다. 두 번째는 9, 세 번째는 14이다.

index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
char	1	2	3	4	5	A	B	1	2	A	B	3	4	5	A	B

지정된 문자열을 찾으면 변수 count를 1증가시키고, 찾기 시작할 위치를 찾은 문자열의 이후로 변경해 주어야 그 다음에 나오는 일치하는 문자열을 찾을 수 있다.

그래서 찾은 위치(pos)에 찾은 문자열의 길이(target.length())를 더해주는 것이다. 여기서는 target.length()의 값이 2이므로 찾은 위치에서 두 글자 뒤부터 찾기 시작한다.

```
// 2. 찾으면 count의 값을 1 증가 시키고, pos의 값을 target.length만큼 증가시킨다.
            if(pos!=-1) {
                count++;
                pos += target.length();  // pos를 찾은 단어 이후로 옮긴다.
            } else {
                ...
```



참고로 반복문을 좀 더 간결히 하면 아래와 같이 할 수 있다.

```
        while((pos = src.indexOf(target,pos))!=-1) {
            count++;
            pos += target.length();
        }
```

[9-6] 다음과 같이 정의된 메서드를 작성하고 테스트하시오.

메서드명 : fillZero
기　　능 : 주어진 문자열(숫자)로 주어진 길이의 문자열로 만들고, 왼쪽 빈 공간은 '0'으로 채운다.
　　만일 주어진 문자열이 null이거나 문자열의 길이가 length의 값과 같으면 그대로 반환한다.
　　만일 주어진 length의 값이 0보다 같거나 작은 값이면, 빈 문자열("")을 반환한다.
반환타입 : String
매개변수 : String src - 변환할 문자열
　　int length - 변환한 문자열의 길이

[연습문제]/ch9/Exercise9_6.java

```
class Exercise9_6 {
   public static String fillZero(String src, int length) {
//         1. src가 널이거나 src.length()가 length와 같으면 src를 그대로 반환한다.
       if(src==null || src.length()==length) {
           return src;
//         2. length의 값이 0보다 같거나 작으면 빈 문자열("")을 반환한다.
       } else if(length <=0) {
           return "";
//         3. src의 길이가 length의 값보다 크면 src를 length만큼 잘라서 반환한다.
       } else if(src.length() > length) {
           return src.substring(0,length);
       }

//         4. 길이가 length인 char배열을 생성한다.
       char[] chArr = new char[length];

//         5. 4에서 생성한 char배열을 '0'으로 채운다.
       for(int i=0;i<chArr.length;i++)
           chArr[i] = '0';

//         6. src에서 문자배열을 뽑아내서 4에서 생성한 배열에 복사한다.
       System.arraycopy(src.toCharArray(),0,chArr,length-src.length(),
src.length());

//         7. 4에서 생성한 배열로 String을 생성해서 반환한다.
       return new String(chArr);
   }

   public static void main(String[] args) {
       String src = "12345";
       System.out.println(fillZero(src,10));
       System.out.println(fillZero(src,-1));
       System.out.println(fillZero(src,3));
   }
}
```

[실행결과]

```
0000012345

123
```

[9-7] 다음과 같이 정의된 메서드를 작성하고 테스트하시오.

메서드명 : contains
기　　능 : 첫 번째 문자열(src)에 두 번째 문자열(target)이 포함되어 있는지 확인한다.
　　포함되어 있으면 true, 그렇지 않으면 false를 반환한다.
반환타입 : boolean
매개변수 : String src
　　String target

[Hint] String클래스의 indexOf()를 사용할 것

[연습문제]/ch9/Exercise9_7.java

```
class Exercise9_7 {
    public static boolean contains(String src, String target) {
        return src.indexOf(target)!=-1;
    }

    public static void main(String[] args) {
        System.out.println(contains("12345","23"));
        System.out.println(contains("12345","67"));
    }
}
```

[실행결과]

```
true
false
```

[해설] indexOf()는 지정된 문자열(src)에서 특정 문자열(target)을 찾아서 그 위치를 알려준다. 만일 찾지 못한다면 -1을 반환하므로 indexOf()의 결과가 -1인지만 확인해서 그 결과를 돌려주면 된다.

예를 들어 'src.indexOf(target)'의 결과가 -1이라면 아래와 같은 연산과정을 거친다.

```
    return src.indexOf(target)!=-1;
→   return -1!=-1;
→   return false;
```

[9-8] 다음과 같이 정의된 메서드를 작성하고 테스트하시오.

메서드명 : round
기　　능 : 주어진 값을 반올림하여, 소수점 이하 n자리의 값을 반환한다.
　　예를 들어 n의 값이 3이면, 소수점 4째 자리에서 반올림하여 소수점 이하 3자리의 수를 반환한다.
반환타입 : double
매개변수 : double d - 변환할 값
　　int n - 반올림한 결과의 소수점 자리

[Hint] Math.round()와 Math.pow()를 이용하라.

[연습문제]/ch9/Exercise9_8.java

```
class Exercise9_8 {
   public static double round(double d, int n) {
       return Math.round(d * Math.pow(10, n)) / Math.pow(10,n);
   }

   public static void main(String[] args) {
       System.out.println(round(3.1415,1));
       System.out.println(round(3.1415,2));
       System.out.println(round(3.1415,3));
       System.out.println(round(3.1415,4));
       System.out.println(round(3.1415,5));
   }
}
```

[실행결과]

```
3.1
3.14
3.142
3.1415
3.1415
```

[해설] Math.round()는 소수점 첫 째 자리에서 반올림해서 long타입의 정수로 반환하고 Math.pow(double a, double b)는 a의 b제곱을 반환한다. 10의 3제곱은 Math.pow(10,3)으로 구할 수 있다.

Math.round()가 소수점 첫 째 자리에서 반올림하기 때문에, 소수점 n+1째자리에서 반올림해서 소수점 n째자리로 만들려면 10의 n제곱을 곱한 다음에 반올림하고 다시 10의 n제곱으로 나눠줘야 한다.

이 두 메서드는 Math클래스에 선언되어 있으며, 선언부는 아래와 같다.

```
public static long round(double a)
public static double pow(double a, double b)
```

[9-9] 다음과 같이 정의된 메서드를 작성하고 테스트하시오.

메서드명 : delChar
기　　능 : 주어진 문자열에서 금지된 문자들을 제거하여 반환한다.
반환타입 : String
매개변수 : String src - 변환할 문자열
　　String delCh - 제거할 문자들로 구성된 문자열

[힌트] StringBuffer와 String클래스의 charAt(int i)과 indexOf(int ch)를 사용하라.

[연습문제]/ch9/Exercise9_9.java

```
class Exercise9_9 {
  public static String delChar(String src, String delCh) {
      StringBuffer sb = new StringBuffer(src.length());

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

          // ch가 delCh에 포함되있지 않으면(indexOf()로 못찾으면) sb에 추가
          if(delCh.indexOf(ch)==-1) // indexOf(int ch)를 호출
              sb.append(ch);
      }

      return sb.toString(); // StringBuffer에 저장된 내용을 String으로 반환
  }

  public static void main(String[] args)     {
      System.out.println("(1!2@3^4~5)"+" -> "
                                  + delChar("(1!2@3^4~5)","~!@#$%^&*()"));
      System.out.println("(1 2    3   4\t5)"+" -> "
                                  + delChar("(1 2    3   4\t5)"," \t"));
  }
}
```

[실행결과]

```
(1!2@3^4~5) -> 12345
(1 2    3   4   5) -> (12345)
```

[해설] 반복문을 이용해서 주어진 문자열(src)의 문자를 순서대로 가져와서, 삭제할 문자열(delCh)에 포함되었는지 확인한다. 포함되어 있지 않을 때만(indexOf()의 결과가 -1일 때만), StringBuffer에 추가한다.

```
int indexOf(int ch)      //  문자열에서 특정 문자(ch)를 찾을 때 사용
                         //  매개변수 타입이 int지만 char값을 넣으면 된다.

int indexOf(String str) // 문자열에서 특정 문자열(str)을 찾을 때 사용
```

[9-10] 다음과 같이 정의된 메서드를 작성하고 테스트하시오.

메서드명 : format
기　　능 : 주어진 문자열을 지정된 크기의 문자열로 변환한다. 나머지 공간은 공백으로 채운다.
반환타입 : String
매개변수 : String str - 변환할 문자열
int length - 변환된 문자열의 길이
int alignment - 변환된 문자열의 정렬조건
(0:왼쪽 정렬, 1: 가운데 정렬, 2:오른쪽 정렬)

[연습문제]/ch9/Exercise9_10.java

```
class Exercise9_10 {
   static String format(String str, int length, int alignment) {

//     1. length의 값이 str의 길이보다 작으면 length만큼만 잘라서 반환한다.
       int diff = length - str.length();
       if(diff < 0) return str.substring(0, length);

//     2. 1의 경우가 아니면, length크기의 char배열을 생성하고 공백으로 채운다.
       char[] source = str.toCharArray(); // 문자열을 char배열로 변환
       char[] result = new char[length];

       for(int i=0; i < result.length; i++)
           result[i] = ' ';      // 배열 result를 공백으로 채운다.

//     3. 정렬조건(alignment)의 값에 따라 문자열(str)을 복사할 위치를 결정한다.
       switch(alignment) {
           case 0 :
           default :
               System.arraycopy(source, 0, result, 0, source.length);
               break;
           case 1 :
               System.arraycopy(source, 0, result, diff/2, source.length);
               break;
           case 2 :
               System.arraycopy(source, 0, result, diff, source.length);
               break;
       }

//     4. 2에서 생성한 char배열을 문자열로 만들어서 반환한다.
       return new String(result);
   } // static String format(String str, int length, int alignment) {

   public static void main(String[] args) {
       String str = "가나다";

       System.out.println(format(str,7,0)); // 왼쪽 정렬
       System.out.println(format(str,7,1)); // 가운데 정렬
       System.out.println(format(str,7,2)); // 오른쪽 정렬
   }
}
```

[실행결과]

```
가나다
  가나다
    가나다
```

[9-11] 커맨드라인으로 2~9사이의 두 개의 숫자를 받아서 두 숫자사이의 구구단을 출력하는 프로그램을 작성하시오.

예를 들어 3과 5를 입력하면 3단부터 5단까지 출력한다.

[실행결과]

```
C:\jdk1.8\work\ch9>java Exercise9_11 2
시작 단과 끝 단, 두 개의 정수를 입력해주세요.
USAGE : GugudanTest 3 5

C:\jdk1.8\work\ch9>java Exercise9_11 1 5
단의 범위는 2와 9사이의 값이어야 합니다.
USAGE : GugudanTest 3 5

C:\jdk1.8\work\ch9>java Exercise9_11 3 5
3*1=3
3*2=6
3*3=9
3*4=12
3*5=15
3*6=18
3*7=21
3*8=24
3*9=27

4*1=4
4*2=8
4*3=12
4*4=16
4*5=20
4*6=24
4*7=28
4*8=32
4*9=36

5*1=5
5*2=10
5*3=15
5*4=20
5*5=25
5*6=30
5*7=35
5*8=40
5*9=45
```

[정답]

[연습문제]/ch9/Exercise9_11.java

```
class Exercise9_11 {
  public static void main(String[] args) {
      int from = 0;
      int to   = 0;

      try {
          if(args.length!=2)
```

```
                throw new Exception("시작 단과 끝 단, 두 개의 정수를 입력해주세요.");

            from = Integer.parseInt(args[0]);
            to   = Integer.parseInt(args[1]);

            if(!(2 <= from && from <= 9 && 2 <= to && to <= 9))
                throw new Exception("단의 범위는 2와 9사이의 값이어야 합니다.");
        } catch(Exception e) {
            System.out.println(e.getMessage());
            System.out.println("USAGE : GugudanTest 3 5");
            System.exit(0);
        }

        // 시작 단(from)이 끝 단(to)보다 작아야하니까
        // to보다 from의 값이 크면 두 값을 바꾼다.
        if(from > to) {
            int tmp = from;
            from = to;
            to = tmp;
        }

        // from단부터 to단까지 출력한다.
        for(int i=from;i<=to;i++) {
            for(int j=1;j<=9;j++) {
                System.out.println(i+"*"+j+"="+i*j);
            }
            System.out.println();
        }
    } // main
}
```

[해설] 커맨드라인을 통해 두 개의 정수를 입력받고 유효성 검사를 하는 부분을 예외로 처리하였는데, 그냥 if문으로만 처리해도 되지만 코드의 중복을 줄이려고 그렇게 했다. 특별히 어려운 부분은 없으므로 자세한 설명은 생략하겠다.

[9-12] 다음과 같이 정의된 메서드를 작성하고 테스트하시오.

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

메서드명 : getRand
기　　능 : 주어진 범위(from~to)에 속한 임의의 정수값을 반환한다.
(양쪽 경계값 모두 범위에 포함)
from의 값이 to의 값보다 클 경우도 처리되어야 한다.
반환타입 : int
매개변수 : int from - 범위의 시작값
int to - 범위의 끝값

[Hint] Math.random()과 절대값을 반환하는 Math.abs(int a), 그리고 둘 중에 작은 값을 반환하는 Math.min(int a, int b)를 사용하라.

[연습문제]/ch9/Exercise9_12.java

```
class Exercise9_12
{
   public static int getRand(int from, int to) {
      return (int)(Math.random() * (Math.abs(to-from)+1))+ Math.min(from,to);
   }

   public static void main(String[] args)
   {
       for(int i=0; i< 20; i++)
           System.out.print(getRand(1,-3)+",");
   }
}
```

[실행결과]

```
0,-1,1,0,-2,-2,1,1,-3,0,-1,1,1,1,0,-1,1,0,-1,-3,
```

[해설] 만일 1~10범위의 임의의 값을 구하려면, 아래와 같은 식으로 구할 수 있다는 것을 이미 배웠다.

```
1 <= (int)(Math.random()*10)+1 < 11
```

위의 식에서 Math.random()에 곱해준 값 10은 바로 범위에 포함된 정수의 개수라는 것을 알 수 있다. 예를 들어 1~10의 범위라면 10개의 정수 중의 하나가 임의로 선택될 것이다. 이 값은 '(끝값-시작값)+1'로 구할 수 있다. '(10-1)+1'을 계산한 결과는 10이 된다.

```
1 <= (int)(Math.random()*(to-from+1)+1 < 11
```

문제의 조건 중에 from의 값이 to의 값보다 클 경우도 처리하라는 것이 있으므로 Math.abs()를 이용해서 '끝값-시작값'이 음수가 되지 않도록 처리해야 한다.

```
1 <= (int)(Math.random()*(Math.abs(to-from)+1)+1 < 11
```

Math.random()에 더해주는 값은 범위의 시작값이다. 문제의 조건 중에 from의 값이 to값보다 클 경우도 처리해야하므로 from과 to중에서 작은 값이 더해져야 한다. 삼항 연산자를 써서 처리할 수도 있지만, Math.min()을 사용했다.

```
          1 <= (int)(Math.random()*(Math.abs(to-from)+1)+1 < 11
1 <= (int)(Math.random()*(Math.abs(to-from)+1) +(from < to ? from:to) < 11
 1 <= (int)(Math.random()*(Math.abs(to-from)+1) + Math.min(from,to) < 11
```

[9-13] 다음은 하나의 긴 문자열(source) 중에서 특정 문자열과 일치하는 문자열의 개수를 구하는 예제이다. 빈 곳을 채워 예제를 완성하시오.

[연습문제]/ch9/Exercise9_13.java

```
public class Exercise9_13 {
   public static void main(String[] args)     {
        String src = "aabbccAABBCCaa";
        System.out.println(src);
        System.out.println("aa를 " + stringCount(src, "aa") +"개 찾았습니다.");
   }

   static int stringCount(String src, String key) {
        return stringCount(src, key, 0);
   }

   static int stringCount(String src, String key, int pos) {
        int count = 0;
        int index = 0;

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

        while((index = src.indexOf(key, pos))!=-1) {
            count++;
            pos = index + key.length();
        }

        return count;
   }
}
```

[실행결과]

```
aabbccAABBCCaa
aa를 2개 찾았습니다.
```

[해설] String클래스의 indexOf()는 특정 문자열(src)에서 찾으려는 문자열(key)이 존재하면 해당 문자열이 시작하는 위치를 반환하고, 존재하지 않으면 -1을 반환한다. pos는 비교를 시작할 위치이며, pos의 값이 0이면 문자열의 처음부터 검색한다.

반복문을 이용해서 indexOf()의 결과가 -1아 될 때까지 반복하면 된다.

```
while((index = src.indexOf(key, pos))!=-1) {
    count++;   // 일치하는 부분을 찾으면, count의 값을 1증가
    pos = index + key.length();  // 검색을 시작할 위치를 변경
}
```

while문의 조건식은 아래의 문장과 조건식을 하나로 합친 것과 같다.

```
index = src.indexOf(key, pos);   // src에 key와 일치하는 부분의 위치를 반환
index !=-1                       // 조건식. index의 값이 -1인지 비교
```

여기서 한가지 주의할 점은, 검색하려는 문자열(key)와 일치하는 위치를 찾으면 그 다음

에는 pos의 위치를 변경해줘야 한다. 그래야 그 다음으로 일치하는 위치를 찾을 수 있다. 검색을 시작할 위치(pos)를 변경하지 않으면, 계속해서 같은 위치를 반환하기 때문에 무한반복에 빠지게 된다.

```
while((index = src.indexOf(key, pos))!=-1) {
    count++;
    pos = index + key.length();  // 검색을 시작할 위치(pos)를
                 // 발견위치(index)+찾을 문자열 길이(key.length())로 변경
}
```

[9-14] 다음은 화면으로부터 전화번호의 일부를 입력받아 일치하는 전화번호를 주어진 문자열 배열에서 찾아서 출력하는 프로그램이다. 알맞은 코드를 넣어 프로그램을 완성하시오.

[Hint] Pattern, Matcher클래스를 사용하라.

[연습문제]/ch11/Exercise11_18.java

```
import java.util.*;
import java.util.regex.*;

class Exercise11_18 {
   public static void main(String[] args) {
        String[] phoneNumArr = {
            "012-3456-7890",
            "099-2456-7980",
            "088-2346-9870",
            "013-3456-7890"
        };

        Vector list = new Vector();  // 검색결과를 담을 Vector
        Scanner s = new Scanner(System.in);

        while(true) {
            System.out.print(">>");
            String input = s.nextLine().trim(); // trim()으로 입력내용에서 공백을 제거

            if(input.equals("")) {
                continue;
            } else if(input.equalsIgnoreCase("Q")) {
                System.exit(0);
            }

            String pattern = ".*"+input+".*";  // input을 포함하는 모든 문자열
            Pattern p = Pattern.compile(pattern);

            for(int i=0; i< phoneNumArr.length;i++) {
                String phoneNum = phoneNumArr[i];
                String tmp = phoneNum.replace("-",""); // phoneNum에서 '-'를 제거

                Matcher m = p.matcher(tmp);

                if(m.find()) {     // 패턴과 일치하면, list에 phoneNum을 추가한다.
                    list.add(phoneNum);
                }
            }

            if(list.size()>0) {             // 검색결과가 있으면
                System.out.println(list);  // 검색결과를 출력하고
                list.clear();              // 검색결과를 삭제
            } else {
                System.out.println("일치하는 번호가 없습니다.");
            }
        }
   } // main
}
```

[실행결과]

```
>>
>>
>>asdf
일치하는 번호가 없습니다.
>>
>>
>>0
[012-3456-7890, 099-2456-7980, 088-2346-9870, 013-3456-7890]
>>234
[012-3456-7890, 088-2346-9870]
>>7890
[012-3456-7890, 013-3456-7890]
>>q
```

[해설] 입력받은 문자열을 포함하는 모든 문자열을 의미하는 패턴은 다음과 같다.

```
String pattern = ".*"+input+".*"; // input을 포함하는 모든 문자열
Pattern p = Pattern.compile(pattern);
```

패턴을 정의하였으니, 이제는 반복문으로 배열 phoneNumArr의 전화번호를 하나씩 읽어서 패턴과 일치하는지 확인한다. 이때 주의해야할 것은 사용자가 "234"를 입력했을 때 "012-3456-8790"과 "088-2346-9870"이 모두 검색될 수 있도록 전화번호에서 '-'를 제거한 후에 패턴과 일치하는지 확인해야한다는 것이다.

```
for(int i=0; i< phoneNumArr.length;i++) {
    String phoneNum = phoneNumArr[i];
    String tmp = phoneNum.replace("-",""); // phoneNum에서 '-'를 제거

    Matcher m = p.matcher(tmp);

    if(m.find()) {     // 패턴과 일치하는 부분을 찾으면, list에 phoneNum을 추가한다.
        list.add(phoneNum);
    }
}
```

[연습문제 - 모범답안]

[10-1] Calendar클래스와 SimpleDateFormat클래스를 이용해서 2010년의 매월 두 번째 일요일의 날짜를 출력하시오.

[실행결과]

```
2010-01-10은 2번째 일요일입니다.
2010-02-14은 2번째 일요일입니다.
2010-03-14은 2번째 일요일입니다.
2010-04-11은 2번째 일요일입니다.
2010-05-09은 2번째 일요일입니다.
2010-06-13은 2번째 일요일입니다.
2010-07-11은 2번째 일요일입니다.
2010-08-08은 2번째 일요일입니다.
2010-09-12은 2번째 일요일입니다.
2010-10-10은 2번째 일요일입니다.
2010-11-14은 2번째 일요일입니다.
2010-12-12은 2번째 일요일입니다.
```

[정답]

[연습문제]/ch10/Exercise10_1.java

```
import java.util.*;
import java.text.*;

class Exercise10_1
{
   public static void main(String[] args)
   {
        Calendar cal = Calendar.getInstance();

        cal.set(2010,0,1); // cal의 날짜를 2010년 1월 1일로 설정한다.

        for(int i=0; i < 12;i++) {
            int weekday = cal.get(Calendar.DAY_OF_WEEK); // 1일의 요일을 구한다.

            // 두 번째 일요일은 1일의 요일에 따라 달라진다.
            // 1일이 일요일인 경우에는 두번째 일요일은 8일이고,
            // 1일이 다른 요일일 때는 16에서 1일의 요일(weekday)을 빼면 알 수 있다.
            int secondSunday = (weekday==1) ? 8 : 16 - weekday;

            // 두 번째 일요일(secondSunday)로 cal의 날짜(DAY_OF_MONTH)를 바꾼다.
            cal.set(Calendar.DAY_OF_MONTH, secondSunday);

            Date d = cal.getTime(); // Calendar를 Date로 변환한다.
            System.out.println(new SimpleDateFormat("yyyy-MM-dd은 F번째 E요일입니
다.").format(d));

            // 날짜를 다음달 1일로 변경한다.
            cal.add(Calendar.MONTH, 1);
            cal.set(Calendar.DAY_OF_MONTH,1);
        }
```

```
    }
}
```

[해설] 매월 두 번째 일요일(secondSunday)을 구하려면, 매월 1일이 무슨 요일인지 알아내야한다. 만일 1일이 일요일이라면 2번째 일요일은 8일이 된다. 1일이 월요일이라면 2번째 일요일은 14일이 된다. 1일이 일요일인 경우를 제외하고는 1일의 요일(weekday)과 2번째 일요일의 날짜를 더하면 일정한 값(16)이라는 것을 알 수 있다. 즉, 16에서 1일의 요일(weekday)을 빼면 2번째 일요일이 며칠인지 알 수 있는 것이다.(1일이 일요일인 경우에는 9에서 1을 뺀 8이 된다.)

```
int secondSunday = (weekday==1) ? 8 : 16 - weekday;
```

1일의 요일	weekday	2번째 일요일	weekday+2번째 일요일
일	1	8일	9
월	2	14일	16
화	3	13일	16
수	4	12일	16
목	5	11일	16
금	6	10일	16
토	7	9일	16

이제 두 번째 일요일의 날짜(secondSunday)를 알아냈으니, set()를 사용해서 cal의 날짜(DAY_OF_MONTH)를 두 번째 일요일의 날짜로 변경한다. SimpleDateFormat클래스의 format메서드는 매개변수로 Date타입을 받기 때문에 cal.getTime()을 호출해서 Calendar를 Date로 변환해야한다.

```
        // 두 번째 일요일(secondSunday)로 cal의 날짜(DAY_OF_MONTH)를 바꾼다.
        cal.set(Calendar.DAY_OF_MONTH, secondSunday);

        Date d = cal.getTime(); // Calendar를 Date로 변환한다.
        System.out.println(new SimpleDateFormat("yyyy-MM-dd은 F번째 E요일입니
다.").format(d));
```

[10-2] 어떤 회사의 월급날이 매월 21일이다. 두 날짜 사이에 월급날이 몇 번있는지 계산해서 반환하는 메서드를 작성하고 테스트 하시오.

[연습문제]/ch10/Exercise10_2.java

```
import java.util.*;
import java.text.*;

class Exercise10_2 {
   static int paycheckCount(Calendar from, Calendar to) {
//   1. from 또는 to가 null이면 0을 반환한다.
       if(from==null || to==null) return 0;

//   2. from와 to가 같고 날짜가 21일이면 1을 반환한다.
       if(from.equals(to) && from.get(Calendar.DAY_OF_MONTH)==21) {
           return 1;
       }

       int fromYear = from.get(Calendar.YEAR);
       int fromMon  = from.get(Calendar.MONTH);
       int fromDay  = from.get(Calendar.DAY_OF_MONTH);

       int toYear = to.get(Calendar.YEAR);
       int toMon  = to.get(Calendar.MONTH);
       int toDay  = to.get(Calendar.DAY_OF_MONTH);

//   3. to와 from이 몇 개월 차이인지 계산해서 변수 monDiff에 담는다.
       int monDiff = (toYear * 12 + toMon) - (fromYear * 12 + fromMon);

//   4. monDiff가 음수이면 0을 반환한다.
       if(monDiff < 0) return 0;

//   5. 만일 from의 일(DAY_OF_MONTH)이 21일이거나 이전이고
//      to의 일(DAY_OF_MONTH)이 21일이거나 이후이면 monDiff의 값을 1 증가시킨다.
       if(fromDay <= 21 && toDay >= 21)
           monDiff++;

//   6. 만일 from의 일(DAY_OF_MONTH)이 21일 이후고
//      to의 일(DAY_OF_MONTH)이 21일 이전이면 monDiff의 값을 1 감소시킨다.
       if(fromDay > 21 && toDay < 21)
           monDiff--;

       return monDiff;
   }

   static void printResult(Calendar from, Calendar to) {
       Date fromDate = from.getTime();
       Date toDate   = to.getTime();

       SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");

       System.out.print(sdf.format(fromDate)+" ~ "
                        +sdf.format(toDate)+":");
       System.out.println(paycheckCount(from, to));
   }

   public static void main(String[] args) {
       Calendar fromCal = Calendar.getInstance();
```

```
        Calendar toCal   = Calendar.getInstance();

        fromCal.set(2010,0,1);
        toCal.set(2010,0,1);
        printResult(fromCal, toCal);

        fromCal.set(2010,0,21);
        toCal.set(2010,0,21);
        printResult(fromCal, toCal);

        fromCal.set(2010,0,1);
        toCal.set(2010,2,1);
        printResult(fromCal, toCal);

        fromCal.set(2010,0,1);
        toCal.set(2010,2,23);
        printResult(fromCal, toCal);

        fromCal.set(2010,0,23);
        toCal.set(2010,2,21);
        printResult(fromCal, toCal);

        fromCal.set(2011,0,22);
        toCal.set(2010,2,21);
        printResult(fromCal, toCal);
    }
}
```

[실행결과]

```
2010-01-01 ~ 2010-01-01:0
2010-01-21 ~ 2010-01-21:1
2010-01-01 ~ 2010-03-01:2
2010-01-01 ~ 2010-03-23:3
2010-01-23 ~ 2010-03-21:2
2011-01-22 ~ 2010-03-21:0
```

[해설] 지정된 날짜범위에 21일이 몇 번 포함되는지 계산하려면, 범위의 시작일 과 마지막일 간의 개월 수 차이를 구한 다음에 시작일 또는 마지막일이 21일인지 아닌지 확인해서 둘 다 21일이면 1을 더하고 둘 다 21일이 아니면 1을 뺀다.

[참고] 1~5의 범위의 정수가 몇 개 인지 알려면 범위의 끝 값인 5에서 시작 값인 1을 뺀 다음에 1과 5가 범위에 포함된다면 1을 더해서 5-1+1=5가 된다.(1,2,3,4,5) 그러나 1과 5가 범위에 포함되지 않는 다면 1을 빼서 5-1-1=3이 된다.(2,3,4) 어느 한쪽 경계값만 범위에 포함된다면 두 경계값의 차이(5-1=4)가 범위에 포함된 정수의 개수이다.

```
//   3. to와 from이 몇 개월 차이인지 계산해서 변수 monDiff에 담는다.
     int monDiff = (toYear * 12 + toMon) - (fromYear * 12 + fromMon);
//   4. monDiff가 음수이면 0을 반환한다.
     if(monDiff < 0) return 0;
//   5. 만일 from의 일(DAY_OF_MONTH)이 21일이거나 이전이고
//      to의 일(DAY_OF_MONTH)이 21일이거나 이후이면 monDiff의 값을 1 증가시킨다.
     if(fromDay <= 21 && toDay >= 21)
         monDiff++;
//   6. 만일 from의 일(DAY_OF_MONTH)이 21일 이후고
//      to의 일(DAY_OF_MONTH)이 21일 이전이면 monDiff의 값을 1 감소시킨다.
     if(fromDay > 21 && toDay < 21)
         monDiff--;
```

[10-3] 문자열 "123,456,789.5"를 소수점 첫 번째 자리에서 반올림하고, 그 값을 만 단위마다 콤마(,)로 구분해서 출력하시오.

[실행결과]

```
data:123,456,789.5
반올림:123456790
만단위:1,2345,6790
```

[정답]

[연습문제]/ch10/Exercise10_3.java

```
import java.text.*;

class Exercise10_3
{
   public static void main(String[] args)
   {
        String data = "123,456,789.5";

        DecimalFormat df = new DecimalFormat("#,###.##"); // 변환할 문자열의 형식을 지정
        DecimalFormat df2 = new DecimalFormat("#,####");

        try {
            Number num = df.parse(data);

            double d = num.doubleValue();
            System.out.println("data:"+data);
            System.out.println("반올림:"+Math.round(d));
            System.out.println("만단위:"+df2.format(d));
        } catch(Exception e) {}
   } // main
}
```

[해설] 특정 형식의 문자열을 숫자로 변환하려면 DecimalFormat클래스에 형식을 정의해준 다음에 parse()를 이용하면 된다. parse()의 반환타입이 Number이기 때문에, Number에서 다시 doubleValue()를 호출해서 double타입의 값을 얻어야 한다.

```
DecimalFormat df = new DecimalFormat("#,###.##"); // 변환할 문자열의 형식을 지정
Number num = df.parse(data);   // 문자열(data)에서 숫자를 뽑아내서 Number를 반환한다.
double d = num.doubleValue();  // Number에서 숫자를 double형태로 뽑아낸다.
```

format()은 주어진 값을 정해진 형식에 맞게 변환한 다음에 문자열(String)로 반환한다.

```
DecimalFormat df2 = new DecimalFormat("#,####");
System.out.println("만단위:"+df2.format(d));
```

[10-4] 화면으로부터 날짜를 "2007/05/11"의 형태로 입력받아서 무슨 요일인지 출력하는 프로그램을 작성하시오.
단, 입력된 날짜의 형식이 잘못된 경우 메세지를 보여주고 다시 입력받아야 한다.

[실행결과]

```
날짜를 yyyy/MM/dd의 형태로 입력해주세요.(입력예:2007/05/11)
>>2009-12-12
날짜를 yyyy/MM/dd의 형태로 입력해주세요.(입력예:2007/05/11)
>>2009/12/12
입력하신 날짜는 토요일입니다.
```

[정답]

[연습문제]/ch10/Exercise10_4.java

```
import java.util.*;
import java.text.*;

class Exercise10_4 {
   public static void main(String[] args) {
        String pattern  = "yyyy/MM/dd";
        String pattern2 = "입력하신 날짜는 E요일입니다."; // 'E'는 일~토 중의 하나가 된다.

        DateFormat df  = new SimpleDateFormat(pattern);
        DateFormat df2 = new SimpleDateFormat(pattern2);

        Scanner s = new Scanner(System.in);

        Date inDate = null;

        do {
            System.out.println("날짜를 " + pattern
                                      + "의 형태로 입력해주세요.(입력예:2007/05/11)");

            try {
                System.out.print(">>");
                inDate = df.parse(s.nextLine()); // 입력받은 날짜를 Date로 변환한다.
                break;  // parse()에서 예외가 발생하면 이 문장은 수행되지 않는다.
            } catch(Exception e) {}
        } while(true);

        System.out.println(df2.format(inDate));
   } // main
}
```

[해설] SimpleDateFormat은 날짜를 지정된 형식으로 출력하거나 문자열을 지정된 형식으로 변환 또는 유효성 검사에 사용할 수 있다. 이 문제에서는 입력된 날짜가 지정된 형식에 맞는지 검사하고, 변환하는 역할을 한다.
Scanner를 이용해서 화면으로 부터 날짜를 입력받고, 입력받은 날짜(문자열)가 지정된 형식과 다르면 parse()에서 ParseException이 발생한다.

```
do {
    System.out.println("날짜를 " + pattern
                       + "의 형태로 입력해주세요.(입력예:2007/05/11)");
    try {
        System.out.print(">>");
        inDate = df.parse(s.nextLine()); // ParseException이 발생할 수 있다.
        break;  // parse()에서 예외가 발생하면 이 문장은 수행되지 않는다.
    } catch(Exception e) {}
} while(true);
```

parse()에서 예외가 발생하면 break문이 수행되지 않기 때문에 예외가 발생하지 않을 때까지, 즉 지정된 형식에 맞게 날짜가 입력될 때까지 반복하게 된다.

[10-5] 다음과 같이 정의된 메서드를 작성하고 테스트하시오.

메서드명 : getDayDiff
기　　능 : yyyymmdd형식의 두 문자열을 넘겨받으면 두 날짜의 차이를 일(day)단위로 반환한다.
　　단, 첫 번째 날짜 빼기 두 번째 날짜의 결과를 반환한다.
　　만일 주어진 문자열이 유효하지 않으면 0을 반환한다.
반환타입 : int
매개변수 : String yyyymmdd1 - 시작날짜
　　String yyyymmdd2 - 끝 날짜

[연습문제]/ch10/Exercise10_5.java

```
import java.util.*;

class Exercise10_5 {
   static int getDayDiff(String yyyymmdd1, String yyyymmdd2) {
        int diff = 0;

        try {
            int year1  = Integer.parseInt(yyyymmdd1.substring(0,4));
            int month1 = Integer.parseInt(yyyymmdd1.substring(4,6)) - 1;
            int day1   = Integer.parseInt(yyyymmdd1.substring(6,8));

            int year2  = Integer.parseInt(yyyymmdd2.substring(0,4));
            int month2 = Integer.parseInt(yyyymmdd2.substring(4,6)) - 1;
            int day2   = Integer.parseInt(yyyymmdd2.substring(6,8));

            Calendar date1 = Calendar.getInstance();
            Calendar date2 = Calendar.getInstance();

            date1.set(year1, month1, day1);
            date2.set(year2, month2, day2);

            diff = (int)((date1.getTimeInMillis()-date2.getTimeInMillis())
                                                        /(24*60*60*1000));
        } catch(Exception e) {
            diff = 0;  // substring(), parseInt()에서 예외가 발생하면 0을 반환한다.
        }

        return diff;
   }

   public static void main(String[] args){
        System.out.println(getDayDiff("20010103","20010101"));
        System.out.println(getDayDiff("20010103","20010103"));
        System.out.println(getDayDiff("20010103","200103"));
   }
}
```

[실행결과]

```
2
0
0
```

[해설] 넘겨받은 문자열을 substring()으로 잘라서 년, 월, 일을 각각 구한 다음, 이 값들을 가지고 Calendar의 년월일을 설정한다.

[참고] Calendar클래스의 month값은 1이 아닌 0부터 시작하기 때문에 1을 빼주어야 한다.

```
int year1  = Integer.parseInt(yyyymmdd1.substring(0,4));
int month1 = Integer.parseInt(yyyymmdd1.substring(4,6)) - 1;
int day1   = Integer.parseInt(yyyymmdd1.substring(6,8));

Calendar date1 = Calendar.getInstance();
date1.set(year1, month1, day1);
```

Calendar의 getTimeInMillis()는 날짜를 천분의 일초단위로 변환해서 반환한다. getTimeMillis()를 이용해서 두 날짜를 천분의 일초로 변환해서 차이를 구한 다음에 일(day) 단위로 변환하면 두 날짜의 날(day) 차이를 구할 수 있다.

천분의 일초 단위를 일 단위로 바꾸려면 24*60*60*1000로 나눠주면 된다.(1일 = 24시간 = 24*60분 = 24*60*60초 = 24*60*60*1000밀리세컨드)

```
date1.set(year1, month1, day1);
date2.set(year2, month2, day2);

diff = (int)((date1.getTimeInMillis()-date2.getTimeInMillis())
                                                    /(24*60*60*1000));
```

[10-6] 자신이 태어난 날부터 지금까지 며칠이 지났는지 계산해서 출력하시오.

[실행결과]

```
birth day=2000-01-01
today    =2016-01-29
5872 days
```

[정답] LocalDate의 until()을 이용하면 쉽게 해결된다.

[연습문제]/ch10/Exercise10_6.java

```
import java.time.*;
import java.time.temporal.*;

class Exercise10_6 {
   public static void main(String[] args) {
        LocalDate birthDay = LocalDate.of(2000, 1, 1); // 자신의 생일을 지정
        LocalDate now      = LocalDate.now();

        long days = birthDay.until(now, ChronoUnit.DAYS);

        System.out.println("birth day="+birthDay);
        System.out.println("today    ="+now);
        System.out.println(days +" days");
   }
}
```

[10-7] 2016년 12월 넷째주 화요일의 날짜를 아래와 같이 출력하시오.

[실행결과]

```
2016-12-27
```

[정답] ?째주 ?요일은 TemporalAdjusters클래스의 dayOfWeekInMonth()를 이용하면 된다.

[연습문제]/ch10/Exercise10_7.java

```
import java.time.*;
import static java.time.DayOfWeek.*;
import static java.time.temporal.TemporalAdjusters.*;

class Exercise10_7 {
   public static void main(String[] args) {
        LocalDate date = LocalDate.of(2016, 12, 1);
        System.out.println(date.with(dayOfWeekInMonth(4, TUESDAY)));
   }
}
```

[10-8] 서울과 뉴욕간의 시차가 얼마인지 계산하여 출력하시오.

[실행결과]

```
2016-01-28T23:01:00.136+09:00[Asia/Seoul]
2016-01-28T09:01:00.138-05:00[America/New_York]
sec1=32400
sec2=-18000
diff=14 hrs
```

[정답]

[연습문제]/ch10/Exercise10_8.java

```
import java.time.*;

class Exercise10_8 {
   public static void main(String[] args) {
        ZonedDateTime zdt = ZonedDateTime.now();
        ZoneId nyId = ZoneId.of("America/New_York");
        ZonedDateTime zdtNY = ZonedDateTime.now().withZoneSameInstant(nyId);

        System.out.println(zdt);
        System.out.println(zdtNY);

        long sec1 = zdt.getOffset().getTotalSeconds();
        long sec2 = zdtNY.getOffset().getTotalSeconds();
        long diff = (sec1 - sec2)/3600;

        System.out.println("sec1="+sec1);
        System.out.println("sec2="+sec2);
        System.out.printf("diff=%d hrs%n",diff);
   }
}
```

[해설] ZonedDateTime의 getOffset()은 ZoneOffset을 반환한다. ZoneOffset의 getTotalSeconds()를 호출하면, 날짜와 시간을 초단위로 변환한 결과를 얻을 수 있다.

```
ZoneOffset offset = zdt.getOffset();
long sec1 = offset.getTotalSeconds();
```

현재 서울과 뉴욕의 날짜와 시간을 초단위로 바꾼다음에, 차이를 구하면 시차를 초단위로 구할 수 있다. 이 값을 3600초(1시간)으로 나누면, 시간단위의 값을 구할 수 있다.

[연습문제 - 모범답안]

[11-1] 다음은 정수집합 1,2,3,4와 3,4,5,6의 교집합, 차집합, 합집합을 구하는 코드이다. 코드를 완성하여 실행결과와 같은 결과를 출력하시오.

[Hint] ArrayList클래스의 addAll(), removeAll(), retainAll()을 사용하라.

[연습문제]/ch11/Exercise11_1.java

```
import java.util.*;

class Exercise11_1 {
   public static void main(String[] args) {
        ArrayList list1 = new ArrayList();
        ArrayList list2 = new ArrayList();
        ArrayList kyo = new ArrayList();  // 교집합
        ArrayList cha = new ArrayList();  // 차집합
        ArrayList hap = new ArrayList();  // 합집합

        list1.add(1);
        list1.add(2);
        list1.add(3);
        list1.add(4);

        list2.add(3);
        list2.add(4);
        list2.add(5);
        list2.add(6);

        kyo.addAll(list1);     // list1의 모든 요소를 kyo에 저장한다.
        kyo.retainAll(list2); // list2와 kyo의 공통요소만 남기고 삭제한다.

        cha.addAll(list1);
        cha.removeAll(list2); // cha에서 list2와 공통된 요소들을 모두 삭제한다.

        hap.addAll(list1);    // list1의 모든 요소를 hap에 저장한다.
        hap.removeAll(kyo);   // hap에서 kyo와 공통된 모든 요소를 삭제한다.
        hap.addAll(list2);    // list2의 모든 요소를 hap에 저장한다.

        System.out.println("list1="+list1);
        System.out.println("list2="+list2);
        System.out.println("kyo="+kyo);
        System.out.println("cha="+cha);
        System.out.println("hap="+hap);
   }
}
```

[실행결과]

```
list1=[1, 2, 3, 4]
list2=[3, 4, 5, 6]
kyo=[3, 4]
cha=[1, 2]
hap=[1, 2, 3, 4, 5, 6]
```

[11-2] 다음 코드의 실행결과를 적으시오.

[연습문제]/ch11/Exercise11_2.java

```
import java.util.*;

class Exercise11_2 {
   public static void main(String[] args) {
        ArrayList list = new ArrayList();
        list.add(3);
        list.add(6);
        list.add(2);
        list.add(2);
        list.add(2);
        list.add(7);

        HashSet set = new HashSet(list);  // 중복요소들이 제거되고 순서유지 안됨 2,6,3,7
        TreeSet tset = new TreeSet(set);  // 오름차순으로 정렬된다. 2,3,6,7
        Stack stack = new Stack();        // Stack에 넣었다 꺼내면 저장순서가 반대가 된다.
        stack.addAll(tset);               // TreeSet의 저장된 모든 요소를 stack에 담는다.

        while(!stack.empty())
            System.out.println(stack.pop()); // stack에 저장된 값을 하나씩 꺼낸다.
   }
}
```

[정답]

[실행결과]

```
7
6
3
2
```

[해설] 각 컬렉션 클래스들의 특징을 이해하고 있는지 확인하는 문제이다. ArrayList는 중복을 허용하고 저장순서를 유지한다. HashSet은 중복을 허용하지 않기 때문에 중복요소들은 저장되지 않는다. 그래서 HashSet에는 2,6,3,7이 저장된다.(HashSet은 저장순서를 유지하지 않는 다는 사실을 잊지 말자)

```
HashSet set = new HashSet(list);  // 중복요소들이 제거되고 순서유지 안됨 2,6,3,7
```

TreeSet은 정렬해서 저장하기 때문에, 그리고 따로 정렬기준을 주지 않았기 때문에 숫자의 기본정렬인 오름차순으로 정렬한다. 그래서 2,3,6,7이 된다.

```
TreeSet tset = new TreeSet(set);  // 오름차순으로 정렬된다. 2,3,6,7
```

Stack은 FILO구조(처음 넣은 것이 마지막에 나오는 구조)로 되어 있기 때문에 TreeSet의 모든 요소들을 저장한 다음 다시 꺼내면 저장한 순서와 반대가 된다.

```
Stack stack = new Stack();        // Stack에 넣었다 꺼내면 저장순서가 반대가 된다.
stack.addAll(tset);               // TreeSet의 저장된 모든 요소를 stack에 담는다.

while(!stack.empty())
    System.out.println(stack.pop()); // stack에 저장된 값을 하나씩 꺼낸다.
```

[11-3] 다음 중 ArrayList에서 제일 비용이 많이 드는 작업은? 단, 작업도중에 ArrayList의 크기 변경이 발생하지 않는다고 가정한다.

a. 첫 번째 요소 삭제

b. 마지막 요소 삭제

c. 마지막에 새로운 요소 추가

d. 중간에 새로운 요소 추가

[정답] a

[해설] ArrayList는 배열을 기반으로 하고, 배열은 크기를 변경할 수 없기 때문에 저장할 공간이 부족하면 새로운 배열을 만들고 내용을 복사해야하므로 많은 비용이 든다.

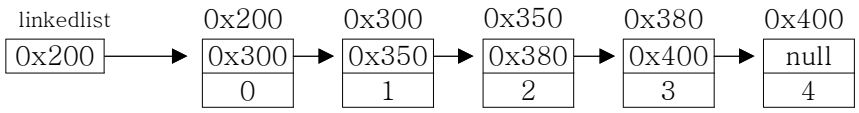
그리고 배열의 중간에 새로운 요소를 추가 또는 삭제하는 것은 다른 요소들을 이동시켜야하기 때문에 배열을 새로 생성하는 것보다는 적지만 역시 비용이 많이 드는 작업이다.

특히 배열의 첫 번째 요소를 삭제하면, 빈자리를 채우기 위해 나머지 모든 요소들을 이동시켜야 하기 때문에 많은 비용이 든다.

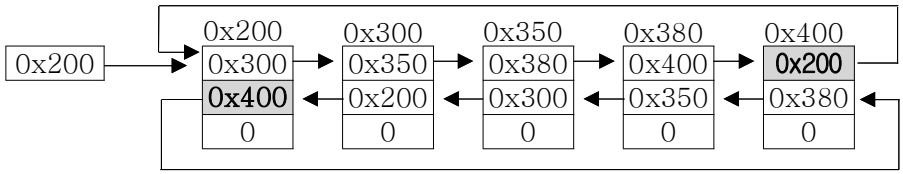
반면에 ArrayList의 마지막에 요소를 추가 또는 삭제하는 것은 다른 요소들을 이동시킬 필요가 없기 때문에 아주 적은 비용만으로 처리가 가능하다.

[11-4] LinkedList클래스는 이름과 달리 실제로는 이중 원형 연결리스트(doubly circular linked list)로 구현되어 있다. LinkedList인스턴스를 생성하고 11개의 요소를 추가했을 때, 이 11개의 요소 중 접근시간(access time)이 가장 오래 걸리는 요소는 몇 번째 요소인가?

[정답] 여섯 번째 요소(LinkedList에서 제일 가운데 위치한 요소)
[해설] LinkedList는 각 요소가 서로 참조로 연결되어 있어서, n번째 요소에 접근하기 위해서는 첫 번째 요소부터 순서대로 각 요소를 거쳐야 된다. 예를 들어 세 번째 요소에 접근하기 위해서는 첫 번째 요소에서 두 번째 요소로, 두 번째 요소에서 세 번째 요소로 이동해야 한다.



이런 식이면 LinkedList의 마지막 요소에 접근하는 것이 시간이 제일 많이 걸릴 것 같은데, 그렇지 않은 이유는 LinkedList가 실제로는 아래의 그림과 같은 이중 원형 연결리스트로 되어 있기 때문이다.



[그림11-11] 더블 써큘러 링크드리스트(이중 원형 연결리스트, doubly circular linked list)

이중 원형 연결리스트는 첫 번째 요소와 마지막 요소를 연결해서 LinkedList의 단점인 접근성(accessibility)를 향상시킨 것이다. 이중 원형 연결리스트에서는 마지막 요소에 접근하기 위해서는 첫 번째 요소에서 한 번만 이동하면 된다. 마지막 요소에서 첫 번째 요소에 접근하기 위해서도 역시 한 번만 이동하면 된다.

이것은 마치 Tv리모콘의 첫 번째 채널에서 채널을 감소시키면 마지막 채널로 이동하고, 마지막 채널에서 채널을 증가시키면 첫 번째 채널로 이동하는 것과 같다고 할 수 있다.

그래서 LinkedList의 제일 가운데 있는 요소가 접근시간이 가장 길다는 것을 알 수 있다.

[11-5] 다음에 제시된 Student클래스가 Comparable인터페이스를 구현하도록 변경해서 이름(name)이 기본 정렬기준이 되도록 하시오.

[연습문제]/ch11/Exercise11_5.java

```
import java.util.*;

class Student implements Comparable {
   String name;
   int    ban;
   int    no;
   int    kor, eng,  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+0.5)/10f;
   }

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

   public int compareTo(Object o) {
        if(o instanceof Student) {
            Student tmp = (Student)o;

            return name.compareTo(tmp.name); // String클래스의 compareTo()를 호출
        } else {
            return -1;
        }
   }
}

class Exercise11_5 {
   public static void main(String[] args) {
        ArrayList list = new ArrayList();
        list.add(new Student("홍길동",1,1,100,100,100));
        list.add(new Student("남궁성",1,2,90,70,80));
        list.add(new Student("김자바",1,3,80,80,90));
        list.add(new Student("이자바",1,4,70,90,70));
        list.add(new Student("안자바",1,5,60,100,80));

        Collections.sort(list); // list에 저장된 요소들을 정렬한다.(compareTo()호출)
        Iterator it = list.iterator();
```

```
        while(it.hasNext())
            System.out.println(it.next());
    }
}
```

[실행결과]

```
김자바,1,3,80,80,90,250,83.3
남궁성,1,2,90,70,80,240,80.0
안자바,1,5,60,100,80,240,80.0
이자바,1,4,70,90,70,230,76.7
홍길동,1,1,100,100,100,300,100.0
```

[해설] Collections.sort(List list)를 이용하면 ArrayList에 저장된 요소들을 쉽게 정렬할 수 있다.

```
Collections.sort(list); // list에 저장된 요소들을 정렬한다.(compareTo()호출)
```

그러나 한 가지 조건이 있다. ArrayList에 저장된 요소들은 모두 Comparable인터페이스를 구현한 것이어야 한다는 것이다. 이 인터페이스에는 compareTo메서드가 정의되어 있는데, 이 메서드는 Collections.sort(List list)에 의해 호출되어 정렬기준을 제공하게 된다.

```
public interface Comparable {
    public int compareTo(Object o); // 주어진 객체(o)와 자신의 멤버변수를 비교
}
```

compareTo메서드는 매개변수로 주어진 객체(o)를 인스턴스자신과 비교해서 자신이 작으면 음수를, 같으면 0을, 크면 양수를 반환하도록 구현되어야 한다.

문제에서는 학생의 이름으로 정렬될 것을 요구했으므로, 인스턴스변수 name만 비교하도록 compareTo메서드를 구현하면 된다. 문자열 name을 어떻게 비교하도록 구현해야할까? 고민되겠지만, String클래스에는 이미 문자열 비교를 위한 compareTo메서드를 구현해 놓았고 우리는 단지 그것을 활용하기만 하면 된다.

```
public final class String
    implements java.io.Serializable, Comparable<String>, CharSequence
{
...
```

```
class Student implements Comparable {
...
public int compareTo(Object o) {
   if(o instanceof Student) {
       Student tmp = (Student)o;
       return name.compareTo(tmp.name); // String클래스의 compareTo()를 호출
   } else {
       return -1;
   }
}
```

참고로 지네릭스(generics)를 적용한 예제를 추가한다. 어떤 차이가 있는지 잘 비교해 보자. 아직 지네릭스를 배우지 않았으므로, 나중에 지네릭스를 배우고 나서 봐도 좋다.

[연습문제]/ch11/Exercise11_5_2.java

```
import java.util.*;

class Student implements Comparable<Student>
{
   String name;
   int    ban;
   int    no;
   int    kor, eng,  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+0.5)/10f;
   }

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

   public int compareTo(Student s) {
            return name.compareTo(s.name);
   }
}

class Exercise11_5_2 {
   public static void main(String[] args) {
        ArrayList<Student> list = new ArrayList<Student>();
        list.add(new Student("홍길동",1,1,100,100,100));
        list.add(new Student("남궁성",1,2,90,70,80));
        list.add(new Student("김자바",1,3,80,80,90));
        list.add(new Student("이자바",1,4,70,90,70));
        list.add(new Student("안자바",1,5,60,100,80));

        Collections.sort(list);
        Iterator it = list.iterator();

        while(it.hasNext())
            System.out.println(it.next());
   }
}
```

[11-6] 다음의 코드는 성적평균의 범위별로 학생 수를 세기 위한 것이다. TreeSet이 학생들의 평균을 기준으로 정렬하도록 compare(Object o1, Object o2)와 평균점수의 범위를 주면 해당 범위에 속한 학생의 수를 반환하는 getGroupCount()를 완성하라.

[Hint] TreeSet의 subSet(Object from, Object to)를 사용하라.

[연습문제]/ch11/Exercise11_6.java

```
import java.util.*;

class Student implements Comparable {
   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+0.5)/10f;
   }

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

   public int compareTo(Object o) {
       if(o instanceof Student) {
           Student tmp = (Student)o;

           return name.compareTo(tmp.name);
       } else {
           return -1;
       }
   }
} // class Student
```

```
class Exercise11_6 {
   static int getGroupCount(TreeSet tset, int from, int to) {
        Student s1 = new Student("",0,0,from,from,from);
        Student s2 = new Student("",0,0,to,to,to);

        return tset.subSet(s1,s2).size();
   }

   public static void main(String[] args) {
        TreeSet set = new TreeSet(new Comparator() {  // 익명클래스
            public int compare(Object o1, Object o2) {
                if(o1 instanceof Student && o2 instanceof Student) {
                    Student s1 = (Student)o1;
                    Student s2 = (Student)o2;

                    return (int)(s1.getAverage() - s2.getAverage());
                }

                return -1;
            }
        });

        set.add(new Student("홍길동",1,1,100,100,100));
        set.add(new Student("남궁성",1,2,90,70,80));
        set.add(new Student("김자바",1,3,80,80,90));
        set.add(new Student("이자바",1,4,70,90,70));
        set.add(new Student("안자바",1,5,60,100,80));

        Iterator it = set.iterator();

        while(it.hasNext())
            System.out.println(it.next());

        System.out.println("[60~69] :"+getGroupCount(set,60,70));
        System.out.println("[70~79] :"+getGroupCount(set,70,80));
        System.out.println("[80~89] :"+getGroupCount(set,80,90));
        System.out.println("[90~100] :"+getGroupCount(set,90,101));
   }
}
```

[실행결과]

```
이자바,1,4,70,90,70,230,76.7
남궁성,1,2,90,70,80,240,80.0
김자바,1,3,80,80,90,250,83.3
홍길동,1,1,100,100,100,300,100.0
[60~69] :0
[70~79] :1
[80~89] :2
[90~100] :1
```

[해설] 평균이 지정된 범위(from~to)에 속하는 학생의 수를 세는 메서드 getGroupCount를 작성하는 문제이다. TreeSet은 정렬된 상태로 요소(element)를 저장하고, 검색과 정렬 특히 부분정렬에 유리하다. TreeSet클래스의 subSet()은 지정된 범위에 속하는 요소들이 포함된 TreeSet을 반환한다. 단, 끝 범위의 경계값(to)은 결과에 포함되지 않는다.(from ≤ x < to)

```
static int getGroupCount(TreeSet tset, int from, int to) {
    Student s1 = new Student("",0,0,from,from,from);
    Student s2 = new Student("",0,0,to,to,to);

    return tset.subSet(s1,s2).size();
}
```

왼쪽의 코드는 오른쪽의 코드를 한 줄로 간단히 쓴 것이다. 두 코드를 잘 비교해 보자.

```
return tset.subSet(s1,s2).size();
```

```
TreeSet tmp = tset.subSet(s1,s2);
return tmp.size(); // 크기를 반환
```

compare(Object o1, Object o2)는 주어진 두 객체를 비교해서 그 결과를 양수, 0, 음수 중의 하나로 반환한다. 그래서 학생점수의 평균을 반환하는 getAverage()로 계산한 다음에 int타입으로 형변환해주었다.

```
TreeSet set = new TreeSet(new Comparator() {  // 익명클래스
  public int compare(Object o1, Object o2) {
      if(o1 instanceof Student && o2 instanceof Student) {
          Student s1 = (Student)o1;
          Student s2 = (Student)o2;
              return (int)(s1.getAverage() - s2.getAverage());
      }

      return -1;
  }
```

TreeSet은 요소(element)를 저장할 때 정렬해서 저장하는 데, 그 정렬기준은 어떤 것일까? 이미 배운 것과 같이 Comparable인터페이스를 구현한 클래스, 예를 들면, Integer와 같은 wrapper클래스나 String은 정렬기준이 내부에 정의되어 있다.

이처럼 Comparable인터페이스를 구현해서 내부적으로 정렬기준을 가지고 있는 클래스의 객체들은 TreeSet에 저장할 때, 정렬기준을 지정해주지 않아도 되지만, Comparable인터페이스를 구현하지 않은 클래스(정렬기준이 없는 클래스)는 생성자 TreeSet(Comparator c)를 이용해서 반드시 정렬기준을 제공해주어야 한다.

정렬기준이 내부에 정의되어 있어도 다른 기준으로 정렬하고 싶을 때 이 생성자를 사용하면 된다.

Student클래스는 Comparable인터페이스를 구현함으로써 기본 정렬기준을 정의하였다.

```
class Student implements Comparable {
   String name;
   ...
   public int compareTo(Object o) {
        if(o instanceof Student) {
            Student tmp = (Student)o;

            return name.compareTo(tmp.name); // 이름을 기준으로 정렬(오름차순)
//          return tmp.name.compareTo(name); // 이름을 기준으로 정렬(내림차순)
        } else {
            return -1;
        }
   } // public int comareTo(Object o)
}
```

이와 달리 평균을 기준으로 오름차순 정렬을 하기위해서는 다른 정렬기준을 제공해야한다. 그래서 Comparator인터페이스를 구현한 클래스를 작성하고 이 클래스의 인스턴스를 생성자 TreeSet(Comparator c)의 매개변수로 넘겨주었다.

```
// 생성자 TreeSet(Comparator c)를 사용.
TreeSet set = new TreeSet(new Comparator() {  // 익명 클래스
   public int compare(Object o1, Object o2) {
        ...
   }
}
```

매개변수를 통해 넘겨진 정렬기준은 TreeSet의 멤버변수로 정의된 comparator에 저장된다. 그리고 이 comparator는 TreeSet에 요소를 저장할 때 사용되는 메서드 put()에서 저장할 위치를 결정하기 위해 사용된다.

[참고] 아래의 코드는 TreeSet의 원본소스를 이해하기 쉽게 재구성한 것이므로 원본과 다르다.

```
TreeSet set = new TreeSet(new Comparator(){...}); // Comparator를 제공받는다.

class TreeSet {
   private Comparator comparator = null; // Comparator(정렬기준)가 정의되어 있다.

   TreeSet(Comparator c) {
        this.comparator = c; // 생성자를 통해 제공받은 정렬기준을 comparator로 설정
   }

   // TreeSet에 요소를 저장하는데 사용되는 메서드. 저장할 때 정렬해서 저장하므로 compare()를 호출
   public Object put(Object key, Object value) {
        ...
        while(true) {
            int cmp = comparator.compare(key, t.key);  // compare()를 호출한다.
            ...
        }
        ...
   }
}
```

[11-7] 다음에 제시된 BanNoAscending클래스를 완성하여, ArrayList에 담긴 Student인스턴스들이 반(ban)과 번호(no)로 오름차순 정렬되게 하시오.(반이 같은 경우 번호를 비교해서 정렬한다.)

[연습문제]/ch11/Exercise11_7.java

```
import java.util.*;

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+0.5)/10f;
   }

   public String toString() {
        return name
             +","+ban
             +","+no
             +","+kor
             +","+eng
             +","+math
             +","+getTotal()
             +","+getAverage()
             ;
   }
} // class Student

class BanNoAscending implements Comparator {
   public int compare(Object o1, Object o2) {
        if(o1 instanceof Student && o2 instanceof Student) {
             Student s1 = (Student)o1;
             Student s2 = (Student)o2;

             int result = s1.ban - s2.ban;

             if(result==0) {  // 반이 같으면, 번호를 비교한다.
                  return s1.no - s2.no;
```

```
            }

            return result;
        }

        return -1;
    }
}

class Exercise11_7 {
    public static void main(String[] args) {
        ArrayList list = new ArrayList();
        list.add(new Student("이자바",2,1,70,90,70));
        list.add(new Student("안자바",2,2,60,100,80));
        list.add(new Student("홍길동",1,3,100,100,100));
        list.add(new Student("남궁성",1,1,90,70,80));
        list.add(new Student("김자바",1,2,80,80,90));

        Collections.sort(list, new BanNoAscending());

        Iterator it = list.iterator();

        while(it.hasNext())
            System.out.println(it.next());
    }
}
```

[실행결과]

```
남궁성,1,1,90,70,80,240,80.0
김자바,1,2,80,80,90,250,83.3
홍길동,1,3,100,100,100,300,100.0
이자바,2,1,70,90,70,230,76.7
안자바,2,2,60,100,80,240,80.0
```

[해설] 조금 어렵다고 느꼈을지도 모르겠다. 그러나 답은 쉽다. 반(ban)을 뺄셈한 결과가 '0' 이면(반이 같으면), 번호(no)를 뺄셈해서 반환하기만 하면 된다.

```
    public int compare(Object o1, Object o2) {
        if(o1 instanceof Student && o2 instanceof Student) {
            Student s1 = (Student)o1;
            Student s2 = (Student)o2;

            int result = s1.ban - s2.ban;

            if(result==0) {  // 반이 같으면, 번호를 비교한다.
                return s1.no - s2.no;
            }

            return result;
        }

        return -1;
    }
```

위의 코드를 삼항연산자를 이용해서 작성하면 다음과 같다.

```
public int compare(Object o1, Object o2) {
    if(o1 instanceof Student && o2 instanceof Student) {
        Student s1 = (Student)o1;
        Student s2 = (Student)o2;

        return s1.ban==s2.ban ? s1.no - s2.no : s1.ban - s2.ban;
    }

    return -1;
}
```

[11-8] 문제11-7의 Student클래스에 총점(total)과 전교등수(schoolRank)를 저장하기 위한 인스턴스변수를 추가하였다. Student클래스의 기본정렬을 이름(name)이 아닌 총점(total)을 기준으로 한 내림차순으로 변경한 다음, 총점을 기준으로 각 학생의 전교등수를 계산하고 전교등수를 기준으로 오름차순 정렬하여 출력하시오.

[연습문제]/ch11/Exercise11_8.java

```
import java.util.*;

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

   int  total;      // 총점
   int  schoolRank; // 전교등수

   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;

        total = kor+eng+math;
   }

   int getTotal() {
        return total;
   }

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

   public int compareTo(Object o) {
        if(o instanceof Student) {
            Student tmp = (Student)o;

            return tmp.total - this.total;  // 총점 기준(내림차순)으로 정렬한다.
        } else {
            return -1;
        }
   }

   public String toString() {
        return name
             +","+ban
             +","+no
             +","+kor
             +","+eng
             +","+math
```

```
                +","+getTotal()
                +","+getAverage()
                +","+schoolRank // 새로추가
                ;
    }
} // class Student

class Exercise11_8 {
    public static void calculateSchoolRank(List list) {
        Collections.sort(list); // 먼저 list를 총점기준 내림차순으로 정렬한다.

        int prevRank = -1;      // 이전 전교등수
        int prevTotal = -1;     // 이전 총점
        int length = list.size();

//      1. 반복문을 이용해서 list에 저장된 Student객체를 하나씩 읽는다.
        for(int i=0;i < length; i++) {
            Student s = (Student)list.get(i);

//          1.1 총점(total)이 이전총점(prevTotal)과 같으면
//              이전 등수(prevRank)를 등수(schoolRank)로 한다.
            if(s.total==prevTotal) {
                s.schoolRank = prevRank;
            } else {
//          1.2 총점이 서로 다르면,
//              등수(schoolRank)의 값을 알맞게 계산해서 저장한다.
//              이전에 동점자였다면, 그 다음 등수는 동점자의 수를 고려해야한다.
                s.schoolRank = i + 1;
            }
//          1.3 현재 총점과 등수를 이전총점(prevTotal)과 이전등수(prevRank)에 저장한다.
            prevRank = s.schoolRank;
            prevTotal = s.total;
        } // for
    }

    public static void main(String[] args) {
        ArrayList list = new ArrayList();
        list.add(new Student("이자바",2,1,70,90,70));
        list.add(new Student("안자바",2,2,60,100,80));
        list.add(new Student("홍길동",1,3,100,100,100));
        list.add(new Student("남궁성",1,1,90,70,80));
        list.add(new Student("김자바",1,2,80,80,90));

        calculateSchoolRank(list);

        Iterator it = list.iterator();

        while(it.hasNext())
            System.out.println(it.next());
    }
}
```

[실행결과]

```
홍길동,1,3,100,100,100,300,100.0,1
김자바,1,2,80,80,90,250,83.3,2
안자바,2,2,60,100,80,240,80.0,3
```

```
남궁성,1,1,90,70,80,240,80.0,3
이자바,2,1,70,90,70,230,76.7,5
```

[해설] 등수를 계산하기 위해서는 먼저 총점 기준의 내림차순으로 정렬되어 있어야 한다. 그리고 바로 전 데이터와의 비교를 위해 이전 등수(prevRank), 이전 총점(prevTotal)을 저장하는 변수를 선언했다.

```
public static void calculateSchoolRank(List list) {
    Collections.sort(list); // 먼저 list를 총점기준 내림차순으로 정렬한다.

    int prevRank = -1;      // 이전 전교등수
    int prevTotal = -1;     // 이전 총점
    int length = list.size();
    ...
```

반복문을 이용해서 list에 저장된 학생정보를 하나씩 가져와 이전 총점(prevTotal)과 비교한다. 현재 데이터의 총점(s.total)과 이전 총점(prevTotal)이 같으면, 등수(s.schoolRank)를 이전 등수(prevRank)로 저장한다.

```
        if(s.total==prevTotal) {        // 총점이 이전 총점과 같으면,
            s.schoolRank = prevRank;    // 등수를 이전 등수와 같은 값으로 한다.
        } else {
            s.schoolRank = i + 1;
        }
```

다음은 실행과정에서 변수 값의 변화를 표로 나타낸 것이다. schoolRank가 i+1의 값으로 저장되지만, s.total과 prevTotal의 값이 같을 때는 i+1의 값이 아닌 prevRank의 값으로 저장되는 것을 알 수 있다.

s.total	prevTotal	prevRank	schoolRank	i+1
300	-1	-1	1	1
250	300	1	2	2
240	250	2	3	3
240	240	3	3	4
230	240	3	5	5

[11-9] 문제11-8의 Student클래스에 반등수(classRank)를 저장하기 위한 인스턴스변수를 추가하였다. 반등수를 계산하고 반과 반등수로 오름차순 정렬하여 결과를 출력하시오. (1)~(2)에 알맞은 코드를 넣어 완성하시오.

[연습문제]/ch11/Exercise11_9.java

```
import java.util.*;

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

   int   total;
   int   schoolRank; // 전교등수
   int   classRank;  // 반등수

   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;

       total = kor+eng+math;
   }

   int getTotal() {
       return total;
   }

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

   public int compareTo(Object o) {
       if(o instanceof Student) {
           Student tmp = (Student)o;

           return tmp.total - this.total;
       } else {
           return -1;
       }
   }

   public String toString() {
       return name
            +","+ban
            +","+no
            +","+kor
            +","+eng
            +","+math
```

```
            +","+getTotal()
            +","+getAverage()
            +","+schoolRank
            +","+classRank   // 새로추가
            ;
    }
} // class Student

class ClassTotalComparator implements Comparator {
    public int compare(Object o1, Object o2) {
        Student s1 = (Student)o1;
        Student s2 = (Student)o2;

        int result = s1.ban - s2.ban; // 반(ban) 기준 정렬(오름차순)

        if(result==0)
            result = s2.total - s1.total;    // 총점(total) 기준 정렬(내림차순)

        return result;
    }
}

class Exercise11_9 {
    public static void calculateClassRank(List list) {
        // 먼저 반별 총점기준 내림차순으로 정렬한다.
        Collections.sort(list, new ClassTotalComparator());

        int prevBan = -1;
        int prevRank = -1;
        int prevTotal = -1;
        int length = list.size();

        for(int i=0, n=0; i < length; i++, n++) {
//      1. 반복문을 이용해서 list에 저장된 Student객체를 하나씩 읽는다.
            Student s = (Student)list.get(i);

//        1.1 반이 달라지면,(ban와 prevBan이 다르면)
//            이전 등수(prevRank)와 이전 총점(prevTotal)을 초기화 한다.
            if(s.ban!=prevBan) {
                prevRank = -1;
                prevTotal = -1;
                n = 0;
            }

//        1.2 총점(total)이 이전총점(prevTotal)과 같으면
//          이전 등수(prevRank)를 등수(classRank)로 한다.
            if(s.total==prevTotal) {
                s.classRank = prevRank;
            } else {
//        1.3 총점이 서로 다르면,
//          등수(classRank)의 값을 알맞게 계산해서 저장한다.
//          이전에 동점자였다면, 그 다음 등수는 동점자의 수를 고려해야한다.
                s.classRank = n + 1;
            }

//        1.4 현재 반과 총점과 등수를 이전 반(prevBan),
//          이전 총점(prevTotal), 이전 등수(prevRank)에 저장한다.
```

```
            prevBan = s.ban;
            prevRank = s.classRank;
            prevTotal = s.total;
        }
    } // public static void calculateClassRank(List list) {

    public static void calculateSchoolRank(List list) {
        Collections.sort(list); // 먼저 list를 총점기준 내림차순으로 정렬한다.

        int prevRank = -1;      // 이전 전교등수
        int prevTotal = -1;      // 이전 총점
        int length = list.size();

        for(int i=0;i < length; i++) {
            Student s = (Student)list.get(i);

            if(s.total==prevTotal) {
                s.schoolRank = prevRank;
            } else {
                s.schoolRank = i + 1;
            }

            prevRank = s.schoolRank;
            prevTotal = s.total;
        } // for
    }

    public static void main(String[] args) {
        ArrayList list = new ArrayList();
        list.add(new Student("이자바",2,1,70,90,70));
        list.add(new Student("안자바",2,2,60,100,80));
        list.add(new Student("홍길동",1,3,100,100,100));
        list.add(new Student("남궁성",1,1,90,70,80));
        list.add(new Student("김자바",1,2,80,80,90));

        calculateSchoolRank(list);
        calculateClassRank(list);

        Iterator it = list.iterator();

        while(it.hasNext())
            System.out.println(it.next());
    }
}
```

[실행결과]

```
홍길동,1,3,100,100,100,300,100.0,1,1
김자바,1,2,80,80,90,250,83.3,2,2
남궁성,1,1,90,70,80,240,80.0,3,3
안자바,2,2,60,100,80,240,80.0,3,1
이자바,2,1,70,90,70,230,76.7,5,2
```

[해설] 문제11-8을 보다 발전시킨 예제이다. 문제11-8과 달리 반별 오름차순과 총점별 내림차순, 2가지 기준으로 정렬한 다음, 반이 달라질 때마다 이전 총점과 이전 등수, 그리고 등수(n)를 초기화 해준다는 것만 다르다.

등수를 계산하기 위해서는 먼저 반별 오름차순과 총점별 내림차순, 2가지 기준으로 정렬한다. 그래서 ClassTotalComparator는 반을 기준으로 비교하고, 반이 같으면 총점을 비교해서 정렬하도록 정의하고, 반별 총점을 계산하는 calculateClassRank메서드 내의 sort()에 ClassTotalComparator인스턴스를 정렬기준으로 제공한다.

```
class ClassTotalComparator implements Comparator {
   public int compare(Object o1, Object o2) {
        Student s1 = (Student)o1;
        Student s2 = (Student)o2;

        int result = s1.ban - s2.ban; // 반(ban) 기준 정렬(오름차순)

        if(result==0) // 반이 같으면, 총점을 비교한다.
            result = s2.total - s1.total;   // 총점(total) 기준 정렬(내림차순)

        return result;
//      return {s1.ban==s2.ban}? s1.ban-s2.ban : s2.total-s1.total;
   }
}

class Exercise11_9 {
   public static void calculateClassRank(List list) {
        // 먼저 반별 총점기준 내림차순으로 정렬한다.
        Collections.sort(list, new ClassTotalComparator());
```

그 다음에는 문제11-8에서 와 같이 반복문을 이용해서 정렬된 데이터를 한 줄 씩 읽어서 등수를 계산한다. 문제11-8과 다른 점은, 반이 달라지면 다시 1등부터 시작해서 등수를 매겨야하므로 이전 총점(prevTotal), 이전 등수(prevRank), 등수(n)을 초기화한다는 것이다.

```
        for(int i=0, n=0; i < length; i++, n++) {
            Student s = (Student)list.get(i);

//          1.1 반이 달라지면,(ban와 prevBan이 다르면)
//              이전 등수(prevRank)와 이전 총점(prevTotal)을 초기화 한다.
            if(s.ban!=prevBan) {
                prevRank = -1;
                prevTotal = -1;
                n = 0;
            }

            if(s.total==prevTotal) {
                s.classRank = prevRank;
            } else {
                s.classRank = n + 1;
            }

            prevBan = s.ban;
            prevRank = s.classRank;
            prevTotal = s.total;
        }
```

다음은 실행과정에서 변수 값의 변화를 표로 나타낸 것이다. 전과는 달리 등수를 매길 때 반복문의 카운터 i대신 n을 사용하였으며, 현재 반(s.ban)과 이전 반(prevBan)의 값이 다르면, 이전 총점(prevTotal), 이전 등수(prevRank), 등수(n)을 초기화한다는 것을 확인하자.

s.total	s.ban	prevBan	prevTotal	prevRank	classRank	i	n	n+1
300	1	-1	-1	-1	1	0	0	1
250	1	1	300	1	2	1	1	2
240	1	1	250	2	3	2	2	3
240	2	1	-1	-1	1	3	0	1
230	2	2	240	1	2	4	1	2

[11-10] 다음 예제의 빙고판은 1~30사이의 숫자들로 만든 것인데, 숫자들의 위치가 잘 섞이지 않는다는 문제가 있다. 이러한 문제가 발생하는 이유와 이 문제를 개선하기 위한 방법을 설명하고, 이를 개선한 새로운 코드를 작성하시오.

[연습문제]/ch11/Exercise11_10.java

```
import java.util.*;

class Exercise11_10 {
   public static void main(String[] args) {
        Set set = new HashSet();
        int[][] board = new int[5][5];

        for(int i=0; set.size() < 25; i++) {
            set.add((int)(Math.random()*30)+1+"");
        }

        Iterator it = set.iterator();

        for(int i=0; i < board.length; i++) {
            for(int j=0; j < board[i].length; j++) {
                board[i][j] = Integer.parseInt((String)it.next());
                System.out.print((board[i][j] < 10 ? "  " : " ")
                                                + board[i][j]);
            }
            System.out.println();
        }
   } // main
}
```

[정답] HashSet은 중복을 허용하지 않고 순서를 유지하지 않기 때문에 발생하는 문제이다. 아무리 임의의 순서로 저장을 해도, 해싱(hashing) 알고리즘의 특성상 한 숫자가 고정된 위치에 저장되기 때문이다.

이 문제를 해결하기 위해서는 저장순서를 유지하는 List인터페이스를 구현한 컬렉션 클래스를 사용하도록 변경해야 한다.

[연습문제]/ch11/Exercise11_10_2.java

```
import java.util.*;

class Exercise11_10_2
{
   public static void main(String[] args)
   {
        Set set = new HashSet();
        int[][] board = new int[5][5];

        for(int i=0; set.size() < 25; i++) {
            set.add((int)(Math.random()*30)+1+"");
        }

        ArrayList list = new ArrayList(set);
        Collections.shuffle(list);
```

```
        Iterator it = list.iterator();

        for(int i=0; i < board.length; i++) {
            for(int j=0; j < board[i].length; j++) {
                board[i][j] = Integer.parseInt((String)it.next());
                System.out.print((board[i][j] < 10 ? "  " : " ") + board[i][j]);
            }
            System.out.println();
        }
    } // main
}
```

[해설] 중복된 값을 허용하지 않는 다는 특성을 이용해서 HashSet에 서로 다른 임의의 값을 저장하는 것까지는 좋았는데, 해싱알고리즘의 특성상 같은 값은 같은 자리에 저장되기 때문에 빙고판의 숫자들이 잘 섞이지 않는다는 문제가 발생하였다.

```
Set set = new HashSet();
int[][] board = new int[5][5];

for(int i=0; set.size() < 25; i++) {
    set.add((int)(Math.random()*30)+1+"");
}
```

그래서 저장순서를 유지하는 ArrayList에 HashSet의 데이터를 옮겨 담은 다음, Collections.shuffle()을 이용해서 저장된 데이터들의 순서를 뒤섞었다.

```
ArrayList list = new ArrayList(set); // set과 같은 데이터를 가진 ArrayList를 생성
Collections.shuffle(list);    // list에 저장된 데이터의 순서를 섞는다.

Iterator it = list.iterator();
```

이처럼 대부분의 컬렉션 클래스들은 다른 컬렉션으로 데이터를 쉽게 옮길 수 있게 설계되어 있다. 매개변수의 타입이 Collection인터페이스이므로 Collection인터페이스의 자손인 List인터페이스와 Set인터페이스를 구현한 모든 클래스의 인스턴스가 매개변수로 가능하다.

```
ArrayList(Collection<? extends E> c)
LinkedList(Collection<? extends E> c)
Vector(Collection<? extends E> c)
HashSet(Collection<? extends E> c)
TreeSet(Collection<? extends E> c)
LinkedHashSet(Collection<? extends E> c)
```

[11-11] 다음은 SutdaCard클래스를 HashSet에 저장하고 출력하는 예제이다. HashSet에 중복된 카드가 저장되지 않도록 SutdaCard의 hashCode()를 알맞게 오버라이딩하시오.

[Hint] String클래스의 hashCode()를 사용하라.

[연습문제]/ch11/Exercise11_11.java

```
import java.util.*;

class SutdaCard {
   int num;
   boolean isKwang;

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

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

   public boolean equals(Object obj) {
       if(obj instanceof SutdaCard) {
           SutdaCard c = (SutdaCard)obj;
           return num==c.num && isKwang==c.isKwang;
       } else {
           return false;
       }
   }

   public String toString() {
       return num + ( isKwang ? "K":"");
   }

   public int hashCode() {
       return toString().hashCode();  // String클래스의 hashCode()를 호출한다.
   }
}

class Exercise11_11 {
   public static void main(String[] args) {
       SutdaCard c1 = new SutdaCard(3,true);
       SutdaCard c2 = new SutdaCard(3,true);
       SutdaCard c3 = new SutdaCard(1,true);

       HashSet set = new HashSet();
       set.add(c1);
       set.add(c2);
       set.add(c3);

       System.out.println(set);
   }
}
```

[실행결과]

```
[3K, 1K]
```

[해설] hashCode()의 기본 구현은 클래스이름과 메모리주소와 관련된 정수값으로 이루어져 있기 때문에, 두 객체의 hashCode()값은 절대로 같을 수가 없다.(서로 다른 두 객체가 같은 메모리 번지에 존재할 수 없기 때문에)

대부분의 경우 서로 다른 객체라도 클래스의 인스턴스변수 값이 같으면, 예를 들어 SutdaCar의 경우 num과 isKwang의 값이 같으면 같은 객체로 인식해야한다. 즉, equals()의 결과가 true이어야하고, 두 객체의 해시코드(hashCode()를 호출한 결과)가 같아야 한다. 그래서 equals()와 hashCode()를 적절히 오버라이딩 해줘야 한다.

때로는 equals()만 오버라이딩해줘도 되지만, 해시알고리즘을 사용하는 HashSet에 담을 때는 반드시 hashCode()도 오버라이딩해줘야 한다.

이 문제의 실행결과를 보면 중복을 허용하지 않는 HashSet을 사용하고도 [1K, 3K, 3K]와 같은 결과를 얻는다. 그 이유는 hashCode()를 오버라이딩 하지 않았기 때문이다.

그러나 hashCode()를 오버라이딩한 후에는 [3K, 1K]와 같이 중복이 제거된 결과를 얻을 수 있다.

hashCode()를 오버라이딩하라고 하면 어떻게 해야 할지 막막할 것이다. 그러나 걱정하지 말자. 이미 다 구현되어 있으니 그냥 가져다 쓰기만 하면 된다.

String클래스의 hashCode()는 객체의 주소가 아닌 문자열의 내용을 기반으로 해시코드를 생성하므로 문자열의 내용이 같으면 항상 같은 값의 해시코드를 반환한다.

SutdaCard의 toString()이 num과 isKwang의 값으로 문자열을 만들어 반환하기 때문에, toString()을 호출한 결과에 hashCode()를 호출함으로써 SutdaCard의 hashCode()를 간단히 구현할 수 있었다.

```
public String toString() {
    return num + ( isKwang ? "K":"");
}

public int hashCode() {
    return toString().hashCode(); // String클래스의 hashCode()를 호출한다.

}
```

인스턴스변수들의 값을 결합한 문자열을 만들고, 그 문자열에 대해 hashCode()를 호출하는 방법은 쉬우면서도 효과적이다.

[11-12] 다음은 섯다게임에서 카드의 순위를 결정하는 등급목록(족보)이다. HashMap에 등급과 점수를 저장하는 registerJokbo()와 게임참가자의 점수를 계산해서 반환하는 getPoint()를 완성하시오.

[참고] 섯다게임은 두 장의 카드의 숫자를 더한 값을 10으로 나눈 나머지가 높은 쪽이 이기는 게임이다. 그 외에도 특정 숫자로 구성된 카드로 이루어진 등급(족보)이 있어서 높은 등급의 카드가 이긴다.

카드1	카드2	점수
K	K	4000
10	10	3100
9	9	3090
8	8	3080
7	7	3070
6	6	3060
5	5	3050
4	4	3040
3	3	3030
2	2	3020
1	1	3010
-	-	-

카드1	카드2	점수
1	2	2060
2	1	2060
1	4	2050
4	1	2050
1	9	2040
9	1	2040
1	10	2030
10	1	2030
4	10	2020
10	4	2020
4	6	2010
6	4	2010

[연습문제]/ch11/Exercise11_12.java

```
import java.util.*;

class Exercise11_12 {
   public static void main(String args[]) throws Exception {
        SutdaDeck deck = new SutdaDeck();

        deck.shuffle();
        Player p1 = new Player("타짜", deck.pick(), deck.pick());
        Player p2 = new Player("고수", deck.pick(), deck.pick());

        System.out.println(p1+" "+deck.getPoint(p1));
        System.out.println(p2+" "+deck.getPoint(p2));
   }
}

class SutdaDeck {
   final int CARD_NUM = 20;
   SutdaCard[] cards = new SutdaCard[CARD_NUM];

   int pos = 0; // 다음에 가져올 카드의 위치
   HashMap jokbo = new HashMap(); // 족보를 저장할 HashMap

   SutdaDeck() {
        for(int i=0;i < cards.length;i++) {
            int num = i%10 + 1;
            boolean isKwang = i < 10 && (num==1 || num==3 || num==8);

            cards[i] = new SutdaCard(num,isKwang);
        }
        registerJokbo();  // 족보를 등록한다.
   }
```

```
    void registerJokbo() {
        jokbo.put("KK",  4000);

        jokbo.put("1010",3100);
        jokbo.put("99",  3090);
        jokbo.put("88",  3080);
        jokbo.put("77",  3070);
        jokbo.put("66",  3060);
        jokbo.put("55",  3050);
        jokbo.put("44",  3040);
        jokbo.put("33",  3030);
        jokbo.put("22",  3020);
        jokbo.put("11",  3010);

        jokbo.put("12",  2060);
        jokbo.put("21",  2060);
        jokbo.put("14",  2050);
        jokbo.put("41",  2050);
        jokbo.put("19",  2040);
        jokbo.put("91",  2040);
        jokbo.put("110", 2030);
        jokbo.put("101", 2030);
        jokbo.put("104", 2020);
        jokbo.put("410", 2020);
        jokbo.put("46",  2010);
        jokbo.put("64",  2010);
    }

    int getPoint(Player p) {
        if(p==null) return 0;

        SutdaCard c1 = p.c1;
        SutdaCard c2 = p.c2;

        Integer result = 0; // Integer result = new Integer(0);

//      1. 카드 두 장이 모두 광이면, jokbo에서 키를 "KK"로 해서 점수를 조회한다.
        if(c1.isKwang && c2.isKwang) {
            result = (Integer)jokbo.get("KK");
        } else {
//      2. 두 카드의 숫자(num)로 jokbo에서 등급을 조회한다.
            result = (Integer)jokbo.get(""+c1.num+c2.num);

//      3. 해당하는 등급이 없으면, 아래의 공식으로 점수를 계산한다.
//        (c1.num + c2.num) % 10 + 1000
            if(result==null) {
                result = new Integer((c1.num + c2.num) % 10 + 1000);
            }
        }

//      4. Player의 점수(point)에 계산한 값을 저장한다.
        p.point = result.intValue();

        return result.intValue();
    }
```

```
    SutdaCard pick() throws Exception {
        SutdaCard c = null;

        if(0 <= pos && pos < CARD_NUM) {
            c = cards[pos];
            cards[pos++] = null;
        } else {
            throw new Exception("남아있는 카드가 없습니다.");
        }

        return c;
    }

    void shuffle() {
        for(int x=0; x < CARD_NUM * 2; x++) {
            int i = (int)(Math.random() * CARD_NUM);
            int j = (int)(Math.random() * CARD_NUM);

            SutdaCard tmp = cards[i];
            cards[i] = cards[j];
            cards[j] = tmp;
        }
    }
} // SutdaDeck

class Player {
    String name;
    SutdaCard c1;
    SutdaCard c2;

    int point; // 카드의 등급에 따른 점수 - 새로 추가

    Player(String name, SutdaCard c1, SutdaCard c2) {
        this.name = name ;
        this.c1 = c1 ;
        this.c2 = c2 ;
    }

    public String toString() {
        return "["+name+"]"+ c1.toString() +","+ c2.toString();
    }
} // class Player

class SutdaCard {
    int num;
    boolean isKwang;

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

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

    public String toString() {
```

```
        return num + ( isKwang ? "K":"");
    }
}
```

[실행결과]

```
[타짜]5,9 1004
[고수]1,1K 3010
```

[해설] registerJokbo()는 단순히 put()을 호출해서 HashMap에 값을 저장하는 것이므로 설명하지 않아도 이해하는데 어려움이 없을 것 같다. 굳이 설명한다면, put(Object key, Object value)에 대한 것인데, 이 메서드의 매개변수 타입을 보면 Object라서 오른쪽의 코드와 같이 wrapper클래스를 써야하는 것이 원칙이다.

그러나 jdk1.5에서 부터 오토박싱기능이 추가되어 컴파일러가 기본형(primitive type)을 자동적으로 wrapper클래스로 변환해준다. 그래서 왼쪽과 같은 코드가 가능한 것이다.

왼쪽과 같이 코드를 작성해도 컴파일러가 오른쪽과 같은 코드로 자동 변환해주는 것이다.

```
jokbo.put("KK", 4000);   ◀──▶   jokbo.put("KK", new Integer(4000));
```

getPoint(Player p)는 게임참가자(Player)의 카드를 평가해서 점수를 반환하는 메서드인데, 게임참가자가 가진 카드의 숫자를 족보(jokbo)에서 찾아서 해당 점수를 가져온다.

카드의 숫자를 문자열로 만들어서 get(Object key)를 호출하면, 족보에 해당하는지 알 수 있다. get메서드의 호출결과가 null이면 족보에 없다는 것을 의미한다. 족보에 없는 숫자조합이라면, 두 숫자를 더한 다음 10으로 나눈 나머지에 1000을 더해서 점수를 계산한다.

```
//      1. 카드 두 장이 모두 광이면, jokbo에서 키를 "KK"로 해서 점수를 조회한다.
        if(c1.isKwang && c2.isKwang) {
            result = (Integer)jokbo.get("KK");
        } else {
//      2. 두 카드의 숫자(num)로 jokbo에서 등급을 조회한다.
            result = (Integer)jokbo.get(""+c1.num+c2.num);

//      3. 해당하는 등급이 없으면, 아래의 공식으로 점수를 계산한다.
//        (c1.num + c2.num) % 10 + 1000
            if(result==null) {
                result = new Integer((c1.num + c2.num) % 10 + 1000);
            }
        }

//      4. Player의 점수(point)에 계산한 값을 저장한다.
        p.point = result.intValue();  // Integer를 int로 변환해서 반환한다.
```

[11-13] 다음 코드는 문제11-12를 발전시킨 것으로 각 Player들의 점수를 계산하고, 점수가 제일 높은 사람을 출력하는 코드이다. TreeMap의 정렬기준을 점수가 제일 높은 사람부터 내림차순이 되도록 아래의 코드를 완성하시오. 단, 동점자 처리는 하지 않는다.

[연습문제]/ch11/Exercise11_13.java

```
import java.util.*;

class Exercise11_13 {
   public static void main(String args[]) throws Exception {
        SutdaDeck deck = new SutdaDeck();

        deck.shuffle();

        Player[] pArr = {
            new Player("타짜", deck.pick(), deck.pick()),
            new Player("고수", deck.pick(), deck.pick()),
            new Player("물주", deck.pick(), deck.pick()),
            new Player("중수", deck.pick(), deck.pick()),
            new Player("하수", deck.pick(), deck.pick())
        };

        TreeMap rank = new TreeMap(new Comparator(){
            public int compare(Object o1, Object o2) {
                if(o1 instanceof Player && o2 instanceof Player) {
                    Player p1 = (Player)o1;
                    Player p2 = (Player)o2;

                    return p2.point - p1.point; // 점수기준의 내림차순으로 정렬
                }

                return -1;
            }
        });

        for(int i=0; i < pArr.length;i++) {
            Player p = pArr[i];
            rank.put(p, deck.getPoint(p));
            System.out.println(p+" "+deck.getPoint(p));
        }

        System.out.println();
        System.out.println("1위는 "+rank.firstKey()+"입니다.");
   }
}

class SutdaDeck {
   final int CARD_NUM = 20;
   SutdaCard[] cards = new SutdaCard[CARD_NUM];

   int pos = 0; // 다음에 가져올 카드의 위치
   HashMap jokbo = new HashMap(); // 족보를 저장할 HashMap

   SutdaDeck() {
        for(int i=0;i < cards.length;i++) {
            int num = i%10 + 1;
            boolean isKwang = i < 10 && (num==1 || num==3 || num==8);
```

```
            cards[i] = new SutdaCard(num,isKwang);
        }

        registerJokbo();  // 족보를 등록한다.
    }

    void registerJokbo() {
        jokbo.put("KK",  4000);
        jokbo.put("1010",3100);  jokbo.put("12",  2060);
        jokbo.put("99",  3090);  jokbo.put("21",  2060);
        jokbo.put("88",  3080);  jokbo.put("14",  2050);
        jokbo.put("77",  3070);  jokbo.put("41",  2050);
        jokbo.put("66",  3060);  jokbo.put("19",  2040);
        jokbo.put("55",  3050);  jokbo.put("91",  2040);
        jokbo.put("44",  3040);  jokbo.put("110", 2030);
        jokbo.put("33",  3030);  jokbo.put("101", 2030);
        jokbo.put("22",  3020);  jokbo.put("104", 2020);
        jokbo.put("11",  3010);  jokbo.put("410", 2020);
                                 jokbo.put("46",  2010);
                                 jokbo.put("64",  2010);
    }

    int getPoint(Player p) {
        if(p==null) return 0;

        SutdaCard c1 = p.c1;
        SutdaCard c2 = p.c2;

        Integer result = 0;

        if(c1.isKwang && c2.isKwang) {
            result = (Integer)jokbo.get("KK");
        } else {
            result = (Integer)jokbo.get(""+c1.num+c2.num);

            if(result==null) {
                result = new Integer((c1.num + c2.num) % 10 + 1000);
            }
        }

        p.point = result.intValue();

        return result.intValue();
    }

    SutdaCard pick() throws Exception {
        SutdaCard c = null;

        if(0 <= pos && pos < CARD_NUM) {
            c = cards[pos];
            cards[pos++] = null;
        } else {
            throw new Exception("남아있는 카드가 없습니다.");
        }

        return c;
    }
```

```
    void shuffle() {
        for(int x=0; x < CARD_NUM * 2; x++) {
            int i = (int)(Math.random() * CARD_NUM);
            int j = (int)(Math.random() * CARD_NUM);

            SutdaCard tmp = cards[i];
            cards[i] = cards[j];
            cards[j] = tmp;
        }
    }
} // SutdaDeck

class Player {
    String name;
    SutdaCard c1;
    SutdaCard c2;

    int point;

    Player(String name, SutdaCard c1, SutdaCard c2) {
        this.name = name ;
        this.c1 = c1 ;
        this.c2 = c2 ;
    }

    public String toString() {
        return "["+name+"]"+ c1.toString() +","+ c2.toString();
    }
} // class Player

class SutdaCard {
    int num;
    boolean isKwang;

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

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

    public String toString() {
        return num + ( isKwang ? "K":"");
    }
}
```

[실행결과]

```
[타짜]7,2 1009
[고수]2,5 1007
[물주]1,7 1008
[중수]10,4 2020
[하수]9,6 1005

1위는 [중수]10,4입니다.
```

[11-14] 다음은 성적처리 프로그램의 일부이다. Scanner클래스를 이용해서 화면으로부터 데이터를 입력하고 보여주는 기능을 완성하시오.

[연습문제]/ch11/Exercise11_14.java

```
import java.io.*;
import java.util.*;

class Exercise11_14
{
  static ArrayList record = new ArrayList(); // 성적데이터를 저장할 공간
  static Scanner s = new Scanner(System.in);

  public static void main(String args[]) {
    while(true) {
      switch(displayMenu()) {
        case 1 :
          inputRecord();
          break;
        case 2 :
          displayRecord();
          break;
        case 3 :
          System.out.println("프로그램을 종료합니다.");
          System.exit(0);
      }
    } // while(true)
  }

  // menu를 보여주는 메서드
  static int displayMenu(){
System.out.println("************************************************");
    System.out.println("*           성적 관리 프로그램          *");
System.out.println("************************************************");
    System.out.println();
    System.out.println(" 1. 학생성적 입력하기 ");
    System.out.println();
    System.out.println(" 2. 학생성적 보기");
    System.out.println();
    System.out.println(" 3. 프로그램 종료 ");
    System.out.println();
    System.out.print("원하는 메뉴를 선택하세요.(1~3) : ");

    int menu = 0;

    do {
      try {
        menu = Integer.parseInt(s.nextLine().trim());

        if(1 <= menu && menu <= 3) {
          break;
        } else {
          throw new Exception();
        }
      } catch(Exception e) {
        System.out.println("메뉴를 잘못 선택하셨습니다. 다시 입력해주세요.");
```

```
            System.out.print("원하는 메뉴를 선택하세요.(1~3) : ");
        }
      } while(true);

      return menu;
    } // public static int displayMenu(){

    // 데이터를 입력받는 메서드
    static void inputRecord() {
     System.out.println("1. 학생성적 입력하기");
     System.out.println("이름,반,번호,국어성적,영어성적,수학성적'의 순서로 공백없이 입력하세요.");
     System.out.println("입력을 마치려면 q를 입력하세요. 메인화면으로 돌아갑니다.");

      while(true) {
        System.out.print(">>");

          try {
              String input = s.nextLine().trim();

              if(!input.equalsIgnoreCase("q")) {
                  // Scanner를 이용해서 화면으로 부터 데이터를 입력받는다.(','를 구분자로)
                  Scanner s2 = new Scanner(input).useDelimiter(",");
                  // 입력받은 값으로 Student인스턴스를 생성하고 record에 추가한다.
                  record.add(new Student(s2.next(), s2.nextInt(), s2.nextInt(),
                                         s2.nextInt(), s2.nextInt(), s2.nextInt()));
                  System.out.println("잘입력되었습니다. 입력을 마치려면 q를 입력하세요.");
              } else {
                  // 입력받은 값이 q 또는 Q이면 메서드를 종료한다.
                  return;
              }
          } catch(Exception e) {
              // 입력받은 데이터에서 예외가 발생하면, "입력오류입니다."를 보여주고 다시 입력받는다.
              System.out.println("입력오류입니다. 이름, 반, 번호, 국어성적, 영어성적, 수학성적'의
순서로 입력하세요.");
          }
      } // end of while
    } // public static void inputRecord() {

    // 데이터 목록을 보여주는 메서드
    static void displayRecord() {
         int koreanTotal = 0;
         int englishTotal = 0;
         int mathTotal = 0;
         int total = 0;

         int length = record.size();

         if(length > 0) {
              System.out.println();
              System.out.println("이름 반 번호 국어 영어 수학 총점 평균 전교등수 반등수");

  System.out.println("=================================================");

              for (int i = 0; i < length ; i++) {
                  Student student = (Student)record.get(i);
                  System.out.println(student);
                  koreanTotal += student.kor;
```

```
                mathTotal += student.math;
                englishTotal += student.eng;
                total += student.total;
            }

System.out.println("================================================");
            System.out.println("총점: "+koreanTotal+" "+englishTotal
                                        +" "+mathTotal+" "+total);
            System.out.println();
        } else {
System.out.println("================================================");
            System.out.println(" 데이터가 없습니다.");
System.out.println("================================================");
        }
    } // static void displayRecord() {
}


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

    int  total;
    int   schoolRank;
    int   classRank; // 반등수

    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;

        total = kor+eng+math;
    }

    int getTotal() {
        return total;
    }

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

    public int compareTo(Object o) {
        if(o instanceof Student) {
            Student tmp = (Student)o;

            return tmp.total - this.total;
        } else {
            return -1;
        }
```

```
    }

    public String toString() {
        return name
             +","+ban
             +","+no
             +","+kor
             +","+eng
             +","+math
             +","+getTotal()
             +","+getAverage()
             +","+schoolRank
             +","+classRank
             ;
    }
  } // class Student
```

[실행결과]

```
**************************************************
*                  성적 관리 프로그램                    *
**************************************************

 1. 학생성적 입력하기

 2. 학생성적 보기

 3. 프로그램 종료

원하는 메뉴를 선택하세요.(1~3) : 5
메뉴를 잘못 선택하셨습니다. 다시 입력해주세요.
원하는 메뉴를 선택하세요.(1~3) : 2
============================================================
 데이터가 없습니다.
============================================================
**************************************************
*                  성적 관리 프로그램                    *
**************************************************

 1. 학생성적 입력하기

 2. 학생성적 보기

 3. 프로그램 종료

원하는 메뉴를 선택하세요.(1~3) : 1
1. 학생성적 입력하기
이름,반,번호,국어성적,영어성적,수학성적'의 순서로 공백없이 입력하세요.
입력을 마치려면 q를 입력하세요. 메인화면으로 돌아갑니다.
>>
입력오류입니다. 이름, 반, 번호, 국어성적, 영어성적, 수학성적'의 순서로 입력하세
요.
>>자바짱,1,1,100,100,100
잘입력되었습니다. 입력을 마치려면 q를 입력하세요.
>>김자바,1,2,80,80,80
잘입력되었습니다. 입력을 마치려면 q를 입력하세요.
```

```
>>q
*************************************************
*                  성적 관리 프로그램                    *
*************************************************

 1. 학생성적 입력하기

 2. 학생성적 보기

 3. 프로그램 종료

원하는 메뉴를 선택하세요.(1~3) : 2

이름 반 번호 국어 영어 수학 총점 평균 전교등수 반등수
=======================================================
자바짱,1,1,100,100,100,300,100.0,0,0
김자바,1,2,80,80,80,240,80.0,0,0
=======================================================
총점: 180 180 180 540

*************************************************
*                  성적 관리 프로그램                    *
*************************************************

 1. 학생성적 입력하기

 2. 학생성적 보기

 3. 프로그램 종료

원하는 메뉴를 선택하세요.(1~3) : 3
프로그램을 종료합니다.
```

[해설] 다음은 사용자로부터 메뉴를 입력받아 처리하는 부분이다. Scanner의 nextLine()을 이용해서 라인단위로 입력을 받고, 공백을 제거하기 위해 trim()을 사용했다.

do-while문을 사용해서 사용자가 유효한 값(1~3범위의 값)을 입력할 때까지 반복해서 입력을 받도록 하였다.

```
int menu = 0;

 do {
   try {
     menu = Integer.parseInt(s.nextLine().trim());

     if(1 <= menu && menu <= 3) {
        break; // 유효한 메뉴값을 입력하면 do-while문을 빠져나간다.
     } else {
        throw new Exception(); // 메뉴의 범위를 벗어나면 예외를 발생시킨다.
     }
   } catch(Exception e) {
     System.out.println("메뉴를 잘못 선택하셨습니다. 다시 입력해주세요.");
     System.out.print("원하는 메뉴를 선택하세요.(1~3) : ");
   }
 } while(true);

 return menu;
```

다음은 성적을 입력하는 부분이다. Scanner를 이용해서 라인단위로 입력받은 다음, Scanner를 한 번 더 사용해서 라인을 ','을 구분자로 다시 나눈다. 그리고 이 값을 이용해서 Student인스턴스를 생성해서 성적데이터를 저장하는 공간인 record에 추가한다. 무한반복문(while)에 둘러쌓여있기 때문에, 'q'나 'Q'를 입력할 때까지 반복해서 성적을 입력할 수 있다.

```
  while(true) {
     System.out.print(">>");

     try {
          String input = s.nextLine().trim();

          if(!input.equalsIgnoreCase("q")) {
               // Scanner를 이용해서 화면으로 부터 데이터를 입력받는다.(','를 구분자로)
               Scanner s2 = new Scanner(input).useDelimiter(",");
               // 입력받은 값으로 Student인스턴스를 생성하고 record에 추가한다.
               record.add(new Student(s2.next(), s2.nextInt(), s2.nextInt(),
                                      s2.nextInt(), s2.nextInt(), s2.nextInt()));
               System.out.println("잘입력되었습니다. 입력을 마치려면 q를 입력하세요.");
          } else {
               // 입력받은 값이 q 또는 Q이면 메서드를 종료한다.
               return;
          }
     } catch(Exception e) {
          // 입력받은 데이터에서 예외가 발생하면, "입력오류입니다."를 보여주고 다시 입력받는다.
          System.out.println("입력오류입니다. 이름, 반, 번호, 국어성적, 영어성적, 수학성적'의
순서로 입력하세요.");
     }
  } // end of while
```

[연습문제 - 모범답안]

[12-1] 클래스 Box가 다음과 같이 정의되어 있을 때, 다음 중 오류가 발생하는 문장은? 경고가 발생하는 문장은?

```
class Box<T> { // 지네릭 타입 T를 선언
    T item;

    void setItem(T item) { this.item = item; }
    T getItem() { return item; }
}
```

a. Box<Object> b = new Box<String>();
b. Box<Object> b = (Object)new Box<String>();
c. new Box<String>().setItem(new Object());
d. new Box<String>().setItem("ABC");

[정답] a, b, c

[해설]

a. Box<Object> b = new Box<String>(); // 에러. 대입된 타입이 반드시 같아야 한다.
b. Box<Object> b = (Object)new Box<String>(); // 에러. Object타입을 Box<Object>타입의 참조변수에 저장불가.(타입 불일치)
c. new Box<String>().setItem(new Object()); // 에러. 대입된 타입이 String이므로, setItem(T item)의 매개변수 역시, String타입만 허용된다.
d. new Box<String>().setItem("ABC"); // OK. 대입된 타입인 String과 일치하는 타입을 매개변수로 지정했기 때문에 OK.

[12-2] 지네릭 메서드 makeJuice()가 아래와 같이 정의되어 있을 때, 이 메서드를 올바르게 호출한 문장을 모두 고르시오. (Apple과 Grape는 Fruit의 자손이라고 가정하자.)

```
class Juicer {
    static <T extends Fruit> String makeJuice(FruitBox<T> box) {
        String tmp = "";
        for(Fruit f : box.getList()) tmp += f + " ";
        return tmp;
    }
}
```

a. Juicer.<Apple>makeJuice(new FruitBox<Fruit>());
b. Juicer.<Fruit>makeJuice(new FruitBox<Grape>());
c. Juicer.<Fruit>makeJuice(new FruitBox<Fruit>());
d. Juicer.makeJuice(new FruitBox<Apple>());
e. Juicer.makeJuice(new FruitBox<Object>());

[정답] c, d

[해설]

a. Juicer.**<Apple>**makeJuice(new FruitBox<**Fruit**>()); // 에러. 지네릭 메서드에 대입된 타입이 Apple이므로, 이 메서드의 매개변수는 'FruitBox<Apple>'타입이 된다. new FruitBox<Fruit>()는 매개변수의 타입과 일치하지 않으며, 자동형변환도 불가능한 타입이므로 에러이다.

b. Juicer.<**Fruit**>makeJuice(new FruitBox<**Grape**>()); // 에러. Grape가 Fruit의 자손이라고 해도, 타입이 다르기 때문에 같은 이유로 에러.

c. Juicer.<**Fruit**>makeJuice(new FruitBox<**Fruit**>()); // OK.

d. Juicer.makeJuice(new FruitBox<**Apple**>()); // OK. 지네릭 메서드의 타입 호출이 생략된 형태. 생략하지 않았다면, 'Juicer.**<Apple>**makeJuice(new FruitBox<**Apple**>());'과 같다. 대부분의 경우 이처럼 생략한다.

e. Juicer.makeJuice(new FruitBox<Object>()); // 에러. 지네릭 메서드의 타입 호출이 생략되지 않았다면, 'Juicer.**<Object>**makeJuice(new FruitBox<**Object**>());'과 같다. d번의 경우와같이 타입이 일치하긴 하지만, <T extends Fruit>로 제한이 걸려있으므로, 타입 T는 Fruit의 자손이어야 한다. Object는 Fruit의 자손이 아니므로 에러.

[12-3] 다음 중 올바르지 않은 문장을 모두 고르시오.

```
class Box<T extends Fruit> { // 지네릭 타입 T를 선언
    T item;

    void setItem(T item) { this.item = item; }
    T getItem() { return item; }
}
```

a. Box<?> b = new Box();
b. Box<?> b = new Box<>();
c. Box<?> b = new Box<Object>();
d. Box<Object> b = new Box<Fruit>();
e. Box b = new Box<Fruit>();
f. Box<? extends Fruit> b = new Box<Apple>();
g. Box<? extends Object> b = new Box<? extends Fruit>();

[정답] c, d, g

[해설]

a. Box<?> b = new Box(); // OK. Box<?>는 Box<? extends Object>를 줄여쓴 것이다. 객체 생성에 지네릭 타입을 지정해 주지 않았지만 문제가 되지는 않는다. 그래도, new Box()대신 new Box<>()를 사용하는 것이 좋다.

b. Box<?> b = new Box<>(); // OK. new Box<>();는 타입을 생략한 것으로, 일반적으로는 참조변수의 타입과 같은 타입으로 간주된다. 참조변수의 타입이 <?>, 즉 <? extends Object>이므로 생략된 타입은 Object이라고 생각하기 쉬운데, 여기서는 지네릭 클래스 Box에 정의된 타입이 <T extends Fruit>와 같이 제한되어 있기 때문에, 'Object'가 아니라 'Fruit'이 생략된 것으로 봐야 한다. 그래서 Box<?> b = new Box<**Object**>();와 같이 하면 에러가 발생한다. Object는 Fruit의 자손이 아니기 때문이다.

```
Box<?> b = new Box<>();         // OK. Box<?> b = new Box<Fruit>();와 동일
Box<?> b = new Box<Object>();   // 에러
Box<?> b = new Box<Fruit>();    // OK
```

'Box<? extends Object>'는 Box<Object>와 같지 않음에 주의하자. 지네릭 클래스 Box는 타입 T가 Fruit의 자손으로 제한되어 있기 때문에, Box<**Object**>와 같이 Fruit의 자손이 아닌 클래스를 대입할 수 없다. 그러나, 'Box<? extneds Object>'와 같이 와일드 카드를 사용하는 것은 가능하다.

c. Box<?> b = new Box<Object>(); // 에러 b의 설명 참조
d. Box<**Object**> b = new Box<**Fruit**>(); // 에러. 타입 불일치
e. Box b = new Box<Fruit>(); // OK. 바람직하지 않음. 'Box<?> b'가 더 나음.
f. Box<? extends Fruit> b = new Box<Apple>(); // OK.
g. Box<? extends Object> b = new Box<? extends Fruit>(); // 에러. new연산자는 타입이 명확해야하므로 와일드 카드와 같이 사용불가

[12-4] 아래의 메서드는 두 개의 ArrayList를 매개변수로 받아서, 하나의 새로운 ArrayList로 병합하는 메서드이다. 이를 지네릭 메서드로 변경하시오.

```
public static ArrayList<? extends Product> merge(
    ArrayList<? extends Product> list, ArrayList<? extends Product> list2) {
        ArrayList<? extends Product> newList = new ArrayList<>(list);

        newList.addAll(list2);

        return newList;
}
```

[정답]

```
    public static <T extends Product> ArrayList<T> merge(
        ArrayList<T> list, ArrayList<T> list2) {
            ArrayList<T> newList = new ArrayList<>(list);

            newList.addAll(list2);

            return newList;
    }
```

[12-5] 아래는 예제7-3에 열거형 Kind와 Number를 새로 정의하여 적용한 것이다. (1)에 알맞은 코드를 넣어 예제를 완성하시오. (Math.random()을 사용했으므로 실행결과가 달라질 수 있다.)

[연습문제]/ch12/Exercise12_5.java

```
class Exercise12_5 {
   public static void main(String args[]) {
        Deck d = new Deck();       // 카드 한 벌(Deck)을 만든다.
        Card c = d.pick(0);        // 섞기 전에 제일 위의 카드를 뽑는다.
        System.out.println(c);   // System.out.println(c.toString());과 같다.

        d.shuffle();             // 카드를 섞는다.
        c = d.pick(0);           // 섞은 후에 제일 위의 카드를 뽑는다.
        System.out.println(c);
   }
}

class Deck {
   final int CARD_NUM = Card.Kind.values().length
                                    * Card.Number.values().length; // 카드의 개수
   Card cardArr[] = new Card[CARD_NUM];  // Card객체 배열을 포함

   Deck () {
        int i=0;

        for(Card.Kind kind : Card.Kind.values()) {
            for(Card.Number num : Card.Number.values()) {
                cardArr[i++] = new Card(kind, num);
            }
        }
   }

   Card pick(int index) {     // 지정된 위치(index)에 있는 카드 하나를 꺼내서 반환
        return cardArr[index];
   }

   Card pick() {          // Deck에서 카드 하나를 선택한다.
        int index = (int)(Math.random() * CARD_NUM);
        return pick(index);
   }

   void shuffle() { // 카드의 순서를 섞는다.
        for(int i=0; i < cardArr.length; i++) {
            int r = (int)(Math.random() * CARD_NUM);

            Card temp = cardArr[i];
            cardArr[i] = cardArr[r];
            cardArr[r] = temp;
        }
   }
} // Deck클래스의 끝

// Card클래스
class Card {
   enum Kind   { CLOVER, HEART, DIAMOND, SPADE }
```

```
    enum Number {
        ACE, TWO, THREE, FOUR, FIVE,
        SIX, SEVEN, EIGHT, NINE, TEN,
        JACK, QUEEN, KING
    }

    Kind  kind;
    Number num;

    Card() {
        this(Kind.SPADE, Number.ACE);
    }

    Card(Kind kind, Number num) {
        this.kind  = kind;
        this.num   = num;
    }

    public String toString() {
        return "[" + kind.name() + "," + num.name() + "]";
    } // toString()의 끝
} // Card클래스의 끝
```

[실행결과]

```
[CLOVER,ACE]
[HEART,TEN]
```

[정답]

```
int i=0;

for(Card.Kind kind : Card.Kind.values()) {
    for(Card.Number num : Card.Number.values()) {
        cardArr[i++] = new Card(kind, num);
    }
}
```

[12-6] 다음 중 메타 애너테이션이 아닌 것을 모두 고르시오.

a. Documented

b. Target

c. Native

d. Inherited

[정답] c

애너테이션	설명
@Override	컴파일러에게 오버라이딩하는 메서드라는 것을 알린다.
@Deprecated	앞으로 사용하지 않을 것을 권장하는 대상에 붙인다.
@SuppressWarnings	컴파일러의 특정 경고메시지가 나타나지 않게 해준다.
@SafeVarargs	지네릭스 타입의 가변인자에 사용한다.(JDK1.7)
@FunctionalInterface	함수형 인터페이스라는 것을 알린다.(JDK1.8)
@Native	native메서드에서 참조되는 상수 앞에 붙인다.(JDK1.8)
@Target*	애너테이션이 적용가능한 대상을 지정하는데 사용한다.
@Documented*	애너테이션 정보가 javadoc으로 작성된 문서에 포함되게 한다.
@Inherited*	애너테이션이 자손 클래스에 상속되도록 한다.
@Retention*	애너테이션이 유지되는 범위를 지정하는데 사용한다.
@Repeatable*	애너테이션을 반복해서 적용할 수 있게 한다.(JDK1.8)

[표12-2] 자바에서 기본적으로 제공하는 표준 애너테이션(*가 붙은 것은 메타 애너테이션)

[12-7] 애너테이션 TestInfo가 다음과 같이 정의되어 있을 대, 이 애너테이션이 올바르게 적용되지 않은 것은?

```
@interface TestInfo {
    int count() default 1;
    String[] value() default "aaa";
}
```

a. @TestInfo class Exercise12_7 {}
b. @TestInfo(1) class Exercise12_7 {}
c. @TestInfo("bbb") class Exercise12_7 {}
d. @TestInfo("bbb","ccc") class Exercise12_7 {}

[정답] b, d

[해설]

a. @TestInfo class Exercise12_7 {}
default값이 지정되어 있는 요소는 애너테이션을 적용할 때값을 생략할 수 있다.

b. @TestInfo(1) class Exercise12_7 {}
요소의 이름이 value가 아닌 경우에는 요소의 이름을 생략할 수 없다.
'@TestInfo(count=1)' 이라고 써야 맞음.

c. @TestInfo("bbb") class Exercise12_7 {}
@TestInfo(count=1, value={"bbb"})의 생략된 형태

d. @TestInfo("bbb","ccc") class Exercise12_7 {}
요소의 타입이 배열이고, 지정하려는 값이 여러 개인 경우 괄호{}가 필요함.
@TestInfo({"bbb", "ccc"}) 또는 @TestInfo(value={"bbb","ccc"})와 같이 써야함

[연습문제 - 모범답안]

[13-1] 쓰레드를 구현하는 방법에는 Thread클래스로부터 상속받는 것과 Runnable인터페이스를 구현하는 것 두 가지가 있는데, 다음의 코드는 Thread클래스를 상속받아서 쓰레드를 구현한 것이다. 이 코드를 Runnable인터페이스를 구현하도록 변경하시오.

[연습문제]/ch13/Exercise13_1.java

```
class Exercise13_1 {
   public static void main(String args[]) {
        Thread1 th1 = new Thread1();

        th1.start();
   }
}

class Thread1 extends Thread {
   public void run() {
        for(int i=0; i < 300; i++) {
            System.out.print('-');
        }
   }
}
```

[정답]

[연습문제]/ch13/Exercise13_1_2.java

```
class Exercise13_1_2 {
   public static void main(String args[]) {
        Thread th1 = new Thread(new Thread1());

        th1.start();
   }
}

class Thread1 implements Runnable {
   public void run() {
        for(int i=0; i < 300; i++) {
            System.out.print('-');
        }
   }
}
```

[13-2] 다음 코드의 실행결과로 옳은 것은?

[연습문제]/ch13/Exercise13_2.java

```
class Exercise13_2 {
   public static void main(String[] args) {
        Thread2 t1 = new Thread2();
        t1.run();

        for(int i=0; i < 10; i++)
            System.out.print(i);

   }
}

class Thread2 extends Thread {
   public void run() {
        for(int i=0; i < 10; i++)
            System.out.print(i);
   }
}
```

a. 0102123345456768978 9처럼 0부터 9까지의 숫자가 섞여서 출력된다.

b. 0123456789012345678 9처럼 0부터 9까지의 숫자가 순서대로 출력된다.

c. IllegalThreadStateException이 발생한다.

[정답] b

[해설] Thread2클래스의 인스턴스를 생성하긴 했지만, start()가 아닌 run()을 호출함으로써 쓰레드를 실행시킨 것이 아니라 단순히 Thread2클래스의 메서드를 호출한 셈이 되었다. 만일 run()이 아닌 start()를 호출하였다면, 숫자가 섞여서 출력되었을 것이다.

[13-3] 다음 중 쓰레드를 일시정지 상태(WAITING)로 만드는 것이 아닌 것은? (모두 고르시오)

a. suspend()
b. resume()
c. join()
d. sleep()
e. wait()
f. notify()

[정답] b, f
[해설] resume()은 suspend()의 호출로 인해 일시정지 상태가 된 쓰레드를 실행대기상태로 바꿔준다. notify()역시 wait()의 호출로 인해 일시정지 상태가 된 쓰레드를 다시 실행대기 상태로 바꿔준다. join()은 현재 실행 중인 쓰레드를 멈추고 다른 쓰레드가 실행되도록 한다.

[13-4] 다음 중 interrupt()에 의해서 실행대기 상태(RUNNABLE)가 되지 않는 경우는? (모두 고르시오)

a. sleep()에 의해서 일시정지 상태인 쓰레드
b. join()에 의해서 일시정지 상태인 쓰레드
c. wait()에 의해서 일시정지 상태인 쓰레드
d. suspend()에 의해서 일시정지 상태인 쓰레드

[정답] d
[해설] suspend()를 제외한 나머지 메서드들은 interrupt()가 호출되면 InterruptedException이 발생하여 일시정지 상태에서 벗어나 실행대기 상태가 된다.(try-catch문으로 InterruptedException을 처리해주어야 한다.)

[13-5] 다음의 코드를 실행한 결과를 예측하고, 직접 실행한 결과와 비교하라. 만일 예측한 결과와 실행한 결과의 차이가 있다면 그 이유를 설명하라.

[연습문제]/ch13/Exercise13_5.java

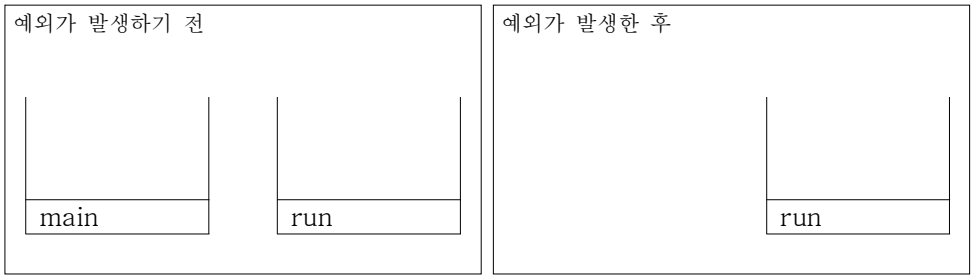
```
class Exercise13_5
{
   public static void main(String[] args) throws Exception
   {
        Thread3 th1 = new Thread3();
        th1.start();

        try {
            Thread.sleep(5*1000);  // main쓰레드를 5초간 정지시킨다.
        } catch(Exception e) {}

        throw new Exception("꽝~!!!");
   }
}

class Thread3 extends Thread {
   public void run() {
        for(int i=0; i < 10; i++) {
            System.out.println(i);

            try {
                Thread.sleep(1000);
            } catch(Exception e) {}
        }
   } // run()
}
```

[정답]

[실행결과]

```
0
1
2
3
4
5
Exception in thread "main" java.lang.Exception: 꽝~!!!
       at Exercise13_5.main(Exercise13_5.java:12)
6
7
8
9
```

[해설] main과 th1 두 개의 쓰레드는 별도의 호출스택(call stack)에서 실행된다. 그래서 main에서 예외가 발생하여 종료되고 호출스택이 없어져도, 쓰레드 th1이 실행되는 호출스택에는 아무런 영향을 미치지 못한다.


예외가 발생하기 전
main
run
예외가 발생한 후
run

[13-6] 다음의 코드를 실행한 결과를 예측하고, 직접 실행한 결과와 비교하라. 만일 예측한 결과와 실행한 결과의 차이가 있다면 그 이유를 설명하라.

[연습문제]/ch13/Exercise13_6.java

```
class Exercise13_6
{
   public static void main(String[] args) throws Exception
   {
        Thread4 th1 = new Thread4();
        th1.setDaemon(true);  // 쓰레드 th1을 데몬쓰레드로 설정한다.
        th1.start();

        try {
            th1.sleep(5*1000);
        } catch(Exception e) {}

        throw new Exception("꽝~!!!");
   }
}

class Thread4 extends Thread {
   public void run() {
        for(int i=0; i < 10; i++) {
            System.out.println(i);

            try {
                Thread.sleep(1000);
            } catch(Exception e) {}
        }
   } // run()
}
```

[정답]

[실행결과]

```
0
1
2
3
4
5Exception in thread "main"
java.lang.Exception: 꽝~!!!
       at Exercise13_6.main(Exercise13_6.java:13)
```

[해설] 문제13-6에 'th1.setDaemon(true);' 한 문장을 추가해서 쓰레드 th1을 데몬 쓰레드로(daemon thread) 설정하였다. 데몬 쓰레드는 일반 쓰레드(데몬 쓰레드가 아닌 쓰레드)가 모두 종료되면 자동 종료되므로, main쓰레드(일반쓰레드)가 종료됨과 동시에 자동 종료된다. 그래서 문제13-6과는 달리 쓰레드th1이 main메서드의 종료와 동시에 종료되었다.

[13-7] 다음의 코드는 쓰레드 th1을 생성해서 실행시킨 다음 6초 후에 정지시키는 코드이다. 그러나 실제로 실행시켜보면 쓰레드를 정지시킨 다음에도 몇 초가 지난 후에서야 멈춘다. 그 이유를 설명하고, 쓰레드를 정지시키면 바로 정지되도록 코드를 개선하시오.

[연습문제]/ch13/Exercise13_7.java

```
class Exercise13_7
{
   static boolean stopped = false;

   public static void main(String[] args)
   {
        Thread5 th1 = new Thread5();
        th1.start();

        try {
            Thread.sleep(6*1000);
        } catch(Exception e) {}

        stopped = true;   // 쓰레드를 정지시킨다.
        th1.interrupt(); // 일시정지 상태에 있는 쓰레드를 깨운다.
        System.out.println("stopped");
   }
}

class Thread5 extends Thread {
   public void run() {
        // Exercise13_7.stopped의 값이 false인 동안 반복한다.
        for(int i=0; !Exercise13_7.stopped; i++) {
            System.out.println(i);

            try {
                Thread.sleep(3*1000);
            } catch(Exception e) {}
        }
   } // run()
}
```

[실행결과]

```
0
1
2
stopped
```

[정답] Exercise13_7.stopped의 값이 바뀌어도 for문내의 Thread.sleep(3*1000);문장에 의해 일시정지 상태에 있는 경우, 시간이 지나서 일시정지 상태를 벗어날 때까지 for문을 벗어날 수 없기 때문에 이런 현상이 발생한다. 그래서 interrupt()를 호출해서 자고 있는(sleep()에 의해 일시정지 상태에 있는) 쓰레드를 깨워야 즉시 정지하게 된다.

[해설] 쓰레드 th1은 아래의 반복문을 수행하다가 main메서드에서 Exercise13_7.stopped의 값을 true로 바꾸면 반복문을 빠져나와 수행을 종료하게 된다. 반복문 안에 쓰레드를 3초 동안 일시정지 상태로 하는 'Thread.sleep(3*1000)'이 있기 때문에 Exercise13_7.

stopped의 값이 바뀌었다 하더라도 일시정지 상태에 있다면, 일시정지 상태가 끝나야만 반복문을 빠져나오게 된다.

```
public void run() {
    // Exercise13_7.stopped의 값이 false인 동안 반복한다.
    for(int i=0; !Exercise13_7.stopped; i++) {
        System.out.println(i);

        try {
            Thread.sleep(3*1000); // 3초간 쉰다.
        } catch(Exception e) {}
    }
} // run()
```

그래서 쓰레드의 실행을 바로 종료시키려면 Exercise13_7.stopped의 값을 true로 바꾸는 것만으로는 부족하다. 그 외에 다른 방법이 더 필요하다. 그것은 바로 interrupt()를 호출하는 것이다.

```
stopped = true; // 쓰레드를 정지시킨다.
```

→

```
stopped = true; // 쓰레드를 정지시킨다.
th1.interrupt(); // 쓰레드를 깨운다.
```

interrupt()는 InterruptedException을 발생시킴으로써 Thread.sleep()에 의해 일시정지 상태에 있던 쓰레드를 즉시 깨운다.

그래서 Exercise13_7.stopped의 값을 true로 바꾸고, interrupt()를 호출하면 지연 없이 즉시 쓰레드를 멈추게 할 수 있다.

[13-8] 다음의 코드는 텍스트기반의 타자연습게임인데 WordGenerator라는 쓰레드가 Vector에 2초마다 단어를 하나씩 추가하고, 사용자가 단어를 입력하면 Vector에서 일치하는 단어를 삭제하도록 되어 있다. WordGenerator의 run()을 완성하시오.

[연습문제]/ch13/Exercise13_8.java

```
import java.util.*;

class Exercise13_8
{
   Vector words = new Vector();
   String[] data = {"태연","유리","윤아","효연","수영","서현","티파니","써니","제시카"};

   int interval = 2 * 1000;  // 2초

   WordGenerator wg = new WordGenerator();

   public static void main(String args[])
   {
       Exercise13_8 game = new Exercise13_8();

       game.wg.start(); // 단어를 생성하는 쓰레드를 실행시킨다.

       Vector words = game.words;

       while(true) {
           System.out.println(words);

           String prompt = ">>";
           System.out.print(prompt);

           // 화면으로부터 라인단위로 입력받는다.
           Scanner s = new Scanner(System.in);
           String input = s.nextLine().trim();

           int index = words.indexOf(input); // 입력받은 단어를 words에서 찾는다.

           if(index!=-1) {  // 찾으면
               words.remove(index); // words에서 해당 단어를 제거한다.
           }
       }
   } // main

   class WordGenerator extends Thread {
       public void run() {
           while(true) {
//             1. interval(2초)마다 배열 data의 값 중 하나를 임의로 선택해서

               int rand = (int)(Math.random()*data.length);
//             2. words에 저장한다.
               words.add(data[rand]);

               try {
                   Thread.sleep(interval);  // 2초(interval) 동안 쉰다.
               } catch(Exception e) {}
           }
```

```
        } // end of run()
    } // class WordGenerator
} // Exercise13_8
```

[실행결과]

```
[]
>>
[서현]
>>서현
[수영, 윤아]
>>수영
[윤아, 유리]
>>유리
[윤아, 티파니]
>>티파니
[윤아, 윤아, 유리]
>>윤아
[윤아, 유리]
>>유리
[윤아, 효연]
>>효연
[윤아, 티파니]
>>윤아
[티파니, 윤아]
>>티파니
[윤아, 수영, 써니]
>>
```

[해설] 쉬운 문제라서 별로 설명할 것이 없다. 쓰레드가 그렇게 어려운 것만은 아니라고 느낄 수 있으면 좋겠다. 이 예제를 발전시켜서 GUI를 갖춘 타자게임을 만들어 보면 재미있을 것이다.(카페의 'java1000제'게시판에 개발단계별로 공개되어 있음)

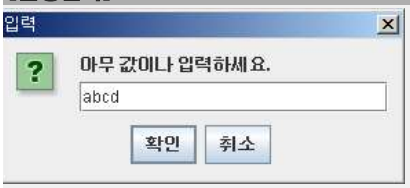
혹시라도 질문이 있으면 http://cafe.naver.com/javachobostudy.cafe의 게시판에 올려주길 바란다.

[13-9] 다음은 사용자의 입력을 출력하고 종료하는 프로그램을 작성한 것으로, 10초 동안 입력이 없으면 자동종료되어야 한다. 그러나 실행결과를 보면, 사용자의 입력이 10초 안에 이루어졌음에도 불구하고 프로그램이 즉시 종료되지 않는다. 사용자로부터 입력받는 즉시 프로그램이 종료되도록 수정하시오.

[연습문제]/ch13/Exercise13_9.java

```
import javax.swing.JOptionPane;

class Exercise13_9 {
   public static void main(String[] args) throws Exception     {
        Exercise13_9_1 th1 = new Exercise13_9_1();
        th1.start();

        String input = JOptionPane.showInputDialog("아무 값이나 입력하세요.");
        System.out.println("입력하신 값은 " + input + "입니다.");
        th1.interrupt();  // 쓰레드에게 작업을 멈추라고 요청한다.
   }
}

class Exercise13_9_1 extends Thread {
   public void run() {
        int i = 10;

        while(i!=0 && !isInterrupted()) {
            System.out.println(i--);

            try {
                Thread.sleep(1000);  // 1초 지연
            } catch(InterruptedException e) {
                interrupt();
            }
        }

        System.out.println("카운트가 종료되었습니다.");
   } // main
}
```

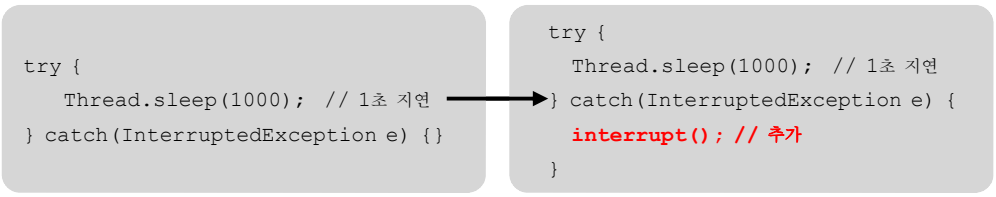
[실행결과]




```
10
9
8
입력하신 값은 abcd입니다.
카운트가 종료되었습니다.
```

[해설]

sleep()에 의해 쓰레드가 잠시 멈춰있을 때, interrupt()를 호출하면 InterruptedException이 발생되고 쓰레드의 interrupted상태는 false로 자동 초기화된다.



그래서, 위와 같이 catch블럭에 interrupt()를 추가로 넣어줘서 쓰레드의 interrupted상태를 true로 다시 바꿔줘야 한다. 보다 자세한 내용은 p.754를 참고하자.

[연습문제 - 모범답안]

[14-1] 메서드를 람다식으로 변환하여 아래의 표를 완성하시오.

메서드	람다식

`int max(int a, int b) {`
`  return a > b ? a : b;`
`}`
	`(int a, int b) -> a > b ? a : b`

`int printVar(String name, int i) {`
`  System.out.println(name+"="+i );`
`}`
	`(String name, int i) ->`
	`  { System.out.println(name+"="+i); }`

	`(name, i) ->` **`  { System.out.println(name+"="+i); }`**
	`(name, i) ->` **`  System.out.println(name+"="+i)`**

`int square(int x) {`
`  return x * x;`
`}`
	`(int x) -> x * x`

	`(x) -> x * x`
	`x -> x * x`

`int roll() {`
`  return (int)(Math.random() * 6);`
`}`
	`() -> { return (int)(Math.random()*6); }`

	`() -> (int)(Math.random() * 6)`

`int sumArr(int[] arr) {`
`  int sum = 0;`
`  for(int i : arr)`
`  sum += i;`
`  return sum;`
`}`
	`(int[] arr) -> {`
	`  int sum = 0;`
	`  for(int i : arr)`
	`  sum += i;`
	`  return sum;`
	`}`

`int[] emptyArr() {`
`  return new int[]{};`
`}`
	`() -> new int[]{}`

[14-2] 람다식을 메서드 참조로 변환하여 표를 완성하시오. (변환이 불가능한 경우, '변환불가' 라고 적어야함.)

람다식	메서드 참조
`(String s) -> s.length()`	**`String::length`**
`()-> new int[]{}`	**`int[]::new`**
`arr -> Arrays.stream(arr)`	**`Arrays::stream`**
`(String str1, String str2) -> str1.equals(str2)`	**`String::equals`**
`(a, b) -> Integer.compare(a, b)`	**`Integer::compare`**
`(String kind, int num) -> new Card(kind, num)`	**`Card::new`**
`(x) -> System.out.println(x)`	**`System.out::println`**
`()-> Math.random()`	**`Math::random`**
`(str) -> str.toUpperCase()`	**`String::toUpperCase`**
`() -> new NullPointerException()`	**`NullPointerException::new`**
`(Optional opt) -> opt.get()`	**`Optional::get`**
`(StringBuffer sb, String s) -> sb.append(s)`	**`StringBuffer::append`**
`(String s) -> System.out.println(s)`	**`System.out::println`**

[14-3] 아래의 괄호안에 알맞은 함수형 인터페이스는?

```
(     ) f;  // 함수형 인터페이스 타입의 참조변수 f를 선언.
f = (int a, int b) -> a > b ? a : b;
```

a. Function
b. BiFunction
c. Predicate
d. IntBinaryOperator
e. IntFunction

[정답] d

[해설]
참조하려는 람다식의 매개변수가 int타입 두 개이고, 반환값의 타입 역시 int이므로, 하나의 매개변수만 정의되어 있는 Function, Predicate, IntFunction은 적합하지 않다.
BiFunction은 두 개의 매개변수를 갖지만, int와 같은 기본형은 사용할 수 없기 때문에 IntBinaryOperator를 사용해야한다.

```
package java.util.function;

@FunctionalInterface
public interface IntBinaryOperator {
    int applyAsInt(int left, int right);
}
```

만일 매개변수의 타입을 생략했다면, BinaryOperator<Integer>로 다룰 수도 있다. 아래의 두 문장은 동일하다.

```
BinaryOperator<Integer> f = (a, b) -> a > b ? a : b;
BinaryOperator<Integer> f = (Integer a, Integer b) -> a > b ? a : b;
```

[14-4] 문자열 배열 strArr의 모든 문자열의 길이를 더한 결과를 출력하시오.

```
String[] strArr = { "aaa","bb","c", "dddd" };
```

[실행결과]

```
sum=10
```

[정답]

[연습문제]/ch14/Exercise14_4.java

```
import java.util.stream.*;

class Exercise14_4 {
   public static void main(String[] args) {
        String[] strArr = { "aaa","bb","c", "dddd" };
        Stream<String> strStream = Stream.of(strArr);

//      int sum = strStream.mapToInt(s-> s.length()).sum();
        int sum = strStream.mapToInt(String::length).sum();
        System.out.println("sum="+sum);
   }
}
```

[해설]

아래와 같이 map()을 이용해서 Stream<String>을 Stream<Integer>으로 변환한 다음에 스트림의 각 요소를 더해도 되지만,

```
Stream<Integer> integerStream = strStream.map(s -> s.length());
Integer sum = integerStream.reduce(0, (a, b)-> Integer.sum(a, b));
```

이럴 때는 mapToInt()를 사용해서 Stream<String>을 IntStream으로 변환하는 것이 좋다. IntStream에는 sum(), average(), max(), min()과 같이 편리한 메서드를 가지고 있기 때문이다.

```
IntStream intStream = strStream.mapToInt(String::length);
int sum = intStream.sum();
```

위의 두 문장을 줄이면, 다음과 같이 더 간단해 진다.

```
int sum = strStream.mapToInt(String::length).sum();
```

[14-5] 문자열 배열 strArr의 문자열 중에서 가장 긴 것의 길이를 출력하시오.

```
String[] strArr = { "aaa","bb","c", "dddd" };
```

[실행결과]

```
4
```

[정답]

[연습문제]/ch14/Exercise14_5.java

```
import java.util.*;
import java.util.stream.*;

class Exercise14_5 {
  public static void main(String[] args) {
      String[] strArr = { "aaa","bb","c", "dddd" };
      Stream<String> strStream = Stream.of(strArr);

      strStream.map(String::length) // strStream.map(s-> s.length())
           .sorted(Comparator.reverseOrder())      // 문자열 길이로 역순 정렬
           .limit(1).forEach(System.out::println); // 제일 긴 문자열의 길이 출력
  }
}
```

[해설]

문제14-5와 유사하지만, sorted()를 사용해서 정렬을 해야한다. 간단한 문제이므로 설명은 생략한다.

참고로 가장 긴 문자열 자체를 출력하려면 다음과 같이 하면 된다.

```
String[] strArr = { "aaa","bb","c", "dddd" };
Stream.of(strArr).sorted(Comparator.comparingInt(String::length).reversed())
                 .limit(1).forEach(System.out::println); // dddd가 출력된다.
```

[14-6] 임의의 로또번호(1~45)를 정렬해서 출력하시오.

[실행결과]

```
1
20
25
33
35
42
```

[정답]

[연습문제]/ch14/Exercise14_6.java

```
import java.util.*;
import java.util.stream.*;

class Exercise14_6 {
   public static void main(String[] args) {
        new Random().ints(1,46)   // 1~45사이의 정수(46은 포함안됨)
                    .distinct()   // 중복제거
                    .limit(6)     // 6개만
                    .sorted()     // 정렬
                    .forEach(System.out::println); // 화면에 출력
   }
}
```

[해설]

Random클래스의 ints()는 지정된 범위 내의 임의의 정수를 요소로 하는 무한 스트림을 반환한다. 무한스트림이므로 limit()이 필요하다. sorted()로 정렬한 다음, forEach()로 화면에 출력한다.

[14-7] 두 개의 주사위를 굴려서 나온 눈의 합이 6인 경우를 모두 출력하시오.

[Hint] 배열을 사용하시오.

[실행결과]

```
[1, 5]
[2, 4]
[3, 3]
[4, 2]
[5, 1]
```

[정답]

[연습문제]/ch14/Exercise14_7.java

```
import java.util.*;
import java.util.stream.*;

class Exercise14_7 {
   public static void main(String[] args) {
        Stream<Integer> die = IntStream.rangeClosed(1,6).boxed();

        die.flatMap(i-> Stream.of(1,2,3,4,5,6).map(i2 -> new int[]{ i, i2 }))
           .filter(iArr-> iArr[0]+iArr[1]==6)
           .forEach(iArr -> System.out.println(Arrays.toString(iArr)));
   }
}
```

[해설]

아래의 붉은 색으로 표시된 람다식은, Stream<Integer>인 die의 요소, 즉 Integer를 int배열의 스트림(Stream<int[]>)로 변환한다.

```
                          Integer → Stream<int[]>
die.flatMap( i-> Stream.of(1,2,3,4,5,6).map(i2 -> new int[]{ i, i2 }) )
Stream<Integer> → Stream<int[]>
```

만일 flatMap을 쓰지 않고 map()을 사용했다면, 연산결과는 'Stream<Stream<int[]>>'가 되었을 것이다.

```
                          Integer → Stream<int[]>
die.map( i-> Stream.of(1,2,3,4,5,6).map(i2 -> new int[]{ i, i2 }) )
Stream<Integer> → Stream<Stream<int[]>>
```

아직 이해가 잘 안간다면, filter()를 주석처리해서 Stream<int[]>의 모든 요소가 출력되도록 변경해보자. 이해에 도움이 될 것이다.

```
        die.flatMap(i-> Stream.of(1,2,3,4,5,6).map(i2 -> new int[]{ i, i2 }))
//          .filter(iArr-> iArr[0]+iArr[1]==6)
            .forEach(iArr -> System.out.println(Arrays.toString(iArr)));
```

자주 쓰이지는 않겠지만, 알아두면 좋은 내용이다. 난이도가 있기 때문에 본문에서는 제외하고 연습문제에 넣게 되었다. 이 문제를 푸느라 책을 복습하며 다방면으로 고민했으면 하는 바램이다.

[14-8] 다음은 불합격(150점 미만)한 학생의 수를 남자와 여자로 구별하여 출력하는 프로그램이다. (1)에 알맞은 코드를 넣어 완성하시오.

[연습문제]/ch14/Exercise14_8.java

```
import java.util.*;
import java.util.function.*;
import java.util.stream.*;
import static java.util.stream.Collectors.*;
import static java.util.Comparator.*;

class Student {
   String name;
   boolean isMale; // 성별
   int hak;        // 학년
   int ban;        // 반
   int score;

   Student(String name, boolean isMale, int hak, int ban, int score) {
       this.name    = name;
       this.isMale  = isMale;
       this.hak= hak;
       this.ban= ban;
       this.score  = score;
   }
   String  getName()  { return name;}
   boolean  isMale()   { return isMale;}
   int     getHak()   { return hak;}
   int     getBan()   { return ban;}
   int     getScore() { return score;}

   public String toString() {
       return String.format("[%s, %s, %d학년 %d반, %3d점]", name, isMale ?
                                                    "남":"여", hak, ban, score);
   }
   // groupingBy()에서 사용
   enum Level { HIGH, MID, LOW }  // 성적을 상, 중, 하 세 단계로 분류
}

class Exercise14_8 {
   public static void main(String[] args) {
       Student[] stuArr = {
           new Student("나자바", true,  1, 1, 300),
           new Student("김지미", false, 1, 1, 250),
           new Student("김자바", true,  1, 1, 200),
           new Student("이지미", false, 1, 2, 150),
           new Student("남자바", true,  1, 2, 100),
           new Student("안지미", false, 1, 2,  50),
           new Student("황지미", false, 1, 3, 100),
           new Student("강지미", false, 1, 3, 150),
           new Student("이자바", true,  1, 3, 200),

           new Student("나자바", true,  2, 1, 300),
           new Student("김지미", false, 2, 1, 250),
           new Student("김자바", true,  2, 1, 200),
           new Student("이지미", false, 2, 2, 150),
```

```
            new Student("남자바", true,  2, 2, 100),
            new Student("안지미", false, 2, 2,  50),
            new Student("황지미", false, 2, 3, 100),
            new Student("강지미", false, 2, 3, 150),
            new Student("이자바", true,  2, 3, 200)
        };

        Map<Boolean, Map<Boolean, Long>> failedStuBySex = Stream.of(stuArr)
            .collect(
                partitioningBy(
                    Student::isMale,
                    partitioningBy(s -> s.getScore() < 150, counting())
                )
            );

        long failedMaleStuNum   = failedStuBySex.get(true).get(true);
        long failedFemaleStuNum = failedStuBySex.get(false).get(true);

        System.out.println("불합격[남자]:"+ failedMaleStuNum   +"명");
        System.out.println("불합격[여자]:"+ failedFemaleStuNum +"명");
    }
}
```

[실행결과]

```
불합격[남자]:2명
불합격[여자]:4명
```

[해설]

합격과 불합격, 즉 true와 false로 나누는 것이므로 groupingBy()보다 partitioningBy()가 더 적합하다. 그리고는 다시 남자와 여자로 나누어야 하므로 한번 더 partitioningBy()를 적용하면 된다.

참고로 counting()은 Collectors클래스의 static메서드이다.

[14-9] 다음은 각 반별 총점을 학년 별로 나누어 출력하는 프로그램이다. (1)에 알맞은 코드를 넣어 완성하시오.

[연습문제]/ch14/Exercise14_9.java

```
import java.util.*;
import java.util.function.*;
import java.util.stream.*;
import static java.util.stream.Collectors.*;
import static java.util.Comparator.*;

class Student {
   String name;
   boolean isMale; // 성별
   int hak;        // 학년
   int ban;        // 반
   int score;

   Student(String name, boolean isMale, int hak, int ban, int score) {
       this.name    = name;
       this.isMale  = isMale;
       this.hak = hak;
       this.ban = ban;
       this.score  = score;
   }

   String  getName()  { return name;}
   boolean  isMale()   { return isMale;}
   int     getHak()   { return hak;}
   int     getBan()   { return ban;}
   int     getScore() { return score;}

   public String toString() {
       return String.format("[%s, %s, %d학년 %d반, %3d점]", name, isMale ?
                                                    "남":"여", hak, ban, score);
   }

   enum Level {
       HIGH, MID, LOW
   }
}

class Exercise14_9 {
   public static void main(String[] args) {
       Student[] stuArr = {
           new Student("나자바", true,  1, 1, 300),
           new Student("김지미", false, 1, 1, 250),
           new Student("김자바", true,  1, 1, 200),
           new Student("이지미", false, 1, 2, 150),
           new Student("남자바", true,  1, 2, 100),
           new Student("안지미", false, 1, 2,  50),
           new Student("황지미", false, 1, 3, 100),
           new Student("강지미", false, 1, 3, 150),
           new Student("이자바", true,  1, 3, 200),

           new Student("나자바", true,  2, 1, 300),
           new Student("김지미", false, 2, 1, 250),
```

```
            new Student("김자바", true,  2, 1, 200),
            new Student("이지미", false, 2, 2, 150),
            new Student("남자바", true,  2, 2, 100),
            new Student("안지미", false, 2, 2,  50),
            new Student("황지미", false, 2, 3, 100),
            new Student("강지미", false, 2, 3, 150),
            new Student("이자바", true,  2, 3, 200)
        };

        Map<Integer, Map<Integer, Long>> totalScoreByHakAndBan =
            Stream.of(stuArr)
                .collect(
                    groupingBy(
                        Student::getHak,
                        groupingBy(
                            Student::getBan,
                            summingLong(Student::getScore)
                        )
                    )
                );

        for(Object e : totalScoreByHakAndBan.entrySet()) {
            System.out.println(e);
        }

    } // main의 끝
}
```

[실행결과]

```
1={1=750, 2=300, 3=450}
2={1=750, 2=300, 3=450}
```

[해설]

여기서는 데이터에 포함된 학년이 2개 뿐이지만, 여러 개 일 수도 있으므로 partitioningBy()가 아닌 groupingBy()을 사용했다. 다시 반별로 구룹화해야하므로 groupingBy()를 한번 더 적용했다. 간단한 2중 그룹화이므로 그리 어렵지는 않을 것이다.

counting()처럼 summingLong()은 Collectors클래스의 static메서드이다.

[연습문제 - 모범답안]

[15-1] 커맨드라인으로 부터 파일명과 숫자를 입력받아서, 입력받은 파일의 내용의 처음부터 입력받은 숫자만큼의 라인을 출력하는 프로그램(FileHead.java)을 작성하라.
[Hint] BufferedReader의 readLine()을 사용하라.

[실행결과]

```
C:\jdk1.8\work\ch15>java FileHead 10
USAGE: java FileHead 10 FILENAME

C:\jdk1.8\work\ch15>java FileHead 10 aaa
aaa은/는 디렉토리이거나, 존재하지 않는 파일입니다.

C:\jdk1.8\work\ch15>java FileHead 10 FileHead.java
1:import java.io.*;
2:
3:class FileHead
4:{
5:     public static void main(String[] args)
6:     {
7:            try {
8:                   int line = Integer.parseInt(args[0]);
9:                   String fileName = args[1];
10:

C:\jdk1.8\work\ch15>
```

[정답]

[연습문제]/ch15/FileHead.java

```
import java.io.*;

class FileHead
{
   public static void main(String[] args)
   {
        try {
            int lineNum = Integer.parseInt(args[0]);
            String fileName = args[1];

            File f = new File(fileName);

            if(f.exists()&&!f.isDirectory()) {
                BufferedReader br =
                                new BufferedReader(new FileReader(fileName));

                String line = "";
                int i=1;
```

```
                while((line = br.readLine())!=null && i <= lineNum) {
                    System.out.println(i+":"+line);
                    i++;
                }
            } else {
                throw new FileNotFoundException(fileName
                                    +"은/는 디렉토리이거나, 존재하지 않는 파일입니다.");
            }
        } catch(FileNotFoundException e) {
            System.out.println(e.getMessage());
        } catch(Exception e) {
            System.out.println("USAGE: java FileHead 10 FILENAME");
        }
    } // main
}
```

[해설] 파일을 라인단위로 읽기 위해 BufferedReader의 readLine()를 사용했다. 작업을 하기에 앞서 사용자로 부터 입력받은 이름의 파일이 존재하는지, 디렉토리는 아닌지 확인해야한다.

```
if(f.exists()&&!f.isDirectory()) {
        BufferedReader br = new BufferedReader(new FileReader(fileName));
```

그 다음에는 반복문을 이용해서 입력받은 라인 수 만큼만 파일의 내용을 라인화면에 출력한다.

```
while((line = br.readLine())!=null && i <= lineNum) {
        System.out.println(i+":"+line);
        i++;
}
```

참고로 while문에 사용된 조건식이 처리되는 순서는 다음과 같다.

```
(line = br.readLine())!=null
① line = br.readLine() // readLine()으로 읽은 라인(문자열)을 line에 저장한다.
② line != nulll        // line에 저장된 값이 null이 아닌지 비교한다.

* readLine()은 더 이상 읽을 라인이 없으면 null을 반환한다.
```

[15-2] 지정된 이진파일의 내용을 실행결과와 같이 16진수로 보여주는 프로그램(HexaViewer.java)을 작성하라.

[Hint] PrintStream과 printf()를 사용하라.

[실행결과]

```
C:\jdk1.8\work\ch15>java HexaViewer HexaViewer.class
CA FE BA BE 00 00 00 31 00 44 0A 00 0C 00 1E 09
00 1F 00 20 08 00 21 0A 00 08 00 22 0A 00 1F 00
23 07 00 24 0A 00 06 00 25 07 00 26 0A 00 08 00
27 0A 00 06 00 28 08 00 29 07 00 2A 0A 00 2B 00
2C 0A 00 08 00 2D 0A 00 08 00 2E 0A 00 06 00 2F
0A 00 08 00 2F 07 00 30 0A 00 12 00 31 07 00 32
01 00 06 3C 69 6E 69 74 3E 01 00 03 28 29 56 01
00 04 43 6F 64 65 01 00 0F 4C 69 6E 65 4E 75 6D
62 65 72 54 61 62 6C 65 01 00 04 6D 61 69 6E 01
00 16 28 5B 4C 6A 61 76 61 2F 6C 61 6E 67 2F 53
... 중간생략

C:\jdk1.8\work\ch15>
```

[정답]

[연습문제]/ch15/HexaViewer.java

```
import java.io.*;

class HexaViewer
{
   public static void main(String[] args) throws IOException
   {
        if(args.length!=1) {
            System.out.println("USAGE: java HexaViewer FILENAME");
            System.exit(0);
        }

        String inputFile = args[0];

        try {
            FileInputStream  input = new FileInputStream(inputFile);
            PrintStream     output = new PrintStream(System.out);

            int data = 0;
            int i=0;

            while((data = input.read())!=-1) {
                output.printf("%02X ", data);
                if(++i%16==0)
                    output.println();
            }

            input.close();
            output.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
```

```
    } // main
}
```

[해설] 사실 System.out이 PrintStream이기 때문에 굳이 PrintStream을 따로 생성해서 사용할 필요는 없다. 그냥 System.out.printf()를 사용하면 된다. 그러나 출력대상이 화면이 아니라 파일로 바뀐다면 아래의 붉은 색 부분만 변경하면 다른 곳은 고치지 않아도 된다는 장점이 있다.

```
FileInputStream  input  = new FileInputStream(inputFile);
PrintStream      output = new PrintStream(System.out);
```

data를 16진수로 출력하려면 printf의 format옵션 중에서 **'%x'**를 사용해야한다. 빈자리를 0으로 채우는 2자리 16진수이어야 하므로 **'%02x'**가 된다.

```
        while((data = input.read())!=-1) {
            output.printf("%02x ", data); // data를 두 자리의 16진수 형태로 출력한다.
            if(i++%16==0)
                output.println();
        }
```

format	설 명	결 과(int i=65)
%d	10진수(decimal integer)	65
%o	8진수(octal integer)	101
%x	**16진수(hexadecimal integer)**	41
%c	문자	A
%s	문자열	65
%5d	5자리 숫자. 빈자리는 공백으로 채운다.	65
%-5d	5자리 숫자. 빈자리는 공백으로 채운다.(왼쪽 정렬)	65
%05d	5자리 숫자. **빈자리는 0으로 채운다.**	00065

[15-3] 다음은 디렉토리의 요약정보를 보여주는 프로그램이다. 파일의 개수, 디렉토리의 개수, 파일의 총 크기를 계산하는 countFiles()를 완성하시오.

[연습문제]/ch15/DirectoryInfoTest.java

```
import java.io.*;

class DirectoryInfoTest {
   static int totalFiles = 0;
   static int totalDirs = 0;
   static int totalSize = 0;

   public static void main(String[] args) {
        if(args.length != 1) {
            System.out.println("USAGE : java DirectoryInfoTest DIRECTORY");
            System.exit(0);
        }

        File dir = new File(args[0]);

        if(!dir.exists() || !dir.isDirectory()) {
            System.out.println("유효하지 않은 디렉토리입니다.");
            System.exit(0);
        }

        countFiles(dir);

        System.out.println();
        System.out.println("총 " + totalFiles + "개의 파일");
        System.out.println("총 " + totalDirs + "개의 디렉토리");
        System.out.println("크기 " + totalSize + " bytes");
   } // main

   public static void countFiles(File dir) {
//      1. dir의 파일목록(File[])을 얻어온다.
        File[] files = dir.listFiles();

        for(int i=0; i < files.length; i++) {
//      2. 얻어온 파일목록의 파일 중에서...
          디렉토리이면, totalDirs의 값을 증가시키고 countFiles()를 재귀호출한다.
            if(files[i].isDirectory()) {
                totalDirs++;
                countFiles(files[i]);
            } else {
//      3. 파일이면, totalFiles를 증가시키고 파일의 크기를 totalSize에 더한다.
                totalFiles++;
                totalSize += files[i].length();
            }
        }
   } // countFiles
}
```

[실행결과]

```
C:\jdk1.8\work\ch15>java DirectoryInfoTest .

총 786개의 파일
```

```
총 27개의 디렉토리
크기 2566228 bytes

C:\jdk1.8\work\ch15>
```

[15-4] 커맨드라인으로 부터 여러 파일의 이름을 입력받고, 이 파일들을 순서대로 합쳐서 새로운 파일을 만들어 내는 프로그램(FileMergeTest.java)을 작성하시오. 단, 합칠 파일의 개수에는 제한을 두지 않는다.

[실행결과]

```
C:\jdk1.8\work\ch15>java FileMergeTest
USAGE: java FileMergeTest MERGE_FILENAME FILENAME1 FILENAME2 ...

C:\jdk1.8\work\ch15>java FileMergeTest result.txt 1.txt 2.txt 3.txt

C:\jdk1.8\work\ch15>type result.txt
1111111111
2222222222
33333333333333

C:\jdk1.8\work\ch15>java FileMergeTest result.txt 1.txt 2.txt

C:\jdk1.8\work\ch15>type result.txt
1111111111
2222222222

C:\jdk1.8\work\ch15>type 1.txt
1111111111

C:\jdk1.8\work\ch15>type 2.txt
2222222222

C:\jdk1.8\work\ch15>type 3.txt
33333333333333

C:\jdk1.8\work\ch15>
```

[정답]

[연습문제]/ch15/FileMergeTest.java

```
import java.io.*;
import java.util.*;

class FileMergeTest {
   public static void main(String[] args) {
        if(args.length < 2) { // 입력값이 2보다 작으면, 메세지를 출력하고 종료한다.
            System.out.println("USAGE:  java  FileMergeTest  MERGE_FILENAME
FILENAME1 FILENAME2 ...");
            System.exit(0);
        }

        try {
            Vector v = new Vector();

            for(int i=1; i < args.length;i++) {
                File f = new File(args[i]);

                if(f.exists()) {
                    v.add(new FileInputStream(args[i]));
```

```
                } else {
                    System.out.println(args[i]+ " - 존재하지 않는 파일입니다.");
                    System.exit(0);
                }
            }

            SequenceInputStream input = new SequenceInputStream(v.elements());
            FileOutputStream output  = new FileOutputStream(args[0]);

            int data = 0;

            while((data = input.read())!=-1) {
                output.write(data);
            }
        } catch(IOException e) {}
    } // main
}
```

[해설] 여러 개의 파일을 하나로 연결하기 위해서는 SequenceInputStream이 적합하다. SequenceInputStream은 여러 Stream을 하나의 Stream처럼 다룰 수 있다.

커맨드라인으로 입력받은 파일을 Vector에 저장하고, 이 Vector로 SequenceInputStream을 생성한 다음에 읽고 쓰면 끝이다. 반복되는 얘기지만, 사용자로 부터 입력받은 값은 항상 유효성체크를 해주어야한다. 입력받은 파일이 존재하지 않을 수도 있기 때문이다.

메서드 / 생성자	설 명
SequenceInputStream(Enumeration e)	Enumeration에 저장된 순서대로 입력스트림을 하나의 스트림으로 연결한다.
SequenceInputStream(InputStream s1, InputStream s2)	두 개의 입력스트림을 하나로 연결한다.

다음은 SequenceInputStream의 사용 예이다

```
[사용예1]
Vector files = new Vector();
files.add(new FileInputStream("file.001"));
files.add(new FileInputStream("file.002"));
SequenceInputStream in = new SequenceInputStream(files.elements());

[사용예2]
FileInputStream file1 = new FileInputStream("file.001");
FileInputStream file2 = new FileInputStream("file.002");
SequenceInputStream in = new SequenceInputStream(file1, file2);
```

[15-5] 다음은 FilterWriter를 상속받아서 직접 구현한 HtmlTagFilterWriter를 사용해서 주어진 파일에 있는 태그를 모두 제거하는 프로그램이다. HtmlTagFilterWriter의 write()가 태그를 제거하도록 코드를 완성하시오.

[연습문제]/ch15/Exercise15_5.java

```
import java.io.*;

class Exercise15_5
{
   public static void main(String[] args)
   {
        if(args.length != 2) {
            System.out.println("USAGE:    java    Exercise15_5    TAGET_FILE
RESULT_FILE");
            System.exit(0);
        }

        String inputFile  = args[0];
        String outputFile = args[1];

        try {
            BufferedReader input
                            = new BufferedReader(new FileReader(inputFile));
            HtmlTagFilterWriter output
                        = new HtmlTagFilterWriter(new FileWriter(outputFile));

            int ch = 0;

            while((ch=input.read())!=-1) {
                output.write(ch);
            }

            input.close();
            output.close();

        } catch(IOException e) {}
   }
}

class HtmlTagFilterWriter extends FilterWriter {
   StringWriter tmp = new StringWriter();
   boolean inTag = false;

   HtmlTagFilterWriter(Writer out) {
        super(out);
   }
   public void write(int c) throws IOException {
//       1. 출력할 문자(c)가 '<'이면 inTag의 값을 true로 한다.
        if(c=='<') {
            inTag = true;
//       2. 출력할 문자(c)가 '>'이면 inTag의 값을 false로 한다.
        } else if(c=='>' && inTag) {
            inTag = false;
//          새로운 StringWriter를 생성한다.(기존 StringWriter의 내용을 버린다.)
            tmp = new StringWriter();
```

```
            return;
        }
//        3. inTag의 값이 true이면, StringWriter에 문자(c)를 출력하고
        if(inTag)
            tmp.write(c);
        else
//          inTag의 값이 false이면, out에 문자(c)를 출력한다.
            out.write(c);
    }

    public void close() throws IOException {
        out.write(tmp.toString()); // StringWriter의 내용을 출력하고
        super.close();             // 조상의 close()를 호출해서 기반 스트림을 닫는다.
    }
}
```

[실행결과]

```
C:\jdk1.8\work\ch15>java Exercise15_5 test.html result.txt

C:\jdk1.8\work\ch15>type result.txt



 New Document


> 안녕하세요. 태그 없애기 테스트용 파일입니다.

< 태그가 열린 채로 끝난 것은 태그로 처리하지 마세요.

C:\jdk1.8\work\ch15>type test.html
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE> New Document </TITLE>
</HEAD>
<BODY>
> 안녕하세요. 태그 없애기 테스트용 파일입니다.
</BODY>
< 태그가 열린 채로 끝난 것은 태그로 처리하지 마세요.
```

[해설] 태그가 아닐 때는 기반스트림(out)에 출력하고, 태그가 시작되면(c가 '<'이면) inTag의 값을 true로 변경하고, inTag의 값이 true이면 tmp에 출력한다.

```
//        3. inTag의 값이 true이면, StringWriter에 문자(c)를 출력하고
        if(inTag)
            tmp.write(c); // StringWriter tmp = new StringWriter();
        else
//          inTag의 값이 false이면, out에 문자(c)를 출력한다.
            out.write(c);
```

태그가 끝나면(c가 '>'이면), inTag의 값을 false로 변경하고, 새로운 StringWriter를 생성해서 tmp에 할당한다. 이렇게 하면 기존의 내용(태그)은 없어지게 된다.

```
//      1. 출력할 문자(c)가 '<'이면 inTag의 값을 true로 한다.
        if(c=='<') {
            inTag = true;
//      2. 출력할 문자(c)가 '>'이면 inTag의 값을 false로 한다.
        } else if(c=='>' && inTag) {
            inTag = false;
//          새로운 StringWriter를 생성한다.(기존 StringWriter의 내용을 버린다.)
            tmp = new StringWriter();
            return;
        }
```

StringWriter에 내용이 남아있을 수 있으므로, 스트림을 닫기 전에 StringWriter의 내용을 스트림에 써주어야 한다.

```
    public void close() throws IOException {
        out.write(tmp.toString()); // StringWriter의 내용을 출력하고
        super.close();             // 조상의 close()를 호출해서 기반 스트림을 닫는다.
    }
```

[15-6] 다음은 콘솔 명령어 중에서 디렉토리를 변경하는 cd명령을 구현한 것이다. 알맞은 코드를 넣어 cd()를 완성하시오.

[연습문제]/ch14/Exercise15_6.java

```
import java.io.*;
import java.util.*;
import java.util.regex.*;

class Exercise15_6 {
   static String[] argArr;      // 입력한 매개변수를 담기위한 문자열배열
   static File curDir;          // 현재 디렉토리

   static {
       try {
           curDir = new File(System.getProperty("user.dir"));
       } catch(Exception e) {}
   }

   public static void main(String[] args) {
       Scanner s = new Scanner(System.in);

       while(true) {
           try {
               String prompt = curDir.getCanonicalPath() + ">>";
               System.out.print(prompt);

               // 화면으로부터 라인단위로 입력받는다.
               String input = s.nextLine();

               input = input.trim(); // 입력받은 값에서 불필요한 앞뒤 공백을 제거한다.
               argArr = input.split(" +");

               String command = argArr[0].trim();

               if("".equals(command)) continue;

               command = command.toLowerCase(); // 명령어를 소문자로 바꾼다.

               if(command.equals("q")) { // q 또는 Q를 입력하면 실행종료한다.
                   System.exit(0);
               } else if(command.equals("cd")) {
                   cd();
               } else {
                   for(int i=0; i < argArr.length;i++) {
                       System.out.println(argArr[i]);
                   }
               }
           } catch(Exception e) {
               e.printStackTrace();
               System.out.println("입력오류입니다.");
           }
       } // while(true)
   } // main

   public static void cd() {
```

```
        if(argArr.length==1) {
            System.out.println(curDir);
            return;
        } else if(argArr.length > 2) {
            System.out.println("USAGE : cd directory");
            return;
        }

        String subDir = argArr[1];

//      1. 입력된 디렉토리(subDir)가 ".."이면,
        if("..".equals(subDir)) { // 부모 디렉토리
//          1.1  현재 디렉토리의 조상 디렉토리를 얻어서 현재 디렉토리로 지정한다.
            File newDir = curDir.getParentFile();

            if(newDir==null) {
                System.out.println("유효하지 않은 디렉토리입니다.");
            } else {
                curDir = newDir;  // 조상 디렉토리를 현재 디렉토리로 지정한다.
            }
//      2. 입력된 디렉토리(subDir)가 "."이면, 단순히 현재 디렉토리의 경로를 화면에 출력한다.
        } else if(".".equals(subDir)) { // 현재 디렉토리
            System.out.println(curDir);
        } else {
//          3. 1 또는 2의 경우가 아니면,
//             3.1 입력된 디렉토리(subDir)가 현재 디렉토리의 하위디렉토리인지 확인한다.
            File newDir = new File(curDir, subDir);
            if(newDir.exists() && newDir.isDirectory()) {
//           3.2 확인결과가 true이면, 현재 디렉토리(curDir)을 입력된 디렉토리(subDir)로
                변경한다.
                curDir = newDir;
            } else {
//           3.3 확인결과가 false이면, "유효하지 않은 디렉토리입니다."고 화면에 출력한다.
                System.out.println("유효하지 않은 디렉토리입니다.");
            }
        } // if
    } // cd()
}
```

[실행결과]

```
C:\jdk1.8\work\ch15>java Exercise15_6
C:\jdk1.8\work\ch15>>
C:\jdk1.8\work\ch15>>cd ch15
유효하지 않은 디렉토리입니다.
C:\jdk1.8\work\ch15>>cd ..
C:\jdk1.8\work>>cd ch15
C:\jdk1.8\work\ch15>>
C:\jdk1.8\work\ch15>>cd .
C:\jdk1.8\work\ch15
C:\jdk1.8\work\ch15>>q
```

```
C:\jdk1.8\work\ch15>
```

[해설] 콘솔(console)의 cd명령(change directory)을 직접 구현해 보는 문제이다. cd명령 다음에 이동할 디렉토리를 적어주는데, 현재 디렉토리를 의미하는 '.'과 조상의 디렉토리를 의미하는 '..'도 처리해주어야 하므로 if-else if로 3가지 경우에 대해 처리하였다.

```
//  1. 입력된 디렉토리(subDir)가 ".."이면,
       if("..".equals(subDir)) { // 부모 디렉토리
           ...
//  2. 입력된 디렉토리(subDir)가 "."이면, 단순히 현재 디렉토리의 경로를 화면에 출력한다.
       } else if(".".equals(subDir)) { // 현재 디렉토리
           System.out.println(curDir);
       } else {
//  3. 1 또는 2의 경우가 아니면,
           ...
       } // if
```

나머지는 별로 어렵지 않으니 설명을 생략하겠다.

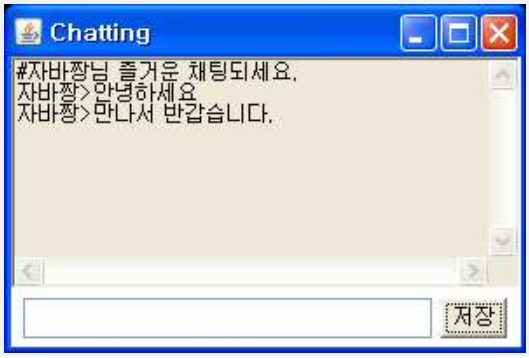
[15-7] 다음의 코드는 대화내용을 파일에 저장할 수 있는 채팅 프로그램이다. '저장' 버튼을 누르면 대화내용이 저장되도록 알맞은 코드를 넣어 완성하시오.

[연습문제]/ch15/ChatWin.java

```
import java.io.*;
import java.awt.*;
import java.awt.event.*;

class ChatWin extends Frame {
   String nickname = "";
   TextArea ta = new TextArea();
   Panel p = new Panel();
   TextField tf = new TextField(30);
   Button bSave = new Button("저장");

   ChatWin(String nickname) {
        super("Chatting");
        this.nickname = nickname;

        setBounds(200, 100, 300, 200);

        p.setLayout(new FlowLayout());
        p.add(tf);
        p.add(bSave);

        add(ta, "Center");
        add(p, "South");

        addWindowListener(new WindowAdapter(){
            public void windowClosing(WindowEvent e) {
                System.exit(0);
            }
        });

        bSave.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent ae) {
                FileDialog fileSave = new FileDialog(ChatWin.this,
                                                     "파일저장", FileDialog.SAVE);
                fileSave.setVisible(true);
                String fileName = fileSave.getDirectory()
                                  + fileSave.getFile();
                saveAs(fileName);
            }
        });

        EventHandler handler = new EventHandler();
        ta.addFocusListener(handler);
        tf.addFocusListener(handler);
        tf.addActionListener(handler);

        ta.setText("#" + nickname + "님 즐거운 채팅되세요.");
        ta.setEditable(false);

        setResizable(false);
        setVisible(true);
        tf.requestFocus();
   }
```

```
    void saveAs(String fileName) {
        FileWriter fw = null;
        BufferedWriter bw = null;

        try {
            fw = new FileWriter(fileName);
            bw= new BufferedWriter(fw);
            bw.write(ta.getText());      // TextArea의 내용을 파일에 저장한다.
        } catch (IOException ie) {
            ie.printStackTrace();
        } finally {
            try {
                if(bw!=null)
                    bw.close();
            } catch(IOException e) {}
        } // try
    } // saveAs메서드의 끝

    public static void main(String[] args) {
        if(args.length != 1) {
            System.out.println("USAGE : java ChatWin NICKNAME");
            System.exit(0);
        }

        new ChatWin(args[0]);
    } // main

    class EventHandler extends FocusAdapter implements ActionListener {
        public void actionPerformed(ActionEvent ae) {
            String msg = tf.getText();
            if("".equals(msg)) return;

            ta.append("\r\n" + nickname +">"+ msg);
            tf.setText("");
        }

        public void focusGained(FocusEvent e) {
            tf.requestFocus();
        }
    } // class EventHandler
} // class
```

[실행결과]



[해설] 답을 보고나니 쉬워서 허탈할 수도 있을 것 같다. BufferedWriter를 생성한 다음에, ta.getText()로 채팅내용을 얻어다 write(String str)을 통해 출력하면 되는 것이다.

```
FileWriter fw = null;
BufferedWriter bw = null;

try {
    fw = new FileWriter(fileName);
    bw= new BufferedWriter(fw);
    bw.write(ta.getText());      // TextArea의 내용을 파일에 저장한다.
```

아래와 같이 코드를 작성하면 write()에서 예외가 발생하면 bw.close()가 수행되지 않기 때문에 예외가 발생해도 수행되는 finally블럭에서 bw.close()가 호출되도록 하는 것이 좋다.

```
try {
    fw = new FileWriter(fileName);
    bw= new BufferedWriter(fw);
    bw.write(ta.getText());      // TextArea의 내용을 파일에 저장한다.
    bw.close();   // 윗 줄 bw.write()에서 예외가 발생하면 이 문장은 수행되지 않는다.
} catch (IOException ie) {
    ie.printStackTrace();
} // try
```

이전의 코드를 변경하여 finally블럭에서 bw.close()를 호출하도록 했다. bw.close() 역시 예외가 발생할 수 있으므로 try-catch문으로 감싸주었다.

```
try {
    fw = new FileWriter(fileName);
    bw= new BufferedWriter(fw);
    bw.write(ta.getText());      // TextArea의 내용을 파일에 저장한다.
} catch (IOException ie) {
    ie.printStackTrace();
} finally {
    try {
        if(bw!=null)
            bw.close();
    } catch(IOException e) {}
} // try
```

[15-8] 다음의 코드는 파일로부터 한 줄 씩 데이터를 읽어서 보여주는 프로그램이다. 버튼을 이용해서 첫 줄, 다음 줄, 이전 줄, 마지막 줄로 이동할 수 있으며, 각 줄의 개행문자는 '|'를 사용한다. (1)~(2)에 알맞은 코드를 넣어 완성하시오.

[연습문제]/ch15/WordStudy.java

```
import java.util.*;
import java.io.*;
import java.awt.*;
import java.awt.event.*;

class WordStudy extends Frame {
   Button first = new Button("<<");
   Button prev  = new Button("<");
   Button next  = new Button(">");
   Button last  = new Button(">>");
   Panel buttons = new Panel();
   TextArea ta = new TextArea();
   ArrayList wordList = new ArrayList();

   final String WORD_FILE = "word_data.txt";
   final String CR_LF = System.getProperty("line.separator"); // 개행문자

   int pos = 0;

   WordStudy(String title) {
        super(title);

        buttons.add(first);
        buttons.add(prev);
        buttons.add(next);
        buttons.add(last);
        add("South", buttons);
        add("Center", ta);

        EventHandler handler = new EventHandler();
        addWindowListener(handler);
        first.addActionListener(handler);
        prev.addActionListener(handler);
        next.addActionListener(handler);
        last.addActionListener(handler);

        loadFile(WORD_FILE);

        setBackground(Color.BLACK);
        setSize(300, 200);
        setLocation(200, 200);
        setResizable(true);
        setVisible(true);

        showFirst();
   }

   void showFirst() {
        pos = 0;
        display(pos);
   }
```

```
    void showPrevious() {
        pos = (pos > 0) ? --pos : 0;
        display(pos);
    }

    void showNext() {
        int size = wordList.size();
        pos = (pos < size-1) ? ++pos : size-1;
        display(pos);
    }

    void showLast() {
        pos = wordList.size()-1;
        display(pos);
    }

    void display(int pos) {  // pos위치에 있는 라인의 내용을 보여준다.
//      1. wordList에서 pos번째의 위치에 있는 데이터를 읽어온다.
        String tmp = (String)wordList.get(pos);
        StringBuffer sb = new StringBuffer(tmp.length());

//      2. StringTokenizer를 이용해서 '|'를 구분자로 자른다.
        StringTokenizer st = new StringTokenizer(tmp, "|");

//      3. 잘라진 Token에 개행문자(CR_LF)를 붙여서 StringBuffer에 붙인다.(append)
        while(st.hasMoreTokens()) {
            sb.append(st.nextToken()).append(CR_LF);
        }
//      4. StringBuffer의 내용을 뽑아서 TextArea에 보여준다.
        ta.setText(sb.toString());
    }

    void loadFile(String fileName) {
        try {
            BufferedReader br = new BufferedReader(new FileReader(fileName));
            String line = "";

//          1. BuffredReader와 FileReader를 이용해서 파일의 내용을 라인 단위로 읽는다.
            while((line = br.readLine())!=null) {
//          2. 읽어온 라인을 wordList에 저장한다.
                wordList.add(line);
            }
        } catch (IOException e) {
//          3. 만일 예외가 발생하면 프로그램을 종료한다.
            System.out.println("데이터 파일을 읽을 수 없습니다.");
            System.exit(1);
        }
    }

    public static void main(String args[]) {
        WordStudy mainWin = new WordStudy("Word Study");
    }

    class EventHandler extends WindowAdapter implements ActionListener {
        public void actionPerformed(ActionEvent ae) {
```

```
            Button b = (Button)ae.getSource();

            if(b==first) {
                showFirst();
            } else if(b==prev) {
                showPrevious();
            } else if(b==next) {
                showNext();
            } else if(b==last) {
                showLast();
            }
        }

        public void windowClosing(WindowEvent e) {
            System.exit(0);
        }
    } // class EventHandler
}
```

[실행결과]

[word_data.txt]

mandatory|1. 명령의|2. 통치를 위임받은|3. 강제의, 의무의(obli-gatory);필수의
preliminary|1. 사전 준비|2. 예비 시험|3. 주 경기 이전에 펼쳐지는 개막 경기
commitment|1. 위탁, 위임; 위원회 회부|2. 인도;투옥, 구류, 수감|3. 언질[공약]을 주기
prominent|1. 현저한, 두드러진 |2. 돌기한, 양각된|3. 탁월한, 걸출한, 유명한;중요한
Tell me the reason for coming here.|여기에 온 이유를 내게 말해라.
There is something different about you today.|너 오늘 평소와 좀 다르구나.
He jumped up and down when he got the news.|그는 뉴스를 듣고 펄쩍 뛰었다.
When I opened it, I found a surprise.|그 것을 열었을 때, 나는 놀라운 것을 발견했다.
I have known him since he was a child.|나는 그를 어려서부터 알고 있다.

[15-9] 다음은 메모장 프로그램의 일부인데, fileOpen()과 saveAs()가 아직 구현되어 있지 않다. 이 두 메서드를 구현하여 프로그램을 완성하시오.

[연습문제]/ch15/Exercise15_9.java

```
import java.awt.*;
import java.awt.event.*;
import java.io.*;

class Exercise15_9 extends Frame {
   String fileName;
   TextArea content;
   MenuBar mb;
   Menu mFile;
   MenuItem miNew, miOpen, miSaveAs, miExit;

   Exercise15_9(String title) {
       super(title);
       content = new TextArea();
       add(content);

       mb       = new MenuBar();
       mFile    = new Menu("File");

       miNew   = new MenuItem("New");
       miOpen  = new MenuItem("Open");
       miSaveAs = new MenuItem("Save As...");
       miExit   = new MenuItem("Exit");

       mFile.add(miNew);
       mFile.add(miOpen);
       mFile.add(miSaveAs);
       mFile.addSeparator(); // 메뉴 분리선(separator)을 넣는다.
       mFile.add(miExit);

       mb.add(mFile);           // MenuBar에 Menu를 추가한다.
       setMenuBar(mb);          // Frame에 MenuBar를 포함시킨다.

       // 메뉴에 이벤트 핸들러를 등록한다.
       MyHandler handler = new MyHandler();
       miNew.addActionListener(handler);
       miOpen.addActionListener(handler);
       miSaveAs.addActionListener(handler);
       miExit.addActionListener(handler);

       setSize(300, 200);
       setVisible(true);
   }

   // 선택된 파일의 내용을 읽어서 TextArea에 보여주는 메서드
   void fileOpen(String fileName) {
       FileReader fr = null;
       BufferedReader br = null;
       StringWriter sw = null;

       try {
           fr = new FileReader(fileName);
           br = new BufferedReader(fr);
```

```
            sw = new StringWriter();

            String line = "";
//          1. BuffredReader와 FileReader를 이용해서 지정된 파일을 읽는다.
            while ((line=br.readLine())!=null) {
//          2. StringWriter에 출력한다.
                sw.write(line);
                sw.write('\n');   // 개행문자를 출력한다.
            }
//          3. StringWriter의 내용을 content(TextArea)에 보여준다.
            content.setText(sw.toString());
        } catch(IOException e) {
            e.printStackTrace();
        } finally {
            try {
                if(br!=null)
                    br.close();
            } catch(IOException e) {}
        }
    } // fileOpen메서드의 끝

    // TextArea의 내용을 지정된 파일에 저장하는 메서드
    void saveAs(String fileName) {
        FileWriter fw = null;
        BufferedWriter bw = null;
        try {
//        1. BuffredWriter와 FileWriter를 생성한다.
            fw = new FileWriter(fileName);
            bw = new BufferedWriter(fw);
//        2. content에 있는 내용을 가져와서 BufferedWriter에 출력한다.
            bw.write(content.getText());    // TextArea의 내용을 파일에 저장한다.
        } catch (IOException ie) {
            ie.printStackTrace();
        } finally {
            try {
//          3. BufferedWriter를 닫는다.
                if(bw!=null)
                    bw.close();
            } catch(IOException e) {}
        } // try
    } // saveAs메서드의 끝

    public static void main(String args[]) {
        Exercise15_9 mainWin = new Exercise15_9("Text Editor");
    }      // main메서드의 끝

    // 메뉴를 클릭했을 때 메뉴별 처리코드
    class MyHandler implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            String command = e.getActionCommand();

            if(command.equals("New")) {
                content.setText("");
            } else if(command.equals("Open")) {
                FileDialog fileOpen = new FileDialog(Exercise15_9.this,
                                                     "파일열기");
```

```
                fileOpen.setVisible(true);
                fileName = fileOpen.getDirectory() + fileOpen.getFile();
                System.out.println(fileName);
                // 선택된 파일의 내용을 TextArea에 보여준다.
                fileOpen(fileName);
            } else if(command.equals("Save As...")) {
                FileDialog fileSave = new FileDialog(Exercise15_9.this,
                                                     "파일저장",FileDialog.SAVE);
                fileSave.setVisible(true);
                fileName = fileSave.getDirectory() + fileSave.getFile();
                System.out.println(fileName);
                // 현재 TextArea의 내용을 선택된 파일에 저장한다.
                saveAs(fileName);
            } else if(command.equals("Exit")) {
                System.exit(0);       // 프로그램을 종료시킨다.
            }
        }
    } // class MyHandler
} // Exercise15_9클래스의 끝
```

[실행결과]

[해설] 입력의 효율을 높이기 위해서 BufferedReader의 readLine()을 사용했다. readLine()대신 read()를 사용해도 괜찮지만 효율은 떨어진다.

```
            String line = "";
//          1. BuffredReader와 FileReader를 이용해서 지정된 파일을
//             라인단위로 읽는다.
            while ((line=br.readLine())!=null) {
//          2. StringWriter에 출력한다.
                sw.write(line);
                sw.write('\n');   // 개행문자를 출력한다.
            }
//          3. StringWriter의 내용을 content(TextArea)에 보여준다.
            content.setText(sw.toString());
```

readLine()대신 read()를 사용한 코드는 다음과 같다. readLine()과 달리 개행문자를 출력할 필요가 없다.

```
int ch = 0;

while ((ch=br.read())!=-1) {
    sw.write(ch);
}

content.setText(sw.toString());
```

읽어서 바로 화면에 출력할 수도 있지만, 효율도 떨어지고 출력하는 과정도 보이기 때문에 StringWriter에 출력한 다음에 한 번에 모아서 출력하도록 했다.

[연습문제 - 모범답안]

[16-1] ip주소가 192.168.10.100이고 서브넷 마스크(subnet mask)가 255.255.255.0일 때, 네트워크 주소와 호스트 주소를 계산하여 화면에 출력하는 프로그램을 작성하시오. 단, 비트연산자를 사용해서 계산해야 한다.

[실행결과]

```
네트워크 주소:192.168.10.0.
호스트 주소:0.0.0.100.
```

[정답]

[연습문제]/ch16/Exercise16_1.java

```
class Exercise16_1
{
   public static void main(String[] args)
   {
        byte[] ip = {(byte)192,(byte)168,(byte)10,(byte)100};
        byte[] subnet = {(byte)255,(byte)255,(byte)255,(byte)0};

        byte[] nwAddress   = new byte[4];  // 네트워크 주소
        byte[] hostAddress = new byte[4];  // 호스트 주소

        System.out.print("네트워크 주소:");

        for(int i=0; i < ip.length;i++) {
            nwAddress[i] = (byte)(ip[i] & subnet[i]); // &연산을 수행한다.
            System.out.print(nwAddress[i] >=0 ?
                                          nwAddress[i] : nwAddress[i]+256);
            System.out.print(".");
        }

        System.out.println();
        System.out.print("호스트 주소:");

        for(int i=0; i < ip.length;i++) {
            hostAddress[i] = (byte)(ip[i] & ~subnet[i]); // &연산을 수행한다.
            System.out.print(hostAddress[i] >=0 ?
                                       hostAddress[i] : hostAddress[i]+256);
            System.out.print(".");
        }

        System.out.println();
   }
}
```

[해설] 바이트 배열에 ip주소와 서브넷 마스크를 저장한 다음에 반복문을 이용해서 1 byte씩 &연산을 하면 호스트 주소를 얻을 수 있다. 그리고 서브넷 마스크에 '~' 연산을 취한 다음에 ip와 &연산을 취하면 네트워크 주소를 얻을 수 있다.

IP주소

192	168	10	100
11000000	10101000	00001010	01100100
네트워크 주소			호스트 주소

서브넷 마스크(Subnet Mask)

255	255	255	0
11111111	11111111	11111111	00000000

[그림16-1] IP주소(192.168.10.100)와 서브넷 마스크(255.255.255.0)의 2진법 표기

IP주소와 서브넷 마스크를 비트연산자 '&'로 연산하면 IP주소에서 네트워크 주소만을 뽑아낼 수 있다.

	11000000 10101000 00001010 **01100100**
&	11111111 11111111 11111111 00000000
	11000000 10101000 00001010 **00000000**

[그림16-2] IP주소(192.168.10.100)와 서브넷 마스크(255.255.255.0)의 &연산 - 네트워크 주소

IP주소와 서브넷 마스크에 비트연산자 '~'로 연산한 다음에 비트연산자 '&'로 연산하면 IP주소에서 호스트 주소만을 뽑아낼 수 있다. 서브넷 마스크에 '~'연산한 결과는 십진수로 0.0.0.255이고 이진수로는 00000000**00000000**00000000**11111111**이다.

	11000000 10101000 00001010 **01100100**
&	00000000 00000000 00000000 11111111
	00000000 00000000 00000000 **01100100**

[그림16-3] IP주소(192.168.10.100)와 '~서브넷 마스크 (0.0.0.255)'의 &연산 - 호스트 주소

11111111	부호 없는 정수 : 255
11111111	부호 있는 정수 : -1

[그림15-3] 모든 bit의 값이 1인 1 byte(8 bit)

위와 같이 모든 bit의 값이 1인 1 byte의 데이터가 있다고 할 때, 왼쪽에서 첫 번째 비트를 부호로 인식하지 않으면 부호 없는 1 byte가 되어 범위는 0~255이므로 이 값은 최대

값인 255가 되지만, 부호로 인식하는 경우 범위는 -128~127이 되고, 이 값은 0보다 1작은 값인 -1이 된다.

결국 같은 데이터이지만 자바의 자료형인 byte의 범위가 부호 있는 1 byte 정수의 범위인 -128~127이기 때문에 -1로 인식한다는 것이다. 그래서 이 값을 0~255사이의 값으로 변환하려면 256을 더해주어야 한다.

```
for(int i=0; i < ip.length;i++) {
    nwAddress[i] = (byte)(ip[i] & subnet[i]); // &연산을 수행한다.
    System.out.print(nwAddress[i] >=0 ?
                                    nwAddress[i] : nwAddress[i]+256);
    System.out.print(".");
}
```

그래서 부호있는 정수를 부호 없는 정수로 바꾸려면, 양수일 때는 그대로 놔두고 음수일 때만 256을 더해주면 된다. 반대로 부호가 없는 값을 부호가 있는 값으로 바꾸려면, 127보다 큰 값의 경우에만 256을 빼주면 된다.

예를 들어 -1을 부호가 없는 값으로 바꾸려면, 256을 더해서 -1 + 256 = 255가 된다.

```
  byte[] subnet = {(byte)255,(byte)255,(byte)255,(byte)0};
→ byte[] subnet = {      -1,       -1,       -1,      0};
```

0~255사이의 부호없는 정수를 부호있는 정수로 바꾸는 또 다른 방법은 위와 같이 byte로 형변환하는 것이다. 위의 두 코드는 같은 결과를 얻는다.

[16-2] 다음 중 TCP의 특징이 아닌 것은?

a. 전화와 같은 1:1 연결기반의 프로토콜이다.

b. 데이터의 전송순서가 보장된다.

c. UDP보다 전송속도가 빠르다.

d. 데이터의 수신여부를 확인한다.

[정답] c

[해설]

항목	TCP	UDP
연결방식	.연결기반(connection-oriented) - 연결 후 통신(전화기) **- 1:1 통신방식**	.비연결기반(connectionless-oriented) - 연결없이 통신(소포) - 1:1, 1:n, n:n 통신방식
특징	.데이터의 경계를 구분안함 (byte-stream) .신뢰성 있는 데이터 전송 **- 데이터의 전송순서가 보장됨** **- 데이터의 수신여부를 확인함** (데이터가 손실되면 재전송됨) - 패킷을 관리할 필요가 없음 **.UDP보다 전송속도가 느림**	.데이터의 경계를 구분함.(datagram) .신뢰성 없는 데이터 전송 - 데이터의 전송순서가 바뀔 수 있음 - 데이터의 수신여부를 확인안함 (데이터가 손실되어도 알 수 없음) - 패킷을 관리해주어야 함 .TCP보다 전송속도가 빠름
관련 클래스	.Socket .ServerSocket	.DatagramSocket .DatagramPacket .MulticastSocket

[표16-5] TCP와 UDP의 비교

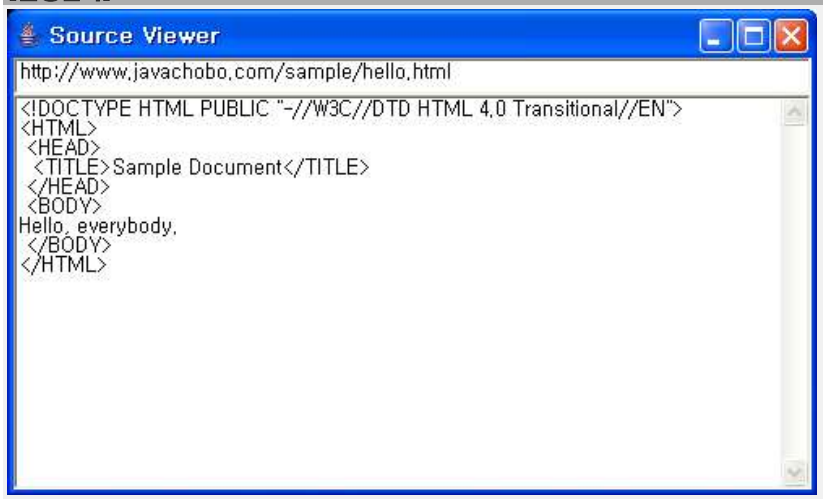
[16-3] TextField에 URL을 입력하고 Enter키를 누르면 해당 페이지의 소스를 보여주는 'Source Viwer'프로그램이다. 예제16-4를 참고해서 (1)에 알맞은 코드를 넣어 완성하시오.

[연습문제]/ch16/Exercise16_3.java

```
import java.net.*;
import java.io.*;
import java.awt.*;
import java.awt.event.*;

class SourceViewer extends Frame {
  TextField tf = new TextField();
  TextArea  ta = new TextArea();

  SourceViewer(String title) {
      super(title);

      add(tf, "North");
      add(ta, "Center");

      tf.addActionListener(new ActionListener(){
          public void actionPerformed(ActionEvent ae) {
              displaySource();
          }
      });

      addWindowListener(new WindowAdapter() {
          public void windowClosing(WindowEvent we) {
              System.exit(0);
          }
      });

      setBounds(500, 200, 500, 300);
      setVisible(true);
  }

  void displaySource() {
      URL url = null;
      BufferedReader input = null;
      String address = tf.getText().trim();
      String line = "";

      ta.setText("");

      try {
          if(!address.startsWith("http://"))
              address = "http://"+address;

              url = new URL(address);
              input =
      new BufferedReader(new InputStreamReader(url.openStream(), "UTF-8"));

          while((line=input.readLine()) !=null) {
              ta.append(line);
              ta.append("\n");
          }
```

```
            input.close();
        } catch(Exception e) {
            ta.setText("유효하지 않은 URL입니다.");
        }
    } // displaySource()

    public static void main(String[] args) {
        SourceViewer mainWin = new SourceViewer("Source Viewer");
    }
}
```

[실행결과]



[해설] URL클래스를 이용해서 해당 URL으로 부터 데이터를 읽어오는 문제이다. 책의 예제 16-4를 참고하면 어렵지 않게 작성할 수 있었을 것이다.

```
            url = new URL(address);
            input =
        new BufferedReader(new InputStreamReader(url.openStream(), "UTF-8"));

            while((line=input.readLine()) !=null) {
                ta.append(line);
                ta.append("\n");
            }
```

좀 더 설명을 해야 할 부분이 있다면 아래의 코드인데, 우리가 읽어오고자 하는 내용이 HTML페이지(텍스트)이기 때문에 openStream()이 반환하는 InputStream을 InputStreamReader를 이용해서 문자기반의 스트림(BufferedReader)으로 변환하였다. 만일 이미지와 같은 바이너리파일을 읽어올 때는 InputStreamReader를 이용할 필요없이 바로 BufferedInputStream을 생성하면 된다.

```
url = new URL(address);
input =
    new BufferedReader(new InputStreamReader(url.openStream(), "UTF-8"));
```

InputStreamReader을 이용해서 URL의 텍스트데이터를 읽어올 때 주의해야할 점은 인코딩을 지정해주어야 한다는 것이다. 인코딩을 지정해주지 않으면 OS에서 사용하는 인코딩, 예를 들어 XP의 경우 MS949로, 지정된다.

```
url = new URL(address);
input =
    new BufferedReader(new InputStreamReader(url.openStream(), "UTF-8"));
```

InputStreamReader를 생성할 때 지정해준 인코딩하고 읽어올 URL의 텍스트의 인코딩이 일치하지 않으면 내용이 바르게 보이지 않을 수 있다. 특히 영어가 아닌 한글이나 일본어 등으로 작성된 페이지들이 그렇다.

다음은 어떤 HTML의 앞부분인데 meta태그의 content속성에 보면 인코딩(캐릭터셋, charset)을 알려준다. 페이지를 보여주기 전에 HTML의 앞부분만 읽어서 인코딩을 알아낸 다음에 화면에 보여주면 페이지의 내용을 온전히 보여줄 수 있을 것이다.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
...
```

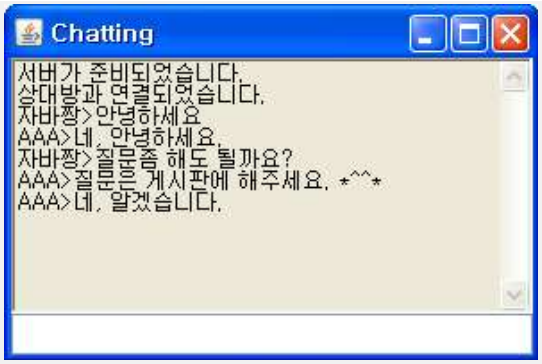
[16-4] 다음의 코드는 TCP통신을 하는 예제16-6, 16-7을 결합하여 GUI채팅프로그램을 작성한 것이다. (1)~(4)에 알맞은 코드를 넣어 프로그램을 완성하시오.

[연습문제]/ch16/ChatServer.java

```
import java.net.*;
import java.io.*;
import java.awt.*;
import java.awt.event.*;

class ChatServer extends Frame {
   String nickname = "";

   DataOutputStream out;
   DataInputStream in;

   Panel p = new Panel();
   TextArea ta = new TextArea();
   TextField tf = new TextField();

   ChatServer(String nickname) {
        super("Chatting");
        this.nickname = nickname;

        p.setLayout(new BorderLayout());
        p.add(tf, "Center");

        add(ta, "Center");
        add(p, "South");

        addWindowListener(new WindowAdapter(){
            public void windowClosing(WindowEvent e) {
                System.exit(0);
            }
        });
        EventHandler handler = new EventHandler();
        ta.addFocusListener(handler);
        tf.addFocusListener(handler);
        tf.addActionListener(handler);

        ta.setEditable(false);
        setBounds(200, 200, 300, 200);
        setVisible(true);
        tf.requestFocus();
   }

   void startServer() {
        ServerSocket serverSocket = null;
        Socket socket = null;

        try {
//          1. 서비소켓을 생성하여 7777번 포트와 결합시킨다.
            serverSocket = new ServerSocket(7777);
//          2. ta에 "서버가 준비되었습니다."라고 보여준다.
            ta.setText("서버가 준비되었습니다.");
//          3. 상대방의 연결을 기다린다.
            socket = serverSocket.accept();
//          4. ta에 "상대방과 연결되었습니다."라고 보여준다.
            ta.append("\r\n" +"상대방과 연결되었습니다.");
//          5. 연결된 상대방 소켓의 입력스트림과 출력스트림을 얻어온다.
```

```
            in = new DataInputStream(socket.getInputStream());
            out = new DataOutputStream(socket.getOutputStream());

//          6. 반복문을 이용해서 입력스트림이 null이 아닌 동안
            while(in!=null) {
                try {
//                  입력스트림으로부터 데이터를 읽어서 ta에 append한다.
                    String msg = in.readUTF();
                    ta.append("\r\n" + msg);
                } catch(IOException e) {}
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        if(args.length != 1) {
            System.out.println("USAGE : java ChatServer NICKNAME");
            System.exit(0);
        }

        ChatServer chatWin = new ChatServer(args[0]);
        chatWin.startServer();
    } // main

    class EventHandler extends FocusAdapter implements ActionListener {
        public void actionPerformed(ActionEvent ae) {
            String msg = tf.getText();

            if("".equals(msg)) return;

            if(out!=null) {
                try {
                    out.writeUTF(nickname+">"+msg);
                } catch(IOException e) {}
            }

            ta.append("\r\n" + nickname +">"+ msg);
            tf.setText("");
        }

        public void focusGained(FocusEvent e) {
            tf.requestFocus();
        }
    } // class EventHandler
} // class
```

[실행결과]



[연습문제]/ch16/ChatClient.java

```
import java.net.*;
import java.io.*;
import java.awt.*;
import java.awt.event.*;

class ChatClient extends Frame {
   String nickname = "";
   String serverIp = "";
   int serverPort = 0;

   DataOutputStream out;
   DataInputStream in;

   Panel p = new Panel();
   TextArea ta = new TextArea();
   TextField tf = new TextField();

   ChatClient(String nickname, String serverIp, String serverPort) {
        super("Chatting with " + serverIp +":" + serverPort);
        this.nickname = nickname;
        this.serverIp = serverIp;
        this.serverPort = Integer.parseInt(serverPort);

        setBounds(600, 200, 300, 200);

        p.setLayout(new BorderLayout());
        p.add(tf, "Center");

        add(ta, "Center");
        add(p, "South");

        addWindowListener(new WindowAdapter(){
            public void windowClosing(WindowEvent e) {
                System.exit(0);
            }
        });
        EventHandler handler = new EventHandler();
        ta.addFocusListener(handler);
        tf.addFocusListener(handler);
        tf.addActionListener(handler);
```

```
        ta.setEditable(false);

        setVisible(true);
        tf.requestFocus();
    }

    void startClient() {
        try {
//          1. 소켓을 생성하여 serverIp의 serverPort에 연결한다.
            Socket socket = new Socket(serverIp, serverPort);
//          2. ta에 "상대방과 연결되었습니다."라고 보여준다.
            ta.setText("상대방과 연결되었습니다.");
//          3. 연결된 상대방 소켓의 입력스트림과 출력스트림을 얻어온다.
            in = new DataInputStream(socket.getInputStream());
            out = new DataOutputStream(socket.getOutputStream());
//          4. 반복문을 이용해서 입력스트림이 null이 아닌 동안
            while(in!=null) {
                try {
//                  입력스트림으로부터 데이터를 읽어서 ta에 append한다.
                    String msg = in.readUTF();
                    ta.append("\r\n" + msg);
                } catch(IOException e) {}
            }
        } catch(ConnectException ce) {
            ta.setText("상대방과 연결할 수 없습니다.");
            ce.printStackTrace();
        } catch(IOException ie) {
            ie.printStackTrace();
        } catch(Exception e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        if(args.length != 3) {
            System.out.println("USAGE : java ChatClient NICKNAME SERVER_IP
SERVER_PORT");
            System.exit(0);
        }

        ChatClient chatWin = new ChatClient(args[0],args[1],args[2]);
        chatWin.startClient();
    } // main

    class EventHandler extends FocusAdapter implements ActionListener {
        public void actionPerformed(ActionEvent ae) {
            String msg = tf.getText();

            if("".equals(msg)) return;

            if(out!=null) {
                try {
                    out.writeUTF(nickname+">"+msg);
                } catch(IOException e) {}
            }

            ta.append("\r\n" + nickname +">"+ msg);
```

```
            tf.setText("");
        }

        public void focusGained(FocusEvent e) {
            tf.requestFocus();
        }
    } // class EventHandler
} // class
```

[실행결과]