연습문제 풀이

ver.20191216v1

답을 보기 원하시면 해당 예제의 번호로 검색하시면 답을 찾으실 수 있습니다. 연습문제 중에 좀 어려운 것들도 있는데, 책 안에 답이 있는 경우가 많이 있습니다. 문제가 안 풀릴 때는 책을 다시 한 번 살펴보세요. 복습도 되고 1석2조입니다. 문제가 바로 답을 보기 보다는 제가 운영하는 카페 '코드초보스터디(http://codechobo.com)에 오셔서 질문하시면 힌트를 드리겠습니다.

연습문제를 못 푼다고 해서 큰 문제가 되는 것은 아닙니다. 고민하고 생각하고 자신의 생각을 정리하는 방법을 배우는 것이 중요한 것이죠. 프로그래밍이라는 것은 반드시 창의적일 필요는 없다고 생각합니다. 기존의 것을 잘 활용하는 것으로도 충분합니다. 문제의 답을 보고, '아, 이렇게 하는 구나.'라고 이해하시고 비슷한 상황에서 응용하실 수 있으면 됩니다.

이 문서는 e-book의 특성을 살려서 시간되는 대로 틈틈이 업데이트할 예정입니다. 첫 번째 페이지의 버전을 꼭 확인해주세요. 그리고 좋은 문제나 오탈자, 문제점 등이 있으면 castello@naver.com으로 메일 주세요. 단, 메일로 질문은 사절합니다. 질문은 카페게시판을 이용해주세요.

마지막으로 이 문서는 상업적인 용도가 아닌 경우에는 얼마든지 자유롭게 배포하실 수 있습니다. 이 문서가 Java를 공부하시는 많은 분들에게 도움이 되길 바랍니다.

감사합니다.

2019년 12월 16일

남 궁 성

[개정이력]

2019. 12. 16 - Java의 정석 3판의 연습문제를 바탕으로 처음으로 작성.
(**ver.20191216v1**)

[연습문제 - 모범답안]

[2-1] 다음 표의 빈 칸에 8개의 기본형(primitive type)을 알맞은 자리에 넣으시오.

크 기 / 종 류	1 byte	2 byte	4 byte	8 byte
논리형	boolean			
문자형		char		
정수형	byte	short	**int**	long
실수형			float	**double**

[2-2] 다음 중 키워드가 아닌 것은?(모두 고르시오)

1. if
2. True
3. NULL
4. Class
5. System

[정답] 2,3,4,5

[해설] Java에서는 대소문자를 구별하기 때문에 true는 키워드이지만 True는 키워드가 아니다. 다음은 Java에서 사용하는 키워드이다.

abstract	default	if	package	this
assert	do	goto	private	throw
boolean	double	implements	protected	throws
break	else	import	public	transient
byte	enum	instanceof	return	true
case	extends	int	short	try
catch	false	interface	static	void
char	final	long	strictfp	volatile
class	finally	native	super	while
const	float	new	switch	
continue	for	null	synchronized	

[2-3] char타입의 변수에 저장될 수 있는 정수 값의 범위는? (10진수로 적으시오)

[정답] 0~65535
[해설] char는 2 byte(2*8=16bit)이므로 '2의 16제곱' 개의 값을 표현할 수 있다.
2의 16제곱은 65536개이며, 0을 포함해야하므로 0~65535(모두 65536개)가 char범위가 된다.

[2-4] 다음중 변수를 잘못 초기화 한 것은? (모두 고르시오)

1. byte b = 256; // byte의 범위(-128~127)를 넘는 값으로 초기화 할 수 없음.
2. char c = ''; // char는 반드시 한 개의 문자를 지정해야함
3. char answer = 'no'; // char에 두 개의 문자를 저장할 수 없음.
4. float f = 3.14 // 3.14는 3.14d의 생략된 형태. 접미사f를 붙이거나 형변환필요
5. double d = 1.4e3f; // double(8byte)에 float값(4byte)을 넣는 것이므로 OK

[정답] 1,2,3,4

[해설] 접미사가 있는 자료형은 long, float, double 모두 세 개의 자료형이며, 접미사는 대소문자를 구별하지 않는다. double은 접미사를 생략할 수 있으므로 float리터럴에는 반드시 접미사를 붙여야한다.

[2-5] 다음의 문장에서 리터럴, 변수, 상수, 키워드를 적으시오.

```
int i = 100;
long l = 100L;
final float PI = 3.14f;
```

- 리터럴 : 100, 100L, 3.14f
- 변수 : i, l
- 키워드 : int, long, final, float
- 상수 : PI

[2-6] 다음 중 기본형(primitive type)이 아닌 것은?

1. int
2. **Byte**
3. double
4. boolean

[정답] 2

[해설] 기본형은 boolean, byte, short, char, int, long, float, double 모두 8개이다. 그 외의 타입은 모두 참조형(reference type)이다.

[2-7] 다음 문장들의 출력결과를 적으세요. 오류가 있는 문장의 경우, 괄호 안에 '오류'라고 적으시오.

```
System.out.println( "1" + "2" )   → (12          )
System.out.println(true + "" )    → (true        )
System.out.println( 'A' + 'B')    → (131         )
System.out.println('1' + 2)       → (51          )
System.out.println('1' + '2')     → (99          )
System.out.println('J' + "ava" )  → (Java        )
System.out.println(true + null)   → (오류        )
```

[해설] 문자열과 덧셈연산을 하면 그 결과는 항상 문자열이 된다.

문자열 + **any type** → 문자열 + **문자열** → **문자열**
any type + 문자열 → **문자열** + 문자열 → **문자열**

```
"" + 7  →  "" + "7"  →  "7"     // 빈 문자열을 더해서 숫자를 문자열로 변환한다.
7 + ""  →  "7" + ""  →  "7"

7 + " " →  "7" + " "  →  "7 "
" " + 7 →  " " + "7"  →  " 7"

7 + "7"  →  "7" + "7"  →  "77"

7 + 7 + ""  →  14 + ""  →  "14" + ""  →  "14"
"" + 7 + 7  →  "7" + 7  →  "7" + "7"  →  "77"

true + "" → "true" + "" → "true"
null + "" → "null" + "" → "null"
```

하지만 문자와 문자의 덧셈연산의 결과는 int형 정수값이 된다. 왜냐하면 int형보다 작은 타입(byte, char, short)은 int형으로 변환된 후에 덧셈연산이 진행되기 때문이다.
'A'+'B'의 경우, char + char → int + int → int의 과정을 통해 최종결과는 int형 정수값이 된다.(3장에서 자세히 설명)

```
'A'+'B' → 65 + 66 → 131      'A'와 'B'의 문자코드의 값은 각각 65와 66이다.
'1'+2   → 49 + 2  → 51       '1'의 문자코드의 값은 49이다.
'1'+'2' → 49 + 50 → 99
```

[2-8] 아래는 변수 x, y, z의 값을 서로 바꾸는 예제이다. 결과와 같이 출력되도록 (1)에 알맞은 코드를 넣으시오.

[연습문제]/ch2/Exercise2_8.java

```
public class Exercise2_8 {
   public static void main(String[] args) {
        int x = 1;
        int y = 2;
        int z = 3;

        int tmp = x;
        x = y;
        y = z;
        z = tmp;

        System.out.println("x=" + x);
        System.out.println("y=" + y);
        System.out.println("z=" + z);
   }
}
```

[실행결과]

```
x=2
y=3
z=1
```

[3-1] 다음 중 형변환을 생략할 수 있는 것은? (모두 고르시오)

```
byte b = 10;
char ch = 'A';
int i = 100;
long l = 1000L;
```

1. b = (byte)i; // int(4byte) → byte(1byte)이므로 반드시 형변환 필요
2. ch = (char)b; // byte(1byte) → char(2byte)이지만 범위가 달라서 형변환 필요
3. short s = (short)ch; // char,short은 2byte이지만 범위가 달라서 형변환 필요
4. **float f = (float)l; // float(4byte)의 범위가 long(8byte)보다 커서 생략가능**
5. **i = (int)ch; // char(2 byte) → int(4byte)이므로 생략가능**

[정답] 4, 5

[3-2] 다음 연산의 결과를 적으시오.

[연습문제]/ch3/Exercise3_2.java

```
class  Exercise3_2 {
   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. 논리연산자 '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.(이항

연산자는 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.

[3-3] 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_3.java

```
class  Exercise3_3 {
   public static void main(String[] args) {
        int num = 456;
        System.out.println(num/100*100);
   }
}
```

[실행결과]

```
400
```

[정답] num/100*100

[해설] 나눗셈연산자는 반올림을 하지 않고 버림을 한다. 이 성질을 이용한 문제이다. 어떤 숫자를 100으로 나누면 십의 자리와 일의 자리가 잘리고 백의 자리가 일의 자리가 된다. 즉, 456을 100으로 나누면 4가 되는 것이다. 여기에 다시 100을 곱하면 400이 된다. 10의 n제곱을 나눴다가 다시 곱하면 간단하게 아래 자리의 수를 0으로 만들 수 있다.

456 / 100 -> 4
4 * 100 -> 400

111 / 100 -> 1
1 * 100 -> 100

[3-4] 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_4.java

```
class  Exercise3_4 {
   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.

[3-5] 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_5.java

```
class  Exercise3_5 {
   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.

[3-6] 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_6.java

```
class  Exercise3_6 {
   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

[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 powerOn이 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 powerOn이 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] 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_7.java

```
class Exercise4_7 {
  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

[4-8] Math.random()을 이용해서 1부터 6사이의 임의의 정수를 변수 value에 저장하는 코드를 완성하라. (1)에 알맞은 코드를 넣으시오.

[연습문제]/ch4/Exercise4_8.java

```
class  Exercise4_8 {
   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.

[4-9] 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_9.java

```
class  Exercise4_9 {
   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] 문제4-7에서처럼 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.

[4-10] 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_10.java

```
class Exercise4_10
{
     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.

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

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

[정답] 4, 5

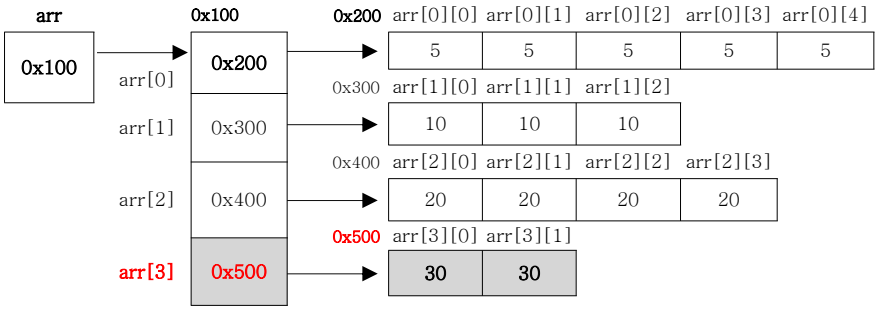
[해설] **4**. int[] arr = new int[]{1,2,3,4,5}에서는 대괄호[]안에 배열의 크기를 지정할 수 없다. 괄호{}안의 데이터의 개수에 따라 자동적으로 결정되기 때문이다.

[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]

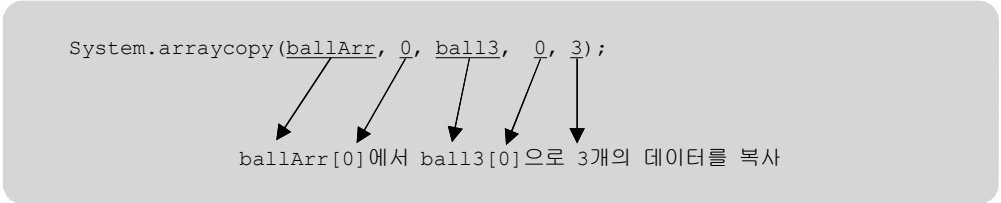
(1)

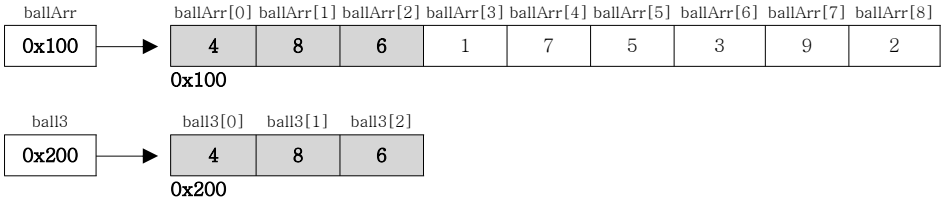
```
tmp = ballArr[i];
ballArr[i] = ballArr[j];
ballArr[j] = tmp;
```

(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] 단어의 글자위치를 섞어서 보여주고 원래의 단어를 맞추는 예제이다. 실행결과와 같이 동작하도록 예제의 빈 곳을 채우시오.

[연습문제5-6]/ch5/Excercise5_6.java

```
import java.util.Scanner;

class Exercise5_6 {
   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] 예제5-4와 10을 응용한 문제이다. 간단한 문제이므로 설명은 생략한다.

[6-1] 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;
}
```

[6-2] 다음과 같은 실행결과를 얻도록 Student클래스에 생성자와 info()를 추가하시오.

[연습문제]/ch6/Exercise6_2.java

```
class Exercise6_2 {
   public static void main(String args[]) {
        Student s = new Student("홍길동",1,1,100,60,76);

        String str = s.info();
        System.out.println(str);
   }
}

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;
   }

   public String info() {
        return name
              +","+ban
              +","+no
              +","+kor
              +","+eng
              +","+math
              +","+(kor+eng+math)
              +","+((int)((kor+eng+math) / 3f * 10 + 0.5f) / 10f)
              ;
   }
}
```

[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.

[6-3] 문제6-1에서 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_3.java

```
class Exercise6_2 {
   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;

   Student() {}  // 기본 생성자

   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]

```
이름:홍길동
총점:236
평균:78.7
```

[정답]

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

    Student() {}  // 기본 생성자

    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()
                ;
    }
}
```

[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
```

[6-4] 두 점의 거리를 계산하는 getDistance()를 완성하시오.

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

[연습문제]/ch6/Exercise6_4.java

```
class Exercise6_4 {
    // 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));
}
```

[6-5] 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

[6-6] 문제6-4에서 작성한 클래스메서드 getDistance()를 MyPoint클래스의 인스턴스메서드로 정의하시오.

[연습문제]/ch6/Exercise6_6.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) { // 인스턴스 메서드
        return Math.sqrt((x-x1)*(x-x1) + (y-y1)*(y-y1)); // x, y는 인스턴스 변수
   }
}

class Exercise6_6 {
   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.

[6-7] 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.

[6-8] 다음 중 생성자에 대한 설명으로 옳지 않은 것은? (모두 고르시오)

1. 모든 생성자의 이름은 클래스의 이름과 동일해야한다.
2. **생성자는 객체를 생성하기 위한 것이다.**
3. 클래스에는 생성자가 반드시 하나 이상 있어야 한다.
4. 생성자가 없는 클래스는 컴파일러가 기본 생성자를 추가한다.
5. **생성자는 오버로딩 할 수 없다.**

[정답] 2, 5

[해설]

2. 생성자는 객체를 생성하기 위한 것이다.

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

5. 생성자는 오버로딩 할 수 없다.

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

[6-9] 다음 중 this에 대한 설명으로 맞지 않은 것은? (모두 고르시오)

1. 객체 자신을 가리키는 참조변수이다.
2. **클래스 내에서라면 어디서든 사용할 수 있다.** → 인스턴스메서드에서만 사용가능
3. 지역변수와 인스턴스변수를 구별할 때 사용한다.
4. 클래스 메서드 내에서는 사용할 수 없다.

[정답] 2

[해설]

2. 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.

[6-10] 다음 중 오버로딩이 성립하기 위한 조건이 아닌 것은? (모두 고르시오)

1. 메서드의 이름이 같아야 한다.
2. 매개변수의 개수나 타입이 달라야 한다.
3. **리턴타입이 달라야 한다.**
4. **매개변수의 이름이 달라야 한다.**

[정답] 3, 4

[해설]

3. 리턴타입이 달라야 한다.

→ 리턴타입은 오버로딩에 영향을 주지 못한다.

4. 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.)

[6-11] 다음 중 아래의 add메서드를 올바르게 오버로딩 한 것은? (모두 고르시오)

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

1. long add(int x, int y) { return x+y;}
2. long add(long a, long b) { return a+b;}
3. int add(byte a, byte b) { return a+b;}
4. int add(long a, int b) { return (int)(a+b);}

[정답] 2, 3, 4

[해설] 2, 3, 4는 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.)

[6-12] 다음 중 초기화에 대한 설명으로 옳지 않은 것은? (모두 고르시오)

1. 멤버변수는 자동 초기화되므로 초기화하지 않고도 값을 참조할 수 있다.
2. 지역변수는 사용하기 전에 반드시 초기화해야 한다.
3. **초기화 블럭보다 생성자가 먼저 수행된다. → 초기화 블럭이 먼저 수행된다.**
4. 명시적 초기화를 제일 우선적으로 고려해야 한다.
5. **클래스변수보다 인스턴스변수가 먼저 초기화된다.→ 클래스변수가 먼저 초기화됨**

[정답] 3, 5

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

[6-13] 다음중 인스턴스변수의 초기화 순서가 올바른 것은?

1. **기본값-명시적초기화-초기화블럭-생성자**
2. 기본값-명시적초기화-생성자-초기화블럭
3. 기본값-초기화블럭-명시적초기화-생성자
4. 기본값-초기화블럭-생성자-명시적초기화

[정답] 1

[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

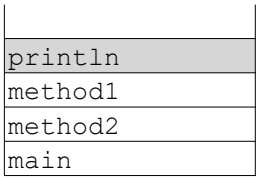
[6-14] 다음 중 지역변수에 대한 설명으로 옳지 않은 것은? (모두 고르시오)

1. **자동 초기화되므로 별도의 초기화가 필요없다.**
2. 지역변수가 선언된 메서드가 종료되면 지역변수도 함께 소멸된다.
3. 매서드의 매개변수로 선언된 변수도 지역변수이다.
4. 클래스변수나 인스턴스변수보다 메모리 부담이 적다.
5. **힙(heap)영역에 생성되며 가비지 컬렉터에 의해 소멸된다.**

[정답] 1, 5

[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.

[6-15] 호출스택이 다음과 같은 상황일 때 옳지 않은 설명은? (모두 고르시오)



1. 제일 먼저 호출스택에 저장된 것은 main메서드이다.
2. **println메서드를 제외한 나머지 메서드들은 모두 종료된 상태이다.**
3. method2메서드를 호출한 것은 main메서드이다.
4. println메서드가 종료되면 method1메서드가 수행을 재개한다.
5. main-method2-method1-println의 순서로 호출되었다.
6. 현재 실행중인 메서드는 println 뿐이다.

[정답] 2

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

[6-16] 다음 코드의 실행 결과를 예측하여 적으시오.

[연습문제]/ch6/Exercise6_16.java

```
class Exercise6_16
{
   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]

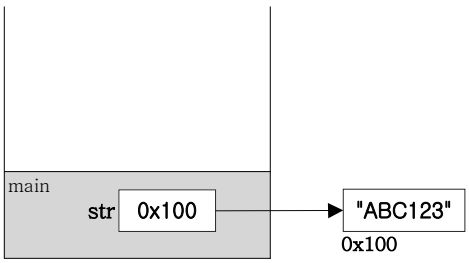
```
ABC123
After change:ABC123
```

[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.

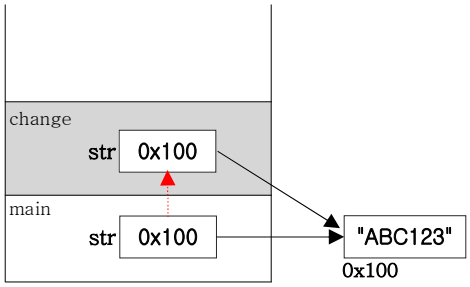
처음에 문자열을 참조변수 str에 저장하면 아래와 같은 그림이 된다.

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



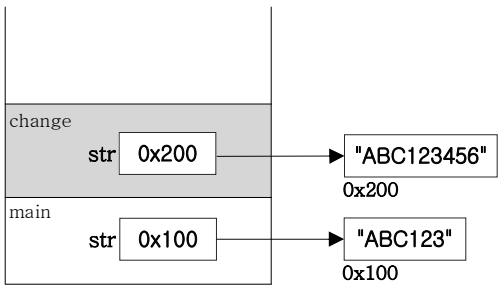
그 다음에 메서드 change를 호출하면서 참조변수 str을 넘겨주면, 메서드 change의 지역변수 str에 주소값 0x100이 저장된다. 이제 메서드 change의 지역변수 str도 문자열 "ABC123"을 참조하게 된다. 이 두 참조변수는 이름은 같지만 분명히 다른 변수 이다. 서로 다른 영역에 존재하기 때문에 이름이 같아도 상관없는 것이다.

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



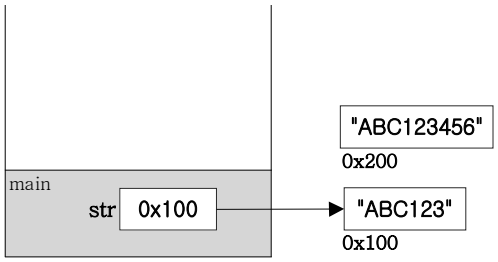
메서드 change에서는 넘겨받은 문자열의 뒤에 "456"을 붙인다. 문자열은 내용을 변경할 수 없기 때문에 덧셈연산을 하면 새로운 문자열이 생성되고 새로운 문자열의 주소가 변수 str에 저장된다.

```
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);
```



[6-17] 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_17.java

```
class Exercise6_17
{
   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.

[6-18] 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 빈 문자열 "" 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_18.java

```
class Exercise6_18 {
    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;
    }
```

[6-19] Tv클래스를 주어진 로직대로 완성하시오. 완성한 후에 실행해서 주어진 실행결과와 일치하는지 확인하라.

[참고] 코드를 단순히 하기 위해서 유효성검사는 로직에서 제외했다.

[연습문제]/ch6/Exercise6_19.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_19 {
    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://codechobo.com의 게시판에 올려주길 바란다.

[6-20] 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_20.java

```
class Exercise6_20{
    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;
```

[6-21] 다음과 같이 정의된 메서드를 작성하고 테스트하시오.

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

[연습문제]/ch6/Exercise6_21.java

```
class Exercise6_21
{
    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.
    }
}
```

[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 3이라면 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 문제6-17에서 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] 다음의 코드는 컴파일하면 에러가 발생한다. 그 이유를 설명하고 에러를 수정하기 위해서는 코드를 어떻게 바꾸어야 하는가?

[연습문제]/ch7/Exercise7_3.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_3 {
   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.

[7-4] MyTv클래스의 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_4.java

```
class MyTv {
   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_4 {
   public static void main(String args[]) {
       MyTv t = new MyTv();

       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.

[7-5] 문제7-4에서 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_5.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_5 {
   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.
}
```

[7-6] Outer클래스의 내부 클래스 Inner의 멤버변수 iv의 값을 출력하시오.

[연습문제]/ch10/Exercise7_6.java

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

class Exercise7_6 {
   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.

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

[연습문제]/ch10/Exercise7_7.java

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

class Exercise7_7 {
   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.

[7-8] 다음과 같은 실행결과를 얻도록 (1)~(4)의 코드를 완성하시오.

[연습문제]/ch10/Exercise7_8.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_8 {
   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.

[7-9] 아래의 EventHandler를 익명 클래스(anonymous class)로 변경하시오.

[연습문제]/ch7/Exercise7_9.java

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

class Exercise7_9
{
   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_9_2.java

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

class Exercise7_9_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
}
```

[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

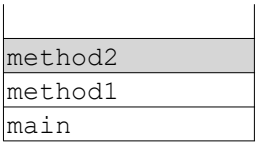
[8-2] 다음은 실행도중 예외가 발생하여 화면에 출력된 내용이다. 이에 대한 설명 중 옳지 않은 것은?

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

1. 위의 내용으로 예외가 발생했을 당시 호출스택에 존재했던 메서드를 알 수 있다.
2. 예외가 발생한 위치는 method2 메서드이며, ExceptionEx18.java파일의 12번째 줄이다.
3. 발생한 예외는 ArithmeticException이며, 0으로 나누어서 예외가 발생했다.
4. **method2메서드가 method1메서드를 호출하였고 그 위치는 ExceptionEx18.java파일의 8번째 줄이다.**

[정답] 4

[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 {}
```

1. void add(int a, int b) throws InvalidNumberException, NotANumberException {}
2. void add(int a, int b) throws InvalidNumberException {}
3. void add(int a, int b) throws NotANumberException {}
4. **void add(int a, int b) throws Exception {}**
5. **void add(int a, int b) throws NumberException {}**

[정답] 4, 5

[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] 아래의 코드가 수행되었을 때의 실행결과를 적으시오.

[연습문제]/ch8/Exercise8_4.java

```
class Exercise8_4 {
   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.

[8-5] 아래의 코드가 수행되었을 때의 실행결과를 적으시오.

[연습문제]/ch8/Exercise8_5.java

```
class Exercise8_5 {
   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.

[8-6] 아래의 코드가 수행되었을 때의 실행결과를 적으시오.

[연습문제]/ch8/Exercise8_6.java

```
class Exercise8_6 {
   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.

[8-7] 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_7.java:16)
```

[연습문제]/ch8/Exercise8_7.java

```
import java.util.*;

class Exercise8_7
{
     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;
}
```

[8-8] 아래의 코드가 수행되었을 때의 실행결과를 적으시오.

[연습문제]/ch8/Exercise8_8.java

```
class Exercise8_8 {
   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);
}
```

[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 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_3.java

```
class Exercise9_3 {
   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
```

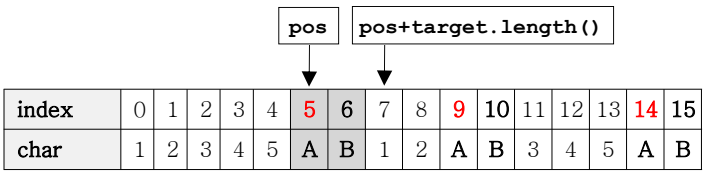
[해설] indexOf()는 지정된 문자열을 찾아서 그 위치를 알려준다. 만일 찾지 못한다면 -1을 반환한다. 문자열 "12345AB12AB345AB"에서 문자열 "AB"를 indexOf()로 찾으면 그 첫 번째 위치는 아래의 표에서 알 수 있듯이 5이다. 두 번째는 9, 세 번째는 14이다.

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();
        }
```

[9-4] 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_4.java

```
class Exercise9_4 {
   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;
```

[9-5] 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_5.java

```
class Exercise9_5 {
   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
```

[9-6] 다음은 화면으로부터 전화번호의 일부를 입력받아 일치하는 전화번호를 주어진 문자열 배열에서 찾아서 출력하는 프로그램이다. 알맞은 코드를 넣어 프로그램을 완성하시오.

[Hint] Pattern, Matcher클래스를 사용하라.

[연습문제]/ch9/Exercise9_6.java

```
import java.util.*;
import java.util.regex.*;

class Exercise9_6 {
   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);
    }
}
```

[10-1] Calendar클래스와 SimpleDateFormat클래스를 이용해서 2020년의 매월 두 번째 일요일의 날짜를 출력하시오.

[실행결과]

```
2020-01-12은 2번째 일요일입니다.
2020-02-09은 2번째 일요일입니다.
2020-03-08은 2번째 일요일입니다.
2020-04-12은 2번째 일요일입니다.
2020-05-10은 2번째 일요일입니다.
2020-06-14은 2번째 일요일입니다.
2020-07-12은 2번째 일요일입니다.
2020-08-09은 2번째 일요일입니다.
2020-09-13은 2번째 일요일입니다.
2020-10-11은 2번째 일요일입니다.
2020-11-08은 2번째 일요일입니다.
2020-12-13은 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(2020, 0, 1); // cal의 날짜를 2020년 1월 1일로 설정

        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 알아

내야한다. 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 E요일입니다.").format(d));
```

[10-2] 화면으로부터 날짜를 "2017/05/11"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:2017/05/11)
>>2009-12-12
날짜를 yyyy/MM/dd의 형태로 입력해주세요.(입력예:2017/05/11)
>>2009/12/12
입력하신 날짜는 토요일입니다.
```

[정답]

[연습문제]/ch10/Exercise10_2.java

```
import java.util.*;
import java.text.*;

class Exercise10_2 {
   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:2017/05/11)");

            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:2017/05/11)");
    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.

[10-3] 어떤 회사의 월급날이 매월 21일이다. 두 날짜 사이에 월급날이 몇 번있는지 계산해서 반환하는 메서드를 작성하고 테스트 하시오.

[연습문제]/ch10/Exercise10_3.java

```
import java.util.*;
import java.text.*;

class Exercise10_3 {
   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(2020,0,1);
        toCal.set(2020,0,1);
        printResult(fromCal, toCal);

        fromCal.set(2020,0,21);
        toCal.set(2020,0,21);
        printResult(fromCal, toCal);

        fromCal.set(2020,0,1);
        toCal.set(2020,2,1);
        printResult(fromCal, toCal);

        fromCal.set(2020,0,1);
        toCal.set(2020,2,23);
        printResult(fromCal, toCal);

        fromCal.set(2020,0,23);
        toCal.set(2020,2,21);
        printResult(fromCal, toCal);

        fromCal.set(2021,0,22);
        toCal.set(2020,2,21);
        printResult(fromCal, toCal);
    }
}
```

[실행결과]

```
2020-01-01 ~ 2020-01-01:0
2020-01-21 ~ 2020-01-21:1
2020-01-01 ~ 2020-03-01:2
2020-01-01 ~ 2020-03-23:3
2020-01-23 ~ 2020-03-21:2
2021-01-22 ~ 2020-03-21:0
```

[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--;
```

[10-4] 자신이 태어난 날부터 지금까지 며칠이 지났는지 계산해서 출력하시오.

[실행결과]

```
birth day=2000-01-01
today    =2016-01-29
5872 days
```

[정답] 예제10-3과 예제10-8, Date와 Calendar간의 변환을 응용하면 된다.

[연습문제]/ch10/Exercise10_4.java

```
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;

class Exercise10_4 {
   public static void main(String[] args) {
        Calendar date1 = Calendar.getInstance();
        Calendar date2 = Calendar.getInstance();

        date1.set(2000, 0, 1);  // 2000년 1월  1일로 날짜를 설정한다.
        date2.set(2016, 0, 29); // 2016년 1월  29일로 날짜를 설정한다.

        SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd");

        System.out.println("birth day="
                           + df.format(new Date(date1.getTimeInMillis())));
        System.out.println("today     ="
                           + df.format(new Date(date2.getTimeInMillis())));

        long difference =
                (date2.getTimeInMillis() - date1.getTimeInMillis())/1000;
        System.out.println(difference/(24*60*60)
                +" days"); // 1일 = 24 * 60 * 60
   }
}
```

[11-1] 다음 코드의 실행결과를 적으시오.

[연습문제]/ch11/Exercise11_1.java

```
import java.util.*;

class Exercise11_1 {
   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.
```

[11-2] 다음 중 ArrayList에서 제일 비용이 많이 드는 작업은? 단, 작업도중에 ArrayList의 크기 변경이 발생하지 않는다고 가정한다.

1. 첫 번째 요소 삭제
2. 마지막 요소 삭제
3. 마지막에 새로운 요소 추가
4. 중간에 새로운 요소 추가

[정답] 1

[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.

[11-3] 다음에 제시된 Student클래스가 Comparable인터페이스를 구현하도록 변경해서 이름(name)이 기본 정렬기준이 되도록 하시오.

[연습문제]/ch11/Exercise11_3.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_3 {
   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_3_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_3_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());
   }
}
```

[11-4] 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_4.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_4 {
    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;
}
```

[11-5] 다음은 SutdaCard클래스를 HashSet에 저장하고 출력하는 예제이다. HashSet에 중복된 카드가 저장되지 않도록 SutdaCard의 hashCode()를 알맞게 오버라이딩하시오.

[Hint] String클래스의 hashCode()를 사용하라.

[연습문제]/ch11/Exercise11_5.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_5 {
   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.

[11-6] 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_6.java

```
import java.util.*;

class Exercise11_6 {
   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_6_2.java

```
import java.util.*;

class Exercise11_6_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)
```

[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;
    }
}
```

1. Box<Object> b = new Box<String>();
2. Box<Object> b = (Object)new Box<String>();
3. new Box<String>().setItem(new Object());
4. new Box<String>().setItem("ABC");

[정답] 1, 2, 3

[해설]

1. Box<**Object**> b = new Box<**String**>(); // 에러. 대입된 타입이 반드시 같아야 한다.
2. **Box<Object>** b = **(Object)**new Box<String>(); // 에러. Object타입을 Box<Object>타입의 참조변수에 저장불가.(타입 불일치)
3. new Box<**String**>().setItem(**new Object()**); // 에러. 대입된 타입이 String이므로, setItem(**T** item)의 매개변수 역시, String타입만 허용된다.
4. 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;
    }
}
```

1. Juicer.<Apple>makeJuice(new FruitBox<Fruit>());
2. Juicer.<Fruit>makeJuice(new FruitBox<Grape>());
3. **Juicer.<Fruit>makeJuice(new FruitBox<Fruit>());**
4. **Juicer.makeJuice(new FruitBox<Apple>());**
5. Juicer.makeJuice(new FruitBox<Object>());

[정답] 3, 4

[해설]

1. 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.
2. Juicer.<**Fruit**>makeJuice(new FruitBox<**Grape**>()); // 에러. Grape가 Fruit의 자손이라고 해도, 타입이 다르기 때문에 같은 이유로 에러.
3. Juicer.<**Fruit**>makeJuice(new FruitBox<**Fruit**>()); // OK.
4. 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.
5. Juicer.makeJuice(new FruitBox<Object>()); // 에러. 지네릭 메서드의 타입 호출이 생략되지 않았다면, 'Juicer.**<Object>**makeJuice(new FruitBox<**Object**>());'과 같다. 4번의 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;
    }
}
```

1. Box<?> b = new Box();
2. Box<?> b = new Box<>();
3. **Box<?> b = new Box<Object>();**
4. **Box<Object> b = new Box<Fruit>();**
5. Box b = new Box<Fruit>();
6. Box<? extends Fruit> b = new Box<Apple>();
7. **Box<? extends Object> b = new Box<? extends Fruit>();**

[정답] 3, 4, 7

[해설]

1. 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.
2. 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.

3. Box<?> b = new Box<Object>(); // 에러 2의 설명 참고
4. Box<**Object**> b = new Box<**Fruit**>(); // 에러. 타입 불일치
5. Box b = new Box<Fruit>(); // OK. 바람직하지 않음. 'Box<?> b'가 더 나음.

6. Box<? extends Fruit> b = new Box<Apple>(); // OK.
7. 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] 다음 중 메타 애너테이션이 아닌 것을 모두 고르시오.

1. Documented
2. Target
3. Native
4. Inherited

[정답] 3

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)

[표] 자바에서 기본적으로 제공하는 표준 애너테이션(*가 붙은 것은 메타 애너테이션)

[12-6] 애너테이션 TestInfo가 다음과 같이 정의되어 있을 대, 이 애너테이션이 올바르게 적용되지 않은 것은?

```
@interface TestInfo {
    int count() default 1;
    String[] value() default "aaa";
}
```

1. @TestInfo class Exercise12_7 {}
2. **@TestInfo(1) class Exercise12_7 {}**
3. @TestInfo("bbb") class Exercise12_7 {}
4. **@TestInfo("bbb","ccc") class Exercise12_7 {}**

[정답] 1, 3

[해설]

1. @TestInfo class Exercise12_7 {}
   default값이 지정되어 있는 요소는 애너테이션을 적용할 때값을 생략할 수 있다.
2. **@TestInfo(1) class Exercise12_7 {}**
   요소의 이름이 value가 아닌 경우에는 요소의 이름을 생략할 수 없다.
   '@TestInfo(count=1)' 이라고 써야 맞음.
3. @TestInfo("bbb") class Exercise12_7 {}
   @TestInfo(count=1, value={"bbb"})의 생략된 형태
4. **@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

[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 중 쓰레드를 일시정지 상태(WAITING)로 만드는 것이 아닌 것은? (모두 고르시오)

1. suspend()
2. **resume()**
3. join()
4. sleep()
5. wait()
6. **notify()**

[정답] 2, 6

[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.

[13-3] 다음 코드의 실행결과로 옳은 것은?

[연습문제]/ch13/Exercise13_3.java

```
class Exercise13_3 {
   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);
   }
}
```

1. 0102123345456768978 9처럼 0부터 9까지의 숫자가 섞여서 출력된다.
2. **0123456789012345678 9처럼 0부터 9까지의 숫자가 순서대로 출력된다.**
3. IllegalThreadStateException이 발생한다.

[정답] 2

[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.

[13-4] 다음 중 interrupt()에 의해서 실행대기 상태(RUNNABLE)가 되지 않는 경우는? (모두 고르시오)

1. sleep()에 의해서 일시정지 상태인 쓰레드
2. join()에 의해서 일시정지 상태인 쓰레드
3. wait()에 의해서 일시정지 상태인 쓰레드
4. **suspend()에 의해서 일시정지 상태인 쓰레드**

[정답] 4

[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 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_5.java

```
class Exercise13_5
{
   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_5.stopped의 값이 false인 동안 반복한다.
        for(int i=0; !Exercise13_5.stopped; i++) {
            System.out.println(i);

            try {
                Thread.sleep(3*1000);
            } catch(Exception e) {}
        }
   } // run()
}
```

[실행결과]

```
0
1
2
stopped
```

[정답] Exercise13_5.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_5.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_5.

stopped의 값이 바뀌었다 하더라도 일시정지 상태에 있다면, 일시정지 상태가 끝나야만 반복문을 빠져나오게 된다.

```
public void run() {
    // Exercise13_5.stopped의 값이 false인 동안 반복한다.
    for(int i=0; !Exercise13_5.stopped; i++) {
        System.out.println(i);

        try {
            Thread.sleep(3*1000); // 3초간 쉰다.
        } catch(Exception e) {}
    }
} // run()
```

그래서 쓰레드의 실행을 바로 종료시키려면 Exercise13_5.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_5.stopped의 값을 true로 바꾸고, interrupt()를 호출하면 지연 없이 즉시 쓰레드를 멈추게 할 수 있다.

[14-1] 메서드를 람다식으로 변환하여 아래의 표를 완성하시오.

<table>
<tr><th>메서드</th><th>람다식</th></tr>
<tr><td>

```
int max(int a, int b) {
    return a > b ? a : b;
}
```

</td><td>

```
(int a, int b) -> a > b ? a : b
```

</td></tr>
<tr><td rowspan="3">

```
int printVar(String name, int i) {
  System.out.println(name+"="+i );
}
```

</td><td>

```
(String name, int i) ->
  { System.out.println(name+"="+i); }
```

</td></tr>
<tr><td>

```
(name, i) ->
  { System.out.println(name+"="+i); }
```

</td></tr>
<tr><td>

```
(name, i) ->
        System.out.println(name+"="+i)
```

</td></tr>
<tr><td rowspan="3">

```
int square(int x) {
   return x * x;
}
```

</td><td>

```
(int x) -> x * x
```

</td></tr>
<tr><td>

```
  (x)  -> x * x
```

</td></tr>
<tr><td>

```
   x   -> x * x
```

</td></tr>
<tr><td rowspan="2">

```
int roll() {
  return (int)(Math.random() * 6);
}
```

</td><td>

```
() -> { return  (int)(Math.random()*6); }
```

</td></tr>
<tr><td>

```
() -> (int)(Math.random() * 6)
```

</td></tr>
<tr><td>

```
int sumArr(int[] arr) {
    int sum = 0;
    for(int i : arr)
        sum += i;
    return sum;
}
```

</td><td>

```
(int[] arr) -> {
    int sum = 0;
    for(int i : arr)
        sum += i;
    return sum;
}
```

</td></tr>
<tr><td>

```
  int[] emptyArr() {
      return new int[]{};
  }
```

</td><td>

```
  () -> new int[]{}
```

</td></tr>
</table>

[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;
```

1. Function
2. BiFunction
3. Predicate
4. IntBinaryOperator
5. IntFunction

[정답] 4

[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] 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_4 {
   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.

[14-5] 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_5 {
   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();
```

[14-6] 문자열 배열 strArr의 문자열 중에서 가장 긴 것의 길이를 출력하시오.

```
String[] strArr = { "aaa","bb","c", "dddd" };
```

[실행결과]

```
4
```

[정답]

[연습문제]/ch14/Exercise14_6.java

```
import java.util.*;
import java.util.stream.*;

class Exercise14_6 {
  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.
```

[14-7] 임의의 로또번호(1~45)를 정렬해서 출력하시오.

[실행결과]

```
1
20
25
33
35
42
```

[정답]

[연습문제]/ch14/Exercise14_7.java

```
import java.util.*;
import java.util.stream.*;

class Exercise14_7 {
   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.

[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] 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);
```

[15-4] 다음은 콘솔 명령어 중에서 디렉토리를 변경하는 cd명령을 구현한 것이다. 알맞은 코드를 넣어 cd()를 완성하시오.

[연습문제]/ch14/Exercise15_4.java

```
import java.io.*;
import java.util.*;
import java.util.regex.*;

class Exercise15_4 {
   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_4
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.