Java 기본정리 스터디북

1113 신잎새

1. 자바언어의 이해

자바 언어의 특징

>> 프로그래밍 언어 : 문제 해결 방법을 나타내기 위해 특정 단어와 기호를 사용한 명령문들의 집합

자바 언어의 역사

>> 1991 선마이크로시스템즈의 제임스 고슬링을 주축으로 가전제품의 셋톱박스에 사용하기 위해 개발
>> 1991 Oak 언어 개발 : 가전제품과 정보기기를 통합하는 새로운 인터페이스를 제공하는 언어
>> 1995 Java로 이름을 바꾸며 일반에게 공개
>> 1996 Java 1.0 버전 발표

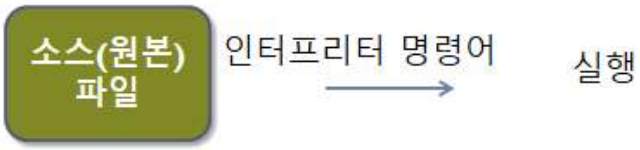
등장배경

>> 특정 컴퓨터마다 실행파일을 만드는 C언어의 문제점
>> 가전제품의 긴 수명으로 인한 완벽한 호환성 및 네트워크를 통한 시스템 업그레이드가 가능한 언어의 필요성
>> C++ 언어의 복잡성

자바 언어의 특징

- 컴파일

▸ 인터프리터



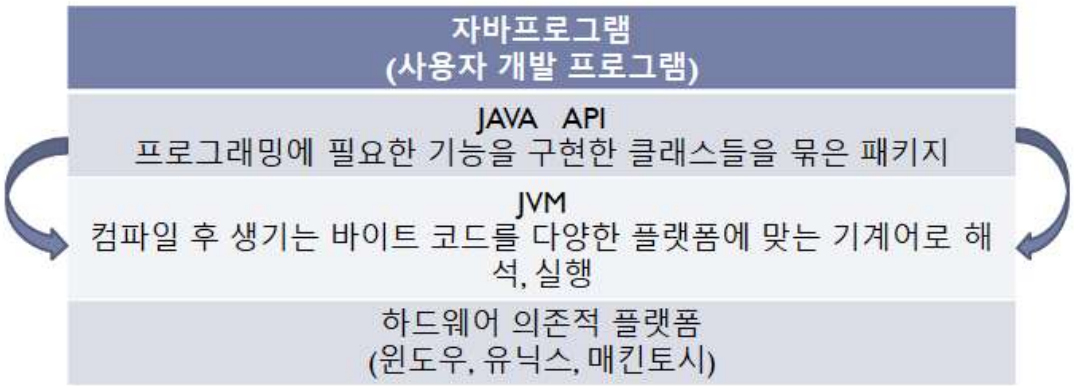
>> 자바는 컴파일러와 인터프리터 언어의 두 가지 특징을 모두 갖는다.

>> 자바 개발환경 J2SE(standard edition)를 이용한다.

1. C++에 비해서 단순하다.
2. 완벽한 객체지향언어이다.
3. 플랫폼에 독립적이다. (VM위에서 실행)
4. 컴파일러와 인터프리터 언어 두 가지 특징을 모두 갖는다.
5. 분산 처리에 용이하다. (네트워크에 강하다)
6. 안전하다. (보안에 강하다.)
7. 멀티스레드를 지원한다. (속도가 빠르다)
8. 동적이다. (라이브러리의 확장이 용이하다)
9. 견고하다.
   9-1) 포인터를 사용하지 않는다.
   9-2) 자동으로 가비지 컬렉션 (garbage collection) 을 수행한다.
   9-3) 엄격한 데이터형 검사를 통해 에러를 조기에 발견한다.
   9-4) 실행시간에 발생하는 에러를 처리한다.

자바 개발 환경

JDK (java developent kit)
>> 하드웨어나 운영체제에 관계없이 실행될 수 있는 자체 플랫폼



자바 개발 환경 종류

>> J2SE : 일반 자바 애플리케이션을 개발하기 위한 기본 자바 개발 환경
>> J2EE : 서버 관련 프로그램을 개발하기 위한 자바 개발 환경
웹서버 환경에서 많이 활용됨 (JSP, Servlet, EJB등)
>> J2ME : 다양한 제품과 임베디든 기기 프로그램을 개발하기 위한 자바 개발환경, 모바일 환경에서 많이 활용됨 (PDA, 핸드폰 등)

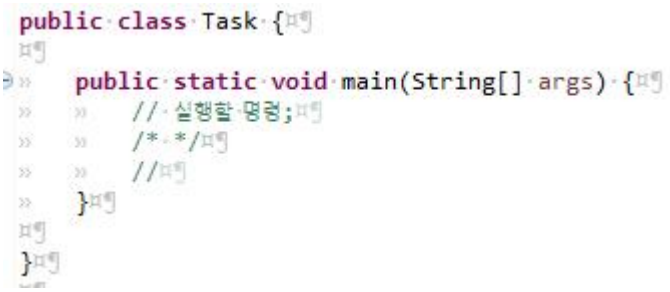
자바 프로그램의 종류

>> Application : 일반 자바 프로그램
>> Servlet 과 JSP : 클라이언트의 요청에 따라 웹서버에서 자바 프로그램을 실행하고 결과만 전송하는 서버 자바 실행기술
>> EJB : 네트워크상에 필요한 객체를 분산시켜 두고 자바 애플리케이션에서 불러 쓸 수 있도록 하는 분산처리 기술
>> 모바일과 임베디드 : 휴대폰을 이용해 무선 인터넷으로 서버로부터 데이터를 다운받는 기술

자바 관련 명령어

>> 컴파일 명령어 : javac 파일이름.java (확장자가 class 인 파일이 생성)
>> 실행 명령어 : java 파일이름 (class를 실행시킴)

자바 프로그램 기본 구조



1. 자바 프로그램은 클래스 들의 집합이다.
2. 클래스 중 하나는 반드시 main() 메서드를 포함한다.
3. VM은 실행 시 main() 메서드를 가장 먼저 호출한다.
4. 클래스명 앞에 public 키워드를 추가하면 그 클래스의 이름은 파일명과 동일해야 하는 규칙이 있기 때문이다.
5. 클래스 이름은 항상 대문자로 시작한다.
6. 참고 : https://wikidocs.net/262

설명문 (comment)

>> 프로그램의 실행에는 영향을 미치지 않으며 프로그램의 설명에 필요한 부분에 작성한다.
>> 한줄 // 두줄이상 /* */

식별자

>> 프로그램의 구성요소를 나타내기 위해 사용하는 단어
>> 식별자의 세 가지 유형
    1. 프로그래머가 선택한 단어
    2. 다른 프로그래머가 선택한 단어
    3. 예약어

실행오류

>> 번역 오류 : 번역과정에서 발생하는 오류
>> 실행 오류 : 잘못된 실행을 하는 경우 프로그램의 동작이 멈추는 경우
>> 논리 오류 : 원하는 결과가 나오지 않고 틀린 답이 나오는 경우로 프로그램의 설계상의 문제가 있는 경우가 대부분, 오류를 수정하기가 매우 어려움

확인 실습

1. '아는 것이 힘이다'를 출력하는 프로그램을 작성하라. (Ex1.java)

```
1  public class Ex1 {
2
3      public static void main(String[] args) {
4
5          System.out.println("아는 것이 힘이다");
6
7      }
8
9  }
10
```

Console

<terminated> Ex1 [Java Application] C:₩Program Files₩Java₩jre1.8.0_201₩bin₩javaw.exe (2020. 2. 10.
아는 것이 힘이다

2. 자신을 소개하는 글을 출력하는 프로그램을 작성하라. (Ex2.java)

```
1  public class Ex2 {
2
3      public static void main(String[] args) {
4
5          System.out.println("안녕하십니까,");
6          System.out.println("미림여자정보과학고등학교 뉴미디어소프트웨어과 ");
7          System.out.println("1학년에 재학중인 신일새입니다.");
8
9      }
10
11 }
```

Console

<terminated> Ex1 [Java Application] C:₩Program Files₩Java₩jre1.8.0_201₩bin₩javaw.exe (2020. 2. 10.
안녕하십니까,
미림여자정보과학고등학교 뉴미디어소프트웨어과
1학년에 재학중인 신일새입니다.

3. 다음과 같은 모양을 출력하는 프로그램을 작성하라. (Ex3.java)

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

```
1  public class Ex3 {
2
3      public static void main(String[] args) {
4
5          System.out.println("  *");
6          System.out.println(" ***");
7          System.out.println("*****");
8          System.out.println(" ***");
9          System.out.println("  *");
10
11     }
12
13 }
```

```
Console
<terminated> Ex1 [Java Application] C:₩Program Files₩Java₩jre1.8.0_201₩
  *
 ***
*****
 ***
  *
```

2. 구조 및 변수

변수

>> 데이터가 저장된 기억 공간(주기억장치)의 이름
>> 데이터는 수시로 바뀔 수 있으므로 변수라고 한다.
>> 덮어쓰기가 가능 > 최종으로 덮어쓰기 된 값이 저장되어 있다.

변수명의 작성규칙

1. 알파벳 대소문자, 숫자, _, $
2. 첫 문자에 숫자는 올 수 없다.
3. 예약어는 사용할 수 없다.
4. 대소문자를 구분한다.
5. 길이 제한은 없다.

확인 실습

1. 정수형(int) 변수 a, b, c를 선언한다.
2. a, b에 각각 10, 20을 배정(=) 한다.
3. c 에 a+b를 배정(=) 한다.
4. 문자형(String) 변수 str 에 “자바를 즐깁시다!~”를 지정한다.
5. a, b, c, str를 화면에 출력한다. (Var_test.java)

```java
public class Var_test {

    public static void main(String[] args) {

        int a = 10;
        int b = 20;
        int c = a + b;
        System.out.println("a :" + a + ", b : " + b);
        System.out.println("a + b = " + c);
        String str = "자바를 즐깁시다!~";
        System.out.println(str);

    }

}
```

Console

```
<terminated> Ex1 [Java Application] C:₩Program Files₩Java₩jre1.8.0_201₩bin₩javaw.exe (2020. 2. 10.
a :10, b : 20
a + b = 30
자바를 즐깁시다!~
```

3. 기본 및 참조 데이터타입

데이터타입 - 기본형 데이터(primitive) 타입

>> 데이터 타입 : 변수에 실제 사용되는 데이터가 저장되는 형(type)

종류	타입	바이트수	비고
논리	boolean	1	소문자 true / false (0,1은 지원안함)
문자	char	2	유니코드 기반, 반드시 작은 따옴표
정수	byte	1	$-2^7 \sim 2^7-1$ (-128 ~ 127)
	short	2	$-2^{15} \sim 2^{15}-1$ (-32768 ~ 32767)
	int	4	기본정수형 $-2^{31} \sim 2^{31}-1$ (-2147483648 ~ 2147483647)
	long	8	숫자의 끝에 l, L을 붙인다. $-2^{63} \sim 2^{63}-1$
실수	float	4	숫자의 끝에 f,F을 붙인다. 약 $-3.4*10^{38} \sim 3.4*10^{38}$
	double	8	기본실수형, 약 $-1.7*10^{308} \sim 1.7*10^{38}$

확인 실습

1. 다음과 같은 형태로 세 명의 학생들 이름과 점수를 출력하고 합계를 계산하는 프로그램을 작성하라. (Grade.java)

```
---------- 자바실행 ----------
이름  과제 보너스 합계
선미   82  7   89
성우   71  4   75
가연   92  8   100

출력 완료 (0초 경과) - 정상 종료
```

```
1  public class Grade {
2
3      public static void main(String[] args) {
4
5          System.out.println("이름  과제  보너스  합계");
6          System.out.println("선미  82    7     89");
7          System.out.println("성우  71    4     75");
8          System.out.println("가연  92    8     100");
9
10     }
11
12 }
13
```

```
Console
<terminated> Ex1 [Java Application] C:₩Program Files₩Java₩jre1.8.0_201₩bin₩javaw.exe (2020. 2. 10.
이름  과제  보너스  합계
선미  82    7     89
성우  71    4     75
가연  92    8     100
```

2. 두 개의 임의의 실수(3.5와 7.2)를 정의하고 사칙연산의 결과를 출력하는 프로그램을 작성하라. (Four_op.java)

```java
public class Four_op {

    public static void main(String[] args) {

        double a = 3.5;
        double b = 7.2;
        System.out.println("a + b = " + (a + b));
        System.out.println("a - b = " + (a - b));
        System.out.println("a * b = " + (a * b));
        System.out.println("a / b = " + (a / b));

    }

}
```

Console

```
<terminated> Ex1 [Java Application] C:₩Program Files₩Java₩jre1.8.0_201₩bin₩javaw.exe (2020. 2. 10.
a + b = 10.7
a - b = -3.7
a * b = 25.2
a / b = 0.4861111111111111
```

3. 직사각형 모양의 땅 면적을 계산하여 출력하는 프로그램을 설계한 후 작성하라. 땅의 가로 길이는 100m이고 세로길이는 50m이다. (Square.java)

```java
public class Square {

    public static void main(String[] args) {

        int weight = 100;
        int height = 50;
        System.out.println("땅의 면적 : " + weight * height + "m^2");

    }

}
```

Console

```
<terminated> Ex1 [Java Application] C:₩Program Files₩Java₩jre1.8.0_201₩bin₩javaw.exe (2020. 2. 11.
땅의 면적 : 5000m^2
```

4. 주어진 섭씨온도를 화씨온도로 바꾸어 주는 프로그램을 설계한 후 작성하라. 주어진 섭씨온도는 15도이다. (섭씨온도 C를 화씨온도 F로 바꾸어 주는 공식은 F = 9/5 * C + 32 이다.) (Temp.java)

```
1  public class Temp {
2
3      public static void main(String[] args) {
4
5          int c_temp = 15;
6          int f_temp = (int) (9.0 / 5.0 * c_temp + 32);
7          System.out.println("섭씨온도 " + c_temp + "도는 화씨온도 " + f_temp + "도 입니다.");
8
9
10     }
11
12 }
```

```
Console
<terminated> Ex1 [Java Application] C:₩Program Files₩Java₩jre1.8.0_201₩bin₩javaw.exe (2020. 2. 11. 오전 12:0
섭씨온도 15도는 화씨온도 59도 입니다.
```

데이터 형변환 - 묵시적 형변환

>> 데이터 형변환 : 이미 선언된 데이터형을 다른 데이터형으로 변환하는 것을 의미

종류	타입	바이트수
논리	boolean	1
문자	char	2
정수	byte	1
	short	2
	int	4
	long	8
실수	float	4
	double	8

데이터 형변환 - 강제형변환(타입캐스팅)

>> 강제형변환(타입캐스팅) : 더 작은 데이터형으로 변환하는 것을 의미

데이터 타입 - 참조 타입

>> 참조타입 : 데이터가 저장되어있는 메모리 주소를 기억하고 이 주소를 통해 해당 데이터를 참조할 수 있도록 한다.

※참고

>> 기본자료형 변수

정의 : 기본 자료형 값을 가진다.
종류 : 정수, 실수, 문자, 논리값 등
예 : int a = 100; (a라는 변수에 실제 100이라는 값이 들어간다)

>> 참조자료형 변수

정의 : 값에 대한 참조, 즉 주소를 갖는다.
종류 : 배열, 클래스차모변수, 인터페이스 등
예 : int[] a = new int[3];
a[0] = 10; (a라는 변수에는 a[0]값이 저장된 위치의 주소값이 저장되어있다)
++ int 타입이므로 다음 주소값들은 4byte씩 더해간다.

상수

정의 : 프로그램의 실행 중 변화하지 않는 값, 정수형, 실수형, 문자형, 논리형 상수가 있다. final 예약어를 변수타입 앞에 쓰며 상수명은 대문자로 작성한다.

상수 사용의 유용한점

1. 불분명한 값에 의미를 부여
2. 프로그램 수정 용이
3. 프로그래머의 의해 잘못 입력되는 것을 방지

Scanner를 활용한 편리한 키보드 입력

1. Java.util. 패키지에 있는 Scanner 클래스를 활용
2. import java.util.Scanner 문장을 첫 줄에 삽입

```
* 키보드로부터 숫자를 입력 받아 score 정수형 변수에 배정 :
Scanner scan = new Scanner(System.in);
int score = scan.nextInt( );
```

```
* 키보드로부터 숫자를 입력 받아 weight 실수형 변수에 배정 :
Scanner scan = new Scanner(System.in);
float weight = scan.nextFloat( );
```

```
* 키보드로부터 한 줄의 문자열 입력 받아 input 문자열에 배정 :
Scanner scan = new Scanner(System.in);
String input = scan.nextLine( );
```

확인 실습

1. 시험점수를 입력받아 기준점수(85점) 이상이면 '통과', 기준점수 미만이면 '탈락'을 출력하는 프로그램을 작성하라. 단, 기준점수는 상수로 정의한다. (Constant_1.java)

```
1  import java.util.Scanner;
2
3  public class Constant_1 {
4
5      public static void main(String[] args) {
6
7          Scanner scan = new Scanner(System.in);
8
9          final int CONSTANT = 85;
10         int score;
11
12         System.out.print("시험점수를 입력하세요 : ");
13         score = scan.nextInt();
14
15         if(score >= CONSTANT) {
16             System.out.println("통과");
17         }
18         else {
19             System.out.println("탈락");
20         }
21
22
23     }
24
25 }
```

```
Console
<terminated> Ex1 [Java Application] C:₩Program Files₩Java₩jre1.8.0_201₩bin₩javaw.exe (2020. 2. 11.
시험점수를 입력하세요 : 90
통과
```

2. 반지름이 10인 원의 둘레와 넓이를 구하는 프로그램을 작성하라. 단, 파이값은 상수로 지정하여 사용하라. (Constant_2.java)

```java
public class Constant_2 {

    public static void main(String[] args) {

        final double PI = 3.14;
        int radius = 10;
        double circum = 2 * PI * radius;
        double area = PI * Math.pow(radius, 2);

        System.out.println("반지름이 " + radius + "인 원의 둘레 : " + circum);
        System.out.println("반지름이 " + radius + "인 원의 넓이 : " + area);

    }

}
```

```
Console
<terminated> Ex1 [Java Application] C:₩Program Files₩Java₩jre1.8.0_201₩bin₩javaw.exe (2020. 2. 11.
반지름이 10인 원의 둘레 : 62.80000000000004
반지름이 10인 원의 넓이 : 314.0
```

3. 사각형의 밑변과 높이를 입력 받아 사각형의 넓이를 구하는 프로그램을 작성하라. (DataTypeTest_04.java)

```java
import java.util.Scanner;

public class DataTypeTest_04 {

    public static void main(String[] args) {

        int width, height;
        int area;

        Scanner scan = new Scanner(System.in);
        System.out.print("사각형 밑변의 길이 : ");
        width = scan.nextInt();
        System.out.print("사각형 높이의 길이 : ");
        height = scan.nextInt();

        area = width * height;
        System.out.println("사각형의 넓이 : " + area);

    }

}
```

```
Console
<terminated> Ex1 [Java Application] C:₩Program Files₩Java₩jre1.8.0_201₩bin₩javaw.exe (2020. 2. 11.
사각형 밑변의 길이 : 10
사각형 높이의 길이 : 5
사각형의 넓이 : 50
```

Scanner를 활용한 편리한 키보드 입력

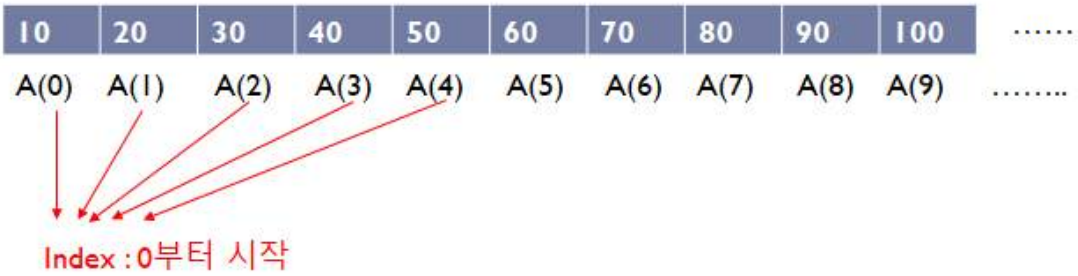
>> Scanner에서 제공하는 메소드 정리

변환 타입	메소드이름	설명
String	next()	다음 한 토큰을 읽는다.
String	nextLine()	다음 한 줄을 읽는다.
byte	nextByte()	다음 한 바이트를 읽는다.
int	nextInt()	다음 정수를 읽는다.
long	nextLong()	다음 Long 정수를 읽는다.
short	nextShort()	다음 Short 정수를 읽는다.
float	nextFloat()	다음 부동소수점 수를 읽는다.
double	nextDouble()	다음 부동소수점 double 수를 읽는다.

3-1. 배열

데이터 타입 - 배열

>> 같은 종류의 데이터를 여러 개 저장하기 위한 기억장소
>> 객체로 처리하며 참조형 변수
>> 메모리를 절약하고 간결한 프로그램 가능



배열의 사용

>> 1단계 : 배열선언
기억 장소의 주소를 가리키는 변수를 선언하며 크기를 지정할 수 없다.
형식 : 데이터형 배열명[] / 데이터형[] 배열명;
예) int a[] 또는 int[] a(권장) // a라는 정수형(int) 기억장소 생성

>> 2단계 : 배열 생성

new 연산자로 기억공간을 확보하고 주소를 1단계의 배열 변수에 저장, 크기를 지정

형식 : 배열명 = new 데이터형[길이];

예) a = new int [3] // a라는 정수형(int) 기억장소 3개 생성

>> 3단계 : 배열 초기화

형식 : 배열명[배열순서(0부터)] = 값; 또는

데이터형 배열명[] = new 데이터형[]{값, 값...};

예) a[0] = 10; a[1] = 20; a[2] = 30;

int a[] = new int[]{1,2,3};

배열의 초기값

배열의 초기값은 특별히 지정하지 않는 경우 다음과 같은 초기값을 가진다.

byte(1), short(2), int(4)	Long(8)	Float(4)	Double(8)	Boolean(1)	char(2)
0	0L	0.0F	0.0	false	'\u0000'

확인 실습

1. 1-2+3-4+5-6 ..... +99-100 의 합계를 1차원 배열을 이용하여 구하라. 조건은 다음과 같다. int형, 배열 요소 개수가 100개인 배열을 선언하라. (Array_sum.java)

✓ for 문을 이용하여 배열의 값을 배정하라.

✓ 더하는 식은 한 번만 사용한다

```
1  public class Array_sum {
2
3      public static void main(String[] args) {
4
5          int sum = 0;
6          int array[];
7          array = new int[100];
8
9          for(int i = 0; i < 100; i++) {
10
11             if((i + 1) % 2 == 0 ) { //짝수일경우
12
13                 array[i] = (i + 1) * -1;
14
15             }
16             else { //홀수일경우
17
18                 array[i] = i + 1;
19
20             }
21
22             sum += array[i];
23
24         }
25         System.out.println("1-2+3-4+5-6+ ..... +99-100 = " + sum);
26     }
27
28 }
```

```
Console
<terminated> Ex1 [Java Application] C:₩Program Files₩Java₩jre1.8.0_201₩bin₩javaw.exe (2020. 2. 11. .
1-2+3-4+5-6+ ..... +99-100 = -50
```

// array배열을 이용하여 초기화 후 sum 변수에 array배열에 초기화 된 값을 누적하여 합하였다.

2. 10명의 점수를 입력 받아 최고점수를 받은 학생을 결정하는 프로그램을 작성하라. (Array_max.java)

✓ Scanner를 통해 입력 받아 score 배열에 점수를 저장하라.

✓ score 배열은 정수형이고 길이는 10이다.

✓ 최댓값을 결정하기 위한 변수 max를 사용하라.

```java
import java.util.Scanner;

public class Array_max {

    public static void main(String[] args) {

        int max = 0;
        int score[];
        score = new int[10];

        Scanner scan = new Scanner(System.in);

        for(int i = 0; i < score.length; i++) {
            System.out.print((i+1) + "번째 학생의 점수를 입력하세요 : ");
            score[i] = scan.nextInt();
            if(score[i] > max) {
                max = score[i];
            }
        }
        System.out.println("최고점수 : " + max + "점");
    }

}
```

```
Console
<terminated> Ex1 [Java Application] C:₩Program Files₩Java₩jre1.8.0_201₩bin₩javaw.exe (2020. 2. 11.
1번째 학생의 점수를 입력하세요 : 10
2번째 학생의 점수를 입력하세요 : 20
3번째 학생의 점수를 입력하세요 : 30
4번째 학생의 점수를 입력하세요 : 40
5번째 학생의 점수를 입력하세요 : 50
6번째 학생의 점수를 입력하세요 : 60
7번째 학생의 점수를 입력하세요 : 90
8번째 학생의 점수를 입력하세요 : 80
9번째 학생의 점수를 입력하세요 : 70
10번째 학생의 점수를 입력하세요 : 60
최고점수 : 90점
```

// if문을 사용하여 점수비교후 max 변수에 대입하여 최고점수를 구하였다.

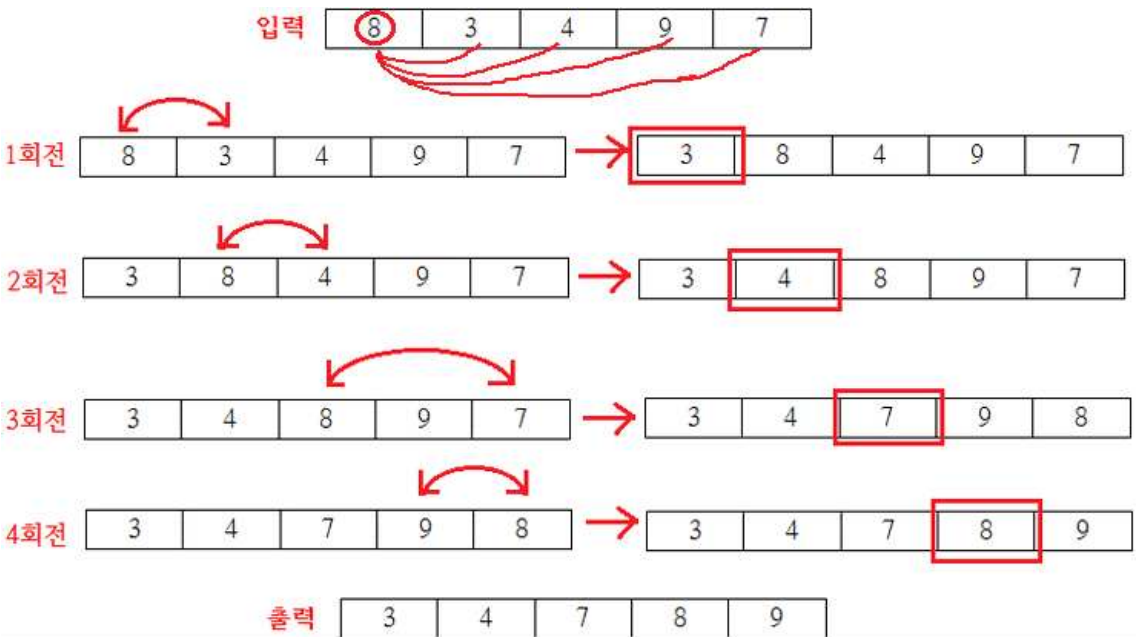
※참고

정렬 : 주어진 자료를 어떤 기준에 의하여 크기 순서로 배열하는 것

>> 오름차순 정렬 : 작은 순서에서 큰 순서로 나열하는 것

>> 내림차순 정렬 : 큰 순서에서 작은 순서로 나열하는 것

선택정렬(selection sort)



1회전	2회전	3회전	4회전
A[0], A[1]	A[0], A[1]	A[0], A[1]	A[0], A[1]
A[0], A[2]	A[0], A[1]	A[0], A[1]	A[0], A[1]
A[0], A[3]	A[0], A[1]	A[0], A[1]	A[0], A[1]
A[0], A[4]	A[0], A[1]	A[0], A[1]	A[0], A[1]
가장 작은값	두 번째 작은값	세 번째 작은값	네 번째 작은값
>> A[0]	>> A[1]	>> A[2]	>> A[3]

버블정렬(bubble sort)

1회전	2회전	3회전	4회전
A[0], A[1]	A[0], A[1]	A[0], A[1]	A[0], A[1]
A[1], A[2]	A[1], A[2]	A[1], A[2]	A[0], A[1]
A[2], A[3]	A[2], A[3]	A[2], A[3]	
A[3], A[4]	A[3], A[4]		
A[4], A[5]			

확인실습

10명의 점수를 입력 받아 오름차순으로 점수를 정렬하는 프로그램을 작성하라. (Array_sort.java)

1. Scanner를 통해 10명의 점수를 입력 받아 score 배열에 점수를 저장하라
2. 유효한 점수(0-100점)만 입력한다고 가정하자.
3. 배열 방법은 선택정렬(select sort)을 이용하라.
4. 오름차순이란 낮은 점수에서 높은 점수 순서로 정렬하는 것을 의미한다.
5. Scanner를 통해 10명의 점수를 입력 받아 score 배열에 점수를 저장하라
6. 유효한 점수(0-100점)만 입력한다고 가정하자.
7. 배열 방법은 버블정렬(select sort)을 이용하라.
8. 오름차순이란 낮은 점수에서 높은 점수 순서로 정렬하는 것을 의미한다.

```java
import java.util.Scanner;

public class Array_sort {

    public static void main(String[] args) {

        int[] score = new int [10];
        int temp;

        Scanner scan = new Scanner(System.in);
        for(int i = 0; i < score.length; i++) {
            System.out.print((i + 1) + "번째 학생의 점수 : ");
            score[i] = scan.nextInt();
        }

        for(int i = 0; i < score.length; i++) {
            for(int j = i + 1; j < score.length; j++) {
                if(score[i] > score[j]) {
                    temp = score[i];
                    score[i] = score[j];
                    score[j] = temp;
                }
            }
        }

        System.out.println("**************************");
        for(int i = 0; i < score.length; i++) {
            System.out.println((score.length - i) + "등 학생의 점수 : " + score[i]);
        }
    }
}
```

Console

```
<terminated> Array_sort (1) [Java Application] C:₩Program Files₩Java₩jre1.8.0_201₩bin₩javaw.exe (2020. 2.
1번째 학생의 점수 : 10
2번째 학생의 점수 : 30
3번째 학생의 점수 : 50
4번째 학생의 점수 : 70
5번째 학생의 점수 : 90
6번째 학생의 점수 : 80
7번째 학생의 점수 : 60
8번째 학생의 점수 : 40
9번째 학생의 점수 : 20
10번째 학생의 점수 : 100
**************************
10등 학생의 점수 : 10
9등 학생의 점수 : 20
8등 학생의 점수 : 30
7등 학생의 점수 : 40
6등 학생의 점수 : 50
5등 학생의 점수 : 60
4등 학생의 점수 : 70
3등 학생의 점수 : 80
2등 학생의 점수 : 90
1등 학생의 점수 : 100
```

데이터 타입 - 2차원 배열

>> 행과 열을 갖는 이차원 배열의 기본 원리

배열이름 (arr)	열 0 ↓	열 1 ↓	열 2 ↓	열 3 ↓
행 0 →	a arr[0][0]	b arr[0][1]	c arr[0][2]	d arr[0][3]
행 1 →	e arr[1][0]	f arr[1][1]	g arr[1][2]	h arr[1][3]
행 2 →	i arr[2][0]	j arr[2][1]	k arr[2][2]	l arr[2][3]

>> 2차원 배열의 형식

배열 선언 -> 배열 생성 -> 배열초기화

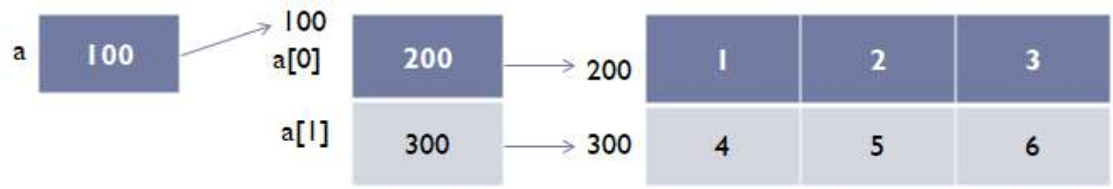
```
int a[ ][ ] = new int[3][2];
a[0][0] = 1;  a[0][1] = 4;
a[1][0] = 2;  a[1][1] = 5;
a[2][0] = 3;  a[2][1] = 6;
```

```
int[ ][ ] a= new int[3][2];
a[0][0] = 1;  a[0][1] = 4;
a[1][0] = 2;  a[1][1] = 5;
a[2][0] = 3;  a[2][1] = 6;
```

```
int a[ ][ ] = new int{{1,2, 3},{4,5,6}};
```

```
int[ ][ ] a = new int{{1,2,3},{4,5,6}};
```

// 위의 배열을 그림으로 그려보면 다음과 같다.



확인실습

1. 2차원 배열을 이용하여 아래와 같이 출력하는 프로그램을 for문 2개를 이용하여 작성하라. (ArrayExam1.java)

```
---------- 자바실행 ----------
1   1   1   1
1   1   1   1
1   1   1   1
1   1   1   1

출력 완료 (0초 경과) - 정상 종료
```

```
1  ¤¶
2  public class ArrayExam11 {¤¶
3  ¤¶
4  »   public static void main(String[] args) {¤¶
5  »   »   int [][] a = new int[4][4];¤¶
6  »   »   for(int i=0; i<a.length; i++) {¤¶
7  »   »   »   for (int j=0; j<a[i].length; j++) {¤¶
8  »   »   »   »   a[i][j] =1;¤¶
9  »   »   »   }¤¶
10 »   »   }¤¶
11 »   »   for(int i=0; i<a.length; i++) {¤¶
12 »   »   »   for (int j=0; j<a.length; j++) {¤¶
13 »   »   »   »   System.out.print(a[i][j]+" ");¤¶
14 »   »   »   »   ¤¶
15 »   »   »   }¤¶
16 »   »   »   System.out.println(" ");¤¶
17 »   »   }¤¶
18 »   }¤¶
19 }¤¶
20 ¤¶
```

Console

<terminated> ArrayExam11 [Java Application] C:₩Program Files₩Java₩j

```
1 1 1 1
1 1 1 1
1 1 1 1
1 1 1 1
```

4. 연산자

산술연산자

>> 가감승제
>> 연산 순위는 대수와 동일
>> 단항 +, - -> *, /, % -> +, -

종류	사용법	기능	비고
+	a+b	덧셈	
-	a-b	뺄셈	
*	a*b	곱셈	
/	a/b	나눗셈	
%	a%b	나머지	실수형도 가능 cf) C언어는 정수형

산술연산자

>> 크기나 타입 비교시 사용, 결과값은 true / false 이다.

종류	사용법	기능	비고
>	a>b	크다	true or false
>=	a>=b	크거나 같다	true or false
<	a<b	작다	true or false
<=	a<=b	작거나 같다	true or false
==	a==b	같다	true or false
!=	a!=b	같지 않다	true or false

※참고

제어문 - for 반복문

>> 반복문이란 특정 부분을 반복해서 실행하는 구조문

예) for(선언 및 초기화; 조건; 증감부){
  // 실행내용(반복될 내용)
}

1. 반복 변수 초기화
2. 조건체크 -> 참이면 실행, 거짓이면 종료
3. 실행 후 반복 변수 증가
4. 예) for(int i = 1; i < 10; i++) -> 9번 실행

for문 실행순서

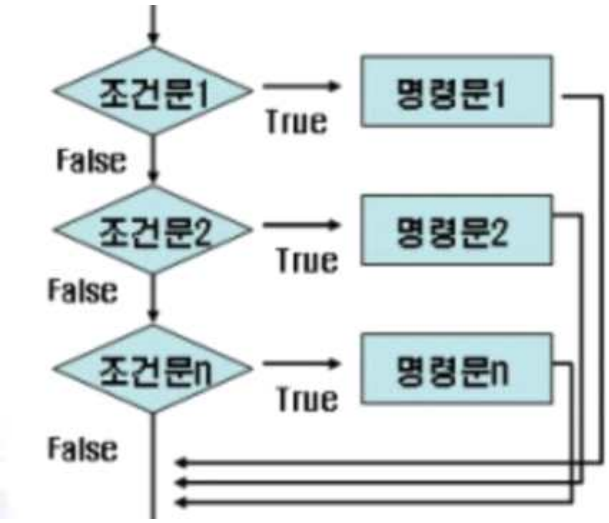
초기식-> 조건비교-> 참이면 실행문 실행-> 증감식-> 거짓이면 빠져나가기

>> 따라서 처음 조건식 비교에서 거짓이면 for문을 한번도 실행하지 않을 수 있음

제어문 - if else

>> 조건에 따라 선택적으로 문장이 실행되는 구조문

예) if(조건식) {
        참일 때 수행;
} else {
        거짓일 때 수행;
}



확인실습

1. 1부터 100사이의 숫자 중 3의 배수인 수를 구해보자. (Multiple_3.java)
 >> %연산자를 이용한다.
 >> 3의 배수는 3으로 나누어 나머지가 0인 수이다.
 >> 반복문 for문을 이용한다.

```
3  public class Multiple_3 {
4  
5      public static void main(String[] args) {
6  
7          for (int i = 1; i <= 100; i++ ) { //초기, 조건, 증감
8  
9              if(i % 3 == 0) {
10                 System.out.print(i+" ");
11             }
12 
13         }
14 
15     }
16 
17 }
18
```

```
Console
<terminated> Multiple_3 [Java Application] C:₩Program Files₩Java₩jre1.8.0_201₩bin₩javaw.exe (2020. 2. 27. 오전 6:21:47)
3 6 9 12 15 18 21 24 27 30 33 36 39 42 45 48 51 54 57 60 63 66 69 72 75 78 81 84 87 90 93 96 99
```

논리 연산자
 >> 주어진 조건식이 참인지 거짓인지를 판단

종류	사용법	기능	비고
&	a & b	a, b 모두 true 인 경우 true	AND
&&	a && b	&와 동일 단, a가 false인 경우는 b를 수행하지 않고 false반환	
|	a | b	a와 b중 하나라도 true 이면 true	OR
||	a || b	|와 동일 단, a가 true인 경우는 b를 수행하지 않고 true반환	
!	! a	a가 true면 false, a가 false면 true	NOT

확인실습

1. 1부터 100사이의 숫자 중 1의자리가 3, 6, 9인 수를 찾아보자. (LogicalOp.java)

>> %연산자를 이용한다.

>> 1의 자리를 표현하는 방법을 생각한다.

>> !=, ==, &&연산자를 적절히 사용한다.

```
3  public class LogicalOp {
4
5      public static void main(String[] args) {
6
7          for (int i = 1; i <= 100; i++) {
8
9              if(i % 10 == 3 || i % 10 == 6 || i % 10 == 9) {
10             System.out.print(i+" ");
11             }
12             
13         }
14     
15     }
16 
17 }
```

```
Console
<terminated> LogicalOp [Java Application] C:₩Program Files₩Java₩jre1.8.0_201₩bin₩javaw.exe (2020. 2. 27. 오전 6:24:22)
3 6 9 13 16 19 23 26 29 33 36 39 43 46 49 53 56 59 63 66 69 73 76 79 83 86 89 93 96 99
```

2. 정수의 절대값을 구하는 프로그램을 작성해보자. 이때, 정수값을 키보드로 입력받는다. (Absolute_test.java)

>> if else 문을 사용한다.

```
3  import java.util.Scanner;
4
5  public class Absolute_test {
6
7      public static void main(String[] args) {
8
9          int num;
10         Scanner scan = new Scanner(System.in);
11
12         System.out.print("수 입력: ");
13         num = scan.nextInt();
14
15         if(num >= 0) {
16             System.out.print("절대값 : " + num);
17             
18         } else {
19             
20             System.out.print("절대값 : " + num * (-1));
21         }
22         
23     }
24 
25 }
```

```
Console
<terminated> Absolute_test [Java Application] C:₩Program Files₩Java₩jre1.8.0_201
수 입력: -7
절대값 : 7
```

증감연산자

>> 변수의 값에 1을 증가 또는 감소시킨 후 그 값을 변수에 다시 저장하는 연산자

종류	사용법	비고	
		연산식	결과
++ a	a를 증가 후 수식에 적용	a=10; b=++a + 10	a = 11 b = 21
a++	수식에 적용 후 a를 증가	a=10; b=(a++) + 10	a = 11 b = 20
-- a	a를 감소 후 수식에 적용	a = 10 b = --a + 10	a = 9 b = 19
a --	수식에 적용 후 a를 감소	a =10; b = a-- + 10;	a = 9 b = 20

실습예제

1. ++num 과 num++ 연산의 차이를 정확히 이해한다.
2. 다음 프로그램의 결과를 예상해보자.

```
3  public class Increase_op {
4
5      public static void main(String[] args) {
6
7          int num = 0;
8          num = num + 1;
9          System.out.println("num : " + num); // 1
10         System.out.println("++num : " + (++num)); // 2
11         System.out.println("num++ : " + (num++)); // 2
12         System.out.println("num : " + num); // 3
13         System.out.println("--num : " + (--num)); // 2
14         System.out.println("num-- : " + (num--)); // 2
15         System.out.println("num : " + num); // 1
16
17     }
18
19 }
```

```
Console
<terminated> Increase_op [Java Application] C:₩Program Files₩Java₩jre1.8.0_2
num : 1
++num : 2
num++ : 2
num : 3
--num : 2
num-- : 2
num : 1
```

비트 연산자

>> 비트 별로 연산을 수행

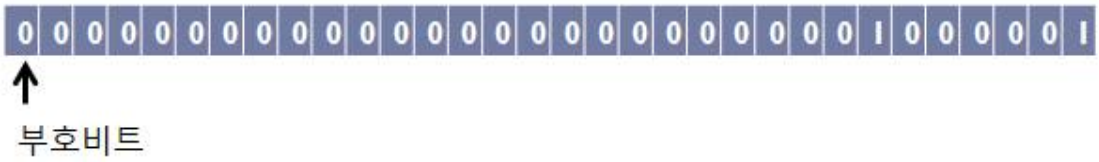
종류	사용법	기능	비고
&	a & b	비트별 AND	00000110 & 11111101 => 00000100
|	a | b	비트별 OR	00000110 | 00001000 => 00001110
^	a ^ b	비트별 XOR (서로 다르면 1, 같으면 0)	00000110 ^ 00001111 => 00001001
~	~a	비트별 반전	~00000110 => 11111001
>>	a >> b	a를 b비트 만큼 오른쪽 이동	양수이면 왼쪽 비트를 0으로 채우기 음수이면 왼쪽 비트를 1로 채우기 /2의 효과 00010100(20)>>2 => 00000101(5) 20을 2의 제곱으로 나눈 효과
<<	a << b	a를 b비트 만큼 왼쪽 이동	양수이면 오른쪽 비트를 0으로 채우기 음수이면 오른쪽 비트를 1로 채우기 *2의 효과 예 : 2^3 00010100(20)<<2 => 01010000(80) 20을 2의 제곱을 곱한 효과
>>>	a >>> b	a를 b비트 만큼 오른쪽 이동	부호에 상관없이 왼쪽 비트를 0으로 채우기

비트 연산자의 활용

종류	사용법	활용
&	a & b	특정 비트를 0으로 만들기 위해 사용
|	a | b	특정 비트를 1로 만들기 위해 사용
^	a ^ b	특정 비트를 비트 반전시키기 위해 사용
~	~a	1의 보수를 만들기 위해 사용
>>	a >> b	곱셈과 나눗셈에 응용
<<	a << b	곱셈과 나눗셈에 응용
>>>	a >>> b	곱셈과 나눗셈에 응용

실습예제

1. 정수 65에 대해 >>3, <<3, >>>3을 적용해보고, 결과를 예상해보자.




```
3  public class ShiftBitOp {
4
5      public static void main(String[] args) {
6
7          int a = 65;
8          System.out.println(a>>3);
9          System.out.println(a<<3);
10         System.out.println(a>>>3);
11     }
12
13 }
```

Console

<terminated> ShiftBitOp [Java Application] C:₩Program Files₩Java₩

```
8
520
8
```

조건연산자(삼항 연산자)

>> 우변의 결과를 좌변에 대입하는 연산자

종류	사용법	기능	비고

? :	조건 ? 처리1 : 처리2	조건이 참이면 처리1을, 조건이 거짓이면 처리2를 처리한다.	a = 2;
			b = 3;
			c = (a>b) ? a: b;
			
			c 는 얼마 ?

대입연산자

종류	기능	비고
=	a = b	b를 a에 대입
+=	a = a + b ;	a를 b만큼 증가시키고 a에 다시 대입
-=	a = a - b ;	a를 b만큼 감소시키고 a에 다시 대입
*=	a = a * b	a를 b만큼 곱하고 a에 다시 대입
/=	a = a / b;	a를 b로 나누고 a에 다시 대입
%=	a = a % b	a를 b로 나눈 나머지를 a에 다시 대입

연산자 우선순위

순위	연산자	순위	연산자
1	() [ ] .(함수, 괄호, 도트)	8	& (비트 AND)
2	++ -- ~ ! (전치)	9	^(비트 XOR)
3	* / % (승제)	10	|(비트 OR)
4	+ - (가감)	11	&&(논리 AND)
5	<< >> >>>(비트이동)	12	||(논리 OR)
6	< <= > >=(비교)	13	? : (삼항)
7	== !=(비교)	14	=(대입)

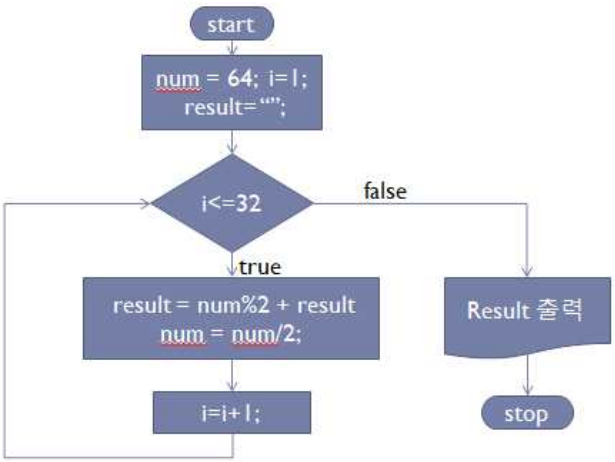
확인실습

1. 10진수(int) 64를 32비트 2진수로 변환하는 프로그램을 작성하라. (DecimalToBin.java)

>> 10진수를 2진수로 바꾸는 연산 이해

>> int의 범위 이해

>> ++, %, /= 연산자의 이해




```
1  import java.util.Scanner;
2
3  public class DecimalToBin {
4
5      public static void main(String[] args) {
6
7          int num;
8          String result = "";
9
10         Scanner scan= new Scanner(System.in);
11         System.out.print("숫자를 입력하세요 : ");
12         num = scan.nextInt();
13
14         for (int i = 1 ; i <= 32; i++) {
15             result = num % 2 + result; // 나머지 붙이기
16             num = num/2; // 몫 => 나누기
17         }
18
19         System.out.println("결과 : " + result);
20     }
21 }
```

```
Console
<terminated> DecimalToBin (1) [Java Application] C:₩Program Files₩Java₩jr
숫자를 입력하세요 : 64
결과 : 00000000000000000000000001000000
```

5, 제어문_반복문

반복문(for문)

>> 조건식이 참인 동안 문장이나 블록을 반복한다.

종류 : for, while, do ~ while

for문

형식 : **for**(1. 초기식; 2. 조건식; 3. 증감식) {

4. 실행문 ...

}

실행순서 : 1 2 4 3 2 4 3 2 4 3 2... 3 2

1. 반복변수 초기화
2. 조건체크 -> 참이면 실행, 거짓이면 종료
3. 실행후
4. 반복 변수 증가
5. 첫 조건이 거짓이면 한 번도 실행하지 않을 수 있다.

확인예제

1. for문을 이용하여 1부터 10까지의 수를 출력하는 프로그램 작성 (For1to10.java)

```
2  public class For1to10 {
3
4      public static void main(String[] args) {
5
6          for(int i = 1; i <= 10; i++) {
7              System.out.print(i + " ");
8          }
9
10     }
11
12 }
```

Console

<terminated> For1to10 (1) [Java Application] C:₩Program Files₩Jav

1 2 3 4 5 6 7 8 9 10

2. for문을 이용하여 10부터 1까지의 수를 출력하는 프로그램 작성 (For10to1.java)

```
2  public class For10to1 {
3
4      public static void main(String[] args) {
5          
6          for(int i = 10; i >= 1; i--) {
7              System.out.print(i + " ");
8          }
9
10     }
11
12 }
```

Console

```
<terminated> For10to1 (1) [Java Application] C:₩Program Files₩
10 9 8 7 6 5 4 3 2 1
```

3. for 문을 이용하여 1부터 10까지의 수의 합을 출력하는 프로그램 작성 (For1to10sum.java)

```
4      public static void main(String[] args) {
5          
6          int sum = 0;
7          
8          for(int i = 1; i <= 10; i++) {
9              sum += i;
10         }
11         
12         System.out.println("합 : " + sum);
13
14     }
15
16 }
```

Console

```
<terminated> For1to10sum (1) [Java Application] C:₩Program Files
합 : 55
```

4. for문을 이용하여 1부터 100사이의 짝수의 합을 출력하는 프로그램 작성 (EvenSum1to100.java)

```
2  public class EvenSum1to100 {
3
4      public static void main(String[] args) {
5          
6          int evensum = 0;
7          
8          for(int i = 1; i <= 100; i++) {
9              
10             if(i % 2 == 0) {
11                 evensum += i;
12             }
13             
14         }
15         System.out.println("합 : " + evensum);
16     }
17 
18 }
```

Console

```
<terminated> EvenSum1to100 (1) [Java Application] C:₩Program Fil
합 : 2550
```

확인예제

1. 중첩 for문을 이용하여 구구단을 출력하는 프로그램을 작성 (Gugudan.java)

```
2  public class Gugudan {
3
4      public static void main(String[] args) {
5          
6          for(int i = 1; i <= 9; i++) {
7              for(int j = 1; j <= 9; j++) {
8                  System.out.println(i + " * " + j + " = " + i * j);
9              }
10         }
11 
12     }
13 
14 }
```

Console

```
<terminated> Gugudan (1) [Java Application] C:₩Program Files₩Java₩jre1.8.0_201₩bin₩javaw
8 * 5 = 40
8 * 6 = 48
8 * 7 = 56
8 * 8 = 64
8 * 9 = 72
9 * 1 = 9
9 * 2 = 18
9 * 3 = 27
9 * 4 = 36
9 * 5 = 45
9 * 6 = 54
9 * 7 = 63
9 * 8 = 72
9 * 9 = 81
```

반복문(while문)

>> 조건문이 참인 동안만 문장이나 블록을 반복실행한다.

>> 조건식을 먼저 테스트하므로 한번도 반복문이 실행되지 않을 수도 있다.

형식 : 초기식

```
        while (조건식) {
        실행문;
        증가식;
}
```

확인예제

1. while문을 이용하여 1부터 10까지의 수를 출력하는 프로그램 작성 (While1to10.java)

```
3  public class While1to10 {
4
5      public static void main(String[] args) {
6
7          int i = 1;
8          while(i <= 10) {
9              System.out.print(i + " ");
10             i++;
11         }
12
13     }
14
15 }
```

Console

```
<terminated> While1to10 [Java Application] C:₩Program Files₩Java
1 2 3 4 5 6 7 8 9 10
```

2. while문을 이용하여 1부터 100사이의 홀수를 출력하는 프로그램 작성 (Whileodd1to100.java)

```
3  public class Whileoddto1to100 {
4
5      public static void main(String[] args) {
6
7          int i = 0;
8          while(i < 100) {
9              i++;
10
11             if(i % 2 == 1) {
12                 System.out.print(i + " ");
13             }
14
15         }
16
17     }
18
19 }
```

Console

<terminated> Whileoddto1to100 [Java Application] C:₩Program Files₩Java₩jre1.8.0_201₩bin₩javaw.exe (2020. 2. 28.

```
39 41 43 45 47 49 51 53 55 57 59 61 63 65 67 69 71 73 75 77 79 81 83 85 87 89 91 93 95 97 99
```

3. while문을 이용하여 1부터 100까지의 합을 출력하는 프로그램 작성 (Whilesum1to100.java)

```
3  public class Whilesum1to100 {
4
5      public static void main(String[] args) {
6
7          int i = 1;
8          int sum = 0;
9
10         while (i <= 100) {
11             sum += i;
12             i++;
13         }
14
15         System.out.print("1부터 100 까지의 합 : " + sum);
16     }
17
18 }
```

Console

<terminated> Whilesum1to100 (1) [Java Application] C:₩Program Files₩Java₩j

```
1부터 100 사이의 합 : 5050
```

반복문(do while문)

>> 조건문이 참인 동안만 문장이나 블록을 반복실행한다.

>> 조건식을 나중에 테스트하므로 반복문이 무조건 한번은 실행된다.

형식 : 초기식;

do {

실행문

} **while**(조건식);

확인예제

1. do while문을 이용하여 1부터 10까지의 수를 출력하는 프로그램 작성 (Dowhile1to10.java)

```
3  public class DoWhile1to10 {
4
5      public static void main(String[] args) {
6
7          int i = 1;
8          do {
9              System.out.print(i + " ");
10             i++;
11         }while(i <= 10);
12
13     }
14
15 }
```

Console

<terminated> DoWhile1to10 [Java Application] C:₩Program Files₩

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

2. do while문을 이용하여 1부터 100사이의 홀수를 출력하는 프로그램 작성 (Dowhileodd1to100.java)

```
3  public class DoWhileodd1to100 {
4
5      public static void main(String[] args) {
6          int i = 1;
7          do {
8              if(i % 2 == 1)
9              System.out.print(i + " ");
10             i++;
11         } while(i <= 100);
12
13     }
14
15 }
```

Console

<terminated> DoWhileodd1to100 [Java Application] C:₩Program Files₩Java₩jre1.8.0_201₩bin₩javaw.exe (2020. 2. 2

1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61 63

3. do while문을 이용하여 1부터 100까지의 합을 출력하는 프로그램 작성 (Dowhilesum1to100.java)

```
3  public class Dowhilesum1to100{
4
5      public static void main(String[] args) {
6          int i = 1;
7          int sum = 0;
8
9          do {
10             sum += i;
11             i++;
12         } while(i <= 100);
13
14         System.out.println("1부터 100까지의 합 : " + sum);
15     }
16
17 }
```

Console

<terminated> Dowhilesum1to100 [Java Application] C:₩Program Files₩Java₩jre1.8.0_2(

1부터 100까지의 합 : 5050

분기문 - break, continue

break

>> Switch나 반복문(for, while)에서 가장 가까운 반복 블록을 빠져 나오게 하기 위해서 사용한다.

```
for ( int i=1; i<=10; i++){
  if(i== 5) break;
  System.out print(i+"\t");
}       => 결과는?
```

// 결과 : 1 2 3 4

continue

>>반복문에서 continue 이후의 문장은 실행하지 않고 그 다음 반복을 계속할 때 사용한다.

```
for ( int i=1; i<=10; i++){
  if(i== 5) continue ;
  System.out print(i+"\t");
}       => 결과는?
```

// 결과 : 1 2 3 4 6 7 8 9 10

확인예제

1. 입력받은 숫자 n에 대해 n!을 출력하는 프로그램을 작성하라. (Fact_method.java)

```java
import java.util.Scanner;

public class Fact_method {

    public static void main(String[] args) {

        int n;
        int n_fact = 1;
        Scanner scan = new Scanner(System.in);
        System.out.print("숫자를 입력하세요 : ");
        n = scan.nextInt();

        for(int i = 1; i <= n; i++) {
            n_fact *= i;
        }

        System.out.println("n! = " + n_fact);

    }

}
```

```
Console
<terminated> Fact_method [Java Application] C:₩Program Files₩Java₩jre1.8.0_201₩b
숫자를 입력하세요 : 5
n! = 120
```

2. 숫자를 입력 받아 약수를 구하는 프로그램을 작성하라. (Divisor.java)

>> num을 1부터 num까지의 숫자로 나누어서 나머지가 0, 즉 나누어떨어지는 수는 num의 약수가 된다.

```
1  import java.util.Scanner;
2
3  public class Divisor {
4
5      public static void main(String[] args) {
6
7          int num;
8          Scanner scan = new Scanner(System.in);
9          System.out.print("숫자를 입력하세요 : ");
10         num = scan.nextInt();
11
12         System.out.print("num의 약수 : ");
13         for(int i = 1; i <= num; i++) {
14
15             if(num % i == 0) {
16                 System.out.print(i + " ");
17             }
18
19         }
20
21     }
22
23 }
```

Console

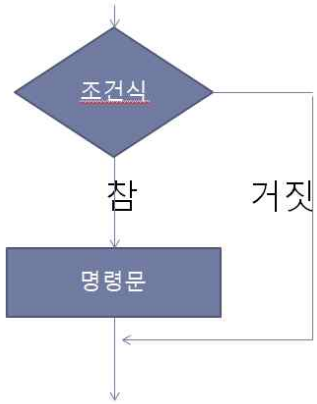
<terminated> Divisor [Java Application] C:₩Program Files₩Java₩jre1.8.0_201₩

숫자를 입력하세요 : 24
num의 약수 : 1 2 3 4 6 8 12 24

5, 제어문_조건문

제어문(if)

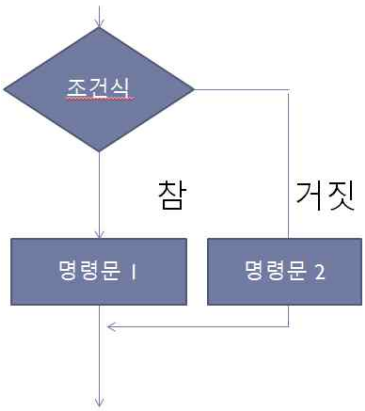
>> 조건에 따라 선택적으로 문장이 실행되는 구조문



형식 : if(조건식) {
참일 때 수행;
// 거짓이면 { } 다음 문장을 수행한다.
}

제어문(if_else)

>> 조건에 따라 선택적으로 문장이 실행되는 구조문

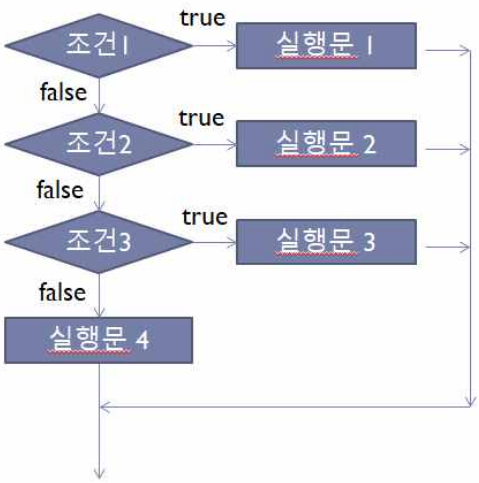


형식 : if(조건식) {
참일 때 수행;
} else {
거짓일 때 수행;
}

제어문(if_else if …… else)

>> 조건에 따라 선택적으로 문장이 실행되는 구조문

형식 : if(조건식1) {
참일 때 수행;
} else if (조건식2) {
참일 때 수행;
} else if (조건식3) {
참일 때 수행;
} …. {
} else {
모두 거짓일 때 수행;
}



실습예제

1. 정수 두 개를 입력 받아 큰 값을 구하는 프로그램을 작성하라. (Max.java)

```java
import java.util.Scanner;

public class Max {

    public static void main(String[] args) {

        int num1, num2;
        Scanner scan = new Scanner(System.in);
        System.out.print("두 정수를 입력하시오 : ");

        num1 = scan.nextInt();
        num2 = scan.nextInt();

        if (num1 > num2) {
            System.out.println("큰 수 : " + num1);
        }

        if (num2 > num1) {
            System.out.println("큰 수 : " + num2);
        }

    }

}
```

2. 큰 값을 구하는 메서드 getMax()를 이용하여 프로그램을 작성하라. (Max_Method.java)

```java
import java.util.Scanner;

public class Max_Method {

    public static void main(String[] args) {

        int num1, num2;
        Scanner scan = new Scanner(System.in);
        System.out.print("두 정수를 입력하시오 : ");
        num1 = scan.nextInt();
        num2 = scan.nextInt();

        System.out.println("큰 수 : " + getMax(num1, num2));

        }

    public static int getMax(int a, int b) { // 선언 후 호출

        int result = 0;
            if(a > b) {
                result = a;
            }

            if(a < b) {
                result = b;
            }

            return result;

            } // end of getMax
    }
```

Console

<terminated> Max_Method [Java Application] C:₩Program Files₩Java₩jre1.8.0_201₩bin₩javaw.exe

```
두 정수를 입력하시오 : 18 17
큰 수 : 18
```

3. 세 수를 입력 받아 가장 작은 값을 구하라. (MinOfThree.java)

```java
import java.util.Scanner;

public class MinOfThree {

	public static void main(String[] args) {

		int num1, num2, num3;
		Scanner scan = new Scanner(System.in);
		System.out.print("세 정수를 입력하시오 : ");

		num1 = scan.nextInt();
		num2 = scan.nextInt();
		num3 = scan.nextInt();

		if (num1 < num2 & num2 < num3 || num1 < num3 & num3 < num2) {
			System.out.println("가장 작은수 : "+ num1);
		}else if(num2 < num1 & num1 < num3 || num2 < num3 & num3 < num1) {
			System.out.println("가장 작은수 :" + num2);
		}else if(num3 < num2 & num2 < num1 || num3 < num2 & num2 < num1) {
			System.out.println("가장 작은수 :" + num3);
		}else if(num1 == num2 & num2 != num3) {
			System.out.println("첫번째 수와 두번째 수가 같습니다.");
		}else if(num2 == num3 & num1 != num3) {
			System.out.println("두번째 수와 세번째 수가 같습니다.");
		}else if(num3 == num1 & num1 != num2) {
			System.out.println("첫번째 수와 세번째 수가 같습니다.");
		}else {
			System.out.println("세 수가 같습니다.");
		}
	}
}
```

```
Console
<terminated> MinOfThree [Java Application] C:₩Program Files₩Java₩jre1.8.0_201₩bin₩javaw.exe (2020. 2
세 정수를 입력하시오 : 3 15 18
가장 작은수 : 3
```

※참고

명령행 매개변수

>> main() 메서드는 문자열을 매개 변수로 받아 프로그램 실행 시 필요한 정보를 프로그램에 전달하는데 이러한 매개 변수를 명령행 매개변수라고 한다.

```
public class Grade_method{
  public static void main(String[ ] args) {
    int score;                    명령행매개변수
      .......
      System.out.println("===> 당신의 성취도는 " + achieve(score)+" 입니다");
    }                                                    인자
}

public static char achieve(int score){
      ......                      매개변수
      return ach_cha;
   }
}
```

>> public static void main(String[ ] args)의 의미

예약어	의미
public	접근제어자로서 모든 클래스에서 접근이 가능
static	메모리에 가장 먼저 로딩되며 프로그램이 종료될 때까지 메모리에 상주한다.
void	메서드의 리턴 타입이 void
main	메서드 이름이 main
String[ ] args	메서드의 매개변수로 string 타입의 배열 args

즉, java 명령어가 프로그램을 실행할 때 메모리에 가장 먼저 로딩되어있는 main 메서드를 실행한다. 이 때 명령줄에 제공되는 인수들이 args 배열에 저장된다. 배열의 타입은 String이다.

제어문(switch~case문)

>> 조건식에 대한 상수값에 따라 여러 갈래로 분기가 되는 문장

형식 : **switch**(조건식) {

case 상수1: 실행문; break;

case 상수2: 실행문; break;

case 상수3: 실행문; break;

……

default: 실행문;

}

>> break를 만나면 switch를 빠져나가고 break가 없으면 다음 실행문이 계속 실행된다.

>> 상수는 조건식의 결과값과 일치하는 상수값을 의미한다.

실습예제

1. 키보드로부터 연산자를 입력 받아 두 수에 대해 연산을 실행하는 프로그램을 switch문을 이용하여 작성하라. (SwitchOp.java)

```
1  import java.util.Scanner;
2
3  public class SwitchOp {
4
5      public static void main(String[] args) {
6
7          int a =20, b = 10;
8          char op;
9
10         Scanner scan = new Scanner(System.in);
11         System.out.print("연산자를 입력하세요 : ");
12         op = scan.next().charAt(0);
13
14         switch(op) {
15         case '+' : System.out.println(a + " + " + b + " = " + (a + b)); break;
16         case '-' : System.out.println(a + " - " + b + " = " + (a - b)); break;
17         case '*' : System.out.println(a + " * " + b + " = " + (a * b)); break;
18         case '/' : System.out.println(a + " / " + b + " = " + (a / b)); break;
19         default : System.out.println("사칙연산자가 아닙니다.");
20         } // 일반적으로 default에는 break를 쓰지 않는다.
21
22     }
23
24 }
```

```
Console
<terminated> SwitchOp [Java Application] C:₩Program Files₩Java₩jre1.8.0_201₩bin₩javaw.exe (2020. 2. 28. 오
연산자를 입력하세요 : +
20 + 10 = 30
```

확인실습

1. 1부터 사용자로부터 입력 받은 숫자 사이에 존재하는 홀수의 합과 짝수의 합을 구하라. (Sum_odd_Even.java)

```java
import java.util.Scanner;

public class Sum_odd_Even {

    public static void main(String[] args) {

        int num = 0;
        int odd_sum = 0;
        int even_sum = 0;

        Scanner scan = new Scanner(System.in);
        System.out.print("숫자를 입력하세요 : ");
        num = scan.nextInt();

        for(int i = 0; i <= num; i++) {

            if(i % 2 == 0) {
                even_sum += i;
            }

            if(i % 2 == 1) {
                odd_sum += i;
            }

        }

        System.out.println("홀수의 합 : " + odd_sum);
        System.out.println("짝수의 합 : " + even_sum);

    }

}
```

Console

<terminated> Sum_odd_Even [Java Application] C:₩Program Files₩Java₩jre1.8.0_201₩bin₩

```
숫자를 입력하세요 : 5
홀수의 합 : 9
짝수의 합 : 6
```

2. 입력받은 양의 정수가 3의 배수, 5의 배수, 혹은 8의 배수인지를 알려주는 프로그램을 설계하고 작성하라. 양의 정수가 3의 배수이면 '3의 배수이다'를 출력하고, 5의 배수이면 '5의 배수이다'를 출력하고, 8의 배수이면 '8의 배수이다'를 출력하라. 그 외의 경우이면 '어느 배수도 아니다'를 출력하라.
단, 공배수는 고려하지 않고 작성한다. (Multiple_if.java)

```java
import java.util.Scanner;

	public class Multiple_if {

		public static void main(String[] args) {

			int number;
			Scanner scan = new Scanner(System.in);
			System.out.print("양의 정수를 입력하시오 : ");

			number = scan.nextInt();

			if (number % 3 == 0) {
				System.out.println("3의 배수이다.");
			}
			else if (number % 5 == 0) {
				System.out.println("5의 배수이다.");
			}
			else if (number % 8 == 0) {
				System.out.println("8의 배수이다.");
			}
			else {
				System.out.println("어느 배수도 아니다.");
			}

		}

	}
```

```
Console
<terminated> Multiple_if [Java Application] C:₩Program Files₩Java₩jre1.8.0_201₩bin₩javaw.exe (202
양의 정수를 입력하시오 : 43
어느 배수도 아니다.
```

3. 한 직장인의 연간 근로소득에 대한 소득세를 계산하는 프로그램을 설계하고 작성하라. 근로소득에 대한 소득세는 다음과 같다 :
연간 근로소득은 입력받아야 한다. (Tax_calculate.java)

- 근로소득이 2,000만 원 이하이면 근로소득의 5%이다.
- 근로소득이 2,000만 원을 초과하고 4,000만 원 이하이면 근로소득의 15%이다.
- 근로소득이 4,000만 원을 초과하고 8,000만 원 이하이면 근로소득의 25%이다.
- 근로소득이 8,000만 원을 초과하면 근로소득의 40%이다.

```java
import java.util.Scanner;

    public class Tax_calculate {

        public static void main(String[] args) {

        double money;
        Scanner scan = new Scanner(System.in);
        System.out.print("연간 근로소득을 입력하세요 : ");
        money = scan.nextInt();

        if (money <= 20000000) {
            System.out.println("소득세 : " + money / 20.0);
        } else if (200000000 < money && money <= 400000000) {
            System.out.println("소득세 : " + money * 3 / 20.0);
        } else if (400000000 < money && money <= 800000000) {
            System.out.println("소득세 : " + money * 5 / 20.0);
        } else {
            System.out.println("소득세 : " + money * 8 / 20.0);
        }

    }

}
```

```
Console
<terminated> Tax_calculate (1) [Java Application] C:₩Program Files₩Java₩jre1.8.0_201₩bin₩javaw.exe
연간 근로소득을 입력하세요 : 850000000
소득세 : 3.4E8
```

4. 직원의 연봉과 근무평가등급을 입력받아 연봉 인상금액과 새 연봉을 계산하여 출력하는 프로그램을 설계하고 구현하라. 직원의 근무평가등급은 우수, 보통, 불량 중의 하나이다. 우수 등급을 받은 직원은 연봉이 6% 인상되고 보통 등급을 받은 직원은 연봉이 4% 인상되고 불량 등급을 받은 직원은 연봉이 2% 인상된다. 다음 모범 출력과 같은 결과가 나오도록 프로그램을 작성하라. (Salary_calculate.java)

```java
import java.util.Scanner;

	public class Salary_calculate {
		
		public static void main(String[] args) {
			
			double currentSalary;
			String rating = "";
			Scanner scan = new Scanner(System.in);
			
			System.out.print("현 연봉을 입력하세요(단위 : 만원) : ");
			currentSalary = scan.nextInt();
			System.out.print("근무 평가등급을 입력하세요(우수, 보통, 불량) : ");
			rating = scan.next();
			setRating(rating, currentSalary);
			
		}

		public static void setRating(String rating, double currentSalary) {
			
			double raise = 0;
			double newSalary;
			
			switch (rating) {
			case "우수" : raise = currentSalary * 6 / 100;
			case "보통" : raise = currentSalary * 4 / 100;
			case "불량" : raise = currentSalary * 2 / 100;
			}

			newSalary = currentSalary + raise;
			
			System.out.println("연봉 인상액 : " + raise);
			System.out.println("새 연봉 : " + newSalary);
		
		}

	}
```

```
Console
<terminated> Salary_switch_1 [Java Application] C:₩Program Files₩Java₩jre1.8.0_201₩bin₩javaw.exe (2020. 2
현 연봉을 입력하세요(단위 : 만원) : 2400
근무 평가등급을 입력하세요(우수, 보통, 불량) : 우수
연봉 인상액 : 48.0
새 연봉 : 2448.0
```

6. 생성자메서드

생성자메서드

>> 클래스로부터 객체를 생성하고 객체의 초기화를 담당하는 특수한 메서드로 객체가 생성될 때 무조건 수행된다.

>> 외형적으로 일반 메서드와 비슷하지만 다음과 같은 특징이 있다.

1. new 연산자와 함께 사용
2. 객체 생성시 멤버 변수의 초기화를 담당
3. 이름은 클래스 이름과 동일하며 첫 문자는 대문자
4. return 유형이 없다.
5. 생성자 메서드가 없는 경우 JVM이 자동으로 기본 생성자를 삽입한다.
6. 사용자가 정의한 생성자 메서드가 있는 경우 JVM이 삽입한 기본 생성자 메서드는 사라진다.

1) 객체를 가리키는 참조 변수를 정의한다.
   ex) Account acct;

2) **생성자 함수**를 이용하여 클래스의 객체를 만든다.
   ex) acct = new Account();

>> 위의 두 과정은 하나로 합칠 수 있다.
   ex) Account acct = new Account();

3) 생성자 메소드를 통해 객체 변수들의 초기 값들을 줄 수 있다.

```
public Account (String name, int amount) {
    setOwnerName(name);
    setBalance(amount);
  }
```

>> 위의 세 과정은 하나로 합칠 수 있다.
   ex) Account acct = new Account("홍길동", 10000);

형식 : 접근제어자 생성자이름(매개변수1, 매개변수2, ……)

>> 접근제어자는 public, protected, private를 사용할 수 있다.

1. 생성자 메서드가 없는 클래스 : 생성자는 객체 생성시 반드시 호출되어야 하는 메서드이다. 그러나 반드시 생성자 메서드를 갖고 있을 필요는 없다.

```
class Student {
    String name;
    int grade;
    int _class;
}

class StudentExam {

    public static void main(String[] args) {
        Student kim = new Student();

    }

}
```

// kim객체는 JVM이 자동으로 기본 생성자를 삽입해서 생성

2. 생성자 메서드가 있는 클래스

```
class ConstructorExam {

    public static void main(String[] args) {
        Student kim = new Student();
        Student jang = new Student("장민재");
        System.out.println("학생의 이름은 " + kim.name + "입니다."); // 1번
        System.out.println("학생의 이름은 " + jang.name + "입니다."); //2번
    }

}

class Student{
    String name;
    String telephone;
    int grade;
    int _class;
    int number;
    public Student(){}; // 매개변수가 없는 생성자메서드1
    public Student(String n) {name = n;} // 매개변수가 한개인 오버로딩된 생성자메서드2
}
```

실습예제

1. 사람의 성명을 모델링하는 Person 클래스를 설계하고 작성하라.
그 클래스는 사람의 성과 이름을 나타내고 다음 메소드들을 가진다.
  > 사람의 성과 이름을 넘겨받아 초기화하는 생성자메서드
  > 성을 반환하는 메서드
  > 이름을 반환하는 메서드
  > 성과 이름 안에 포함된 문자들의 총 수를 반환하는 메서드

```
class Person{
    
    private String lastName;
    private String firstName;
    
    public Person(String lastName, String firstName) {
        this.lastName = lastName;
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }
    public int getLength() {
        return (lastName + firstName).length();
    }
    public String toString() {
        return "성 : " + getLastName() + ", 이름 : " + getFirstName() + ", 성명의 길이 : " + getLength() + " 글자";
    }
    
}
```

2. 1의 클래스를 시험하는 PersonDriver 클래스를 작성하라.
 > 사용자로부터 성과 이름을 입력받은후
 > 각 성명에 대해 Person 인스터스를 만들고
 > 성과 이름을 반환한 후 성명의 길이를 출력하는 프로그램을 작성하라.

```
3  import java.util.Scanner;
4
5  public class PersonDriver {
6
7      public static void main(String[] args) {
8
9          Scanner scan = new Scanner(System.in);
10         String lastName;
11         String firstName;
12
13         System.out.print("성을 입력하세요 : ");
14         lastName = scan.next();
15         System.out.print("이름을 입력하세요 : ");
16         firstName = scan.next();
17         Person p1 = new Person(lastName, firstName);
18         System.out.println(p1.toString());
19
20     }
21
22 }
```

Console

```
<terminated> PersonDriver [Java Application] C:₩Program Files₩Java₩jre1.8.0_20
성을 입력하세요 : 신
이름을 입력하세요 : 잎새
성 : 신, 이름 : 잎새, 성명의 길이 : 3 글자
```

3. 원을 모델링하는 Circle 클래스를 설계하고 작성하라.

> 원의 반지름을 통해 원을 생성하고, 원의 반지름을 알려주는 메서드

> 원의 면적을 계산하는 메서드

> 원의 둘레를 계산하는 메서드

> 원의 반지름을 주어진 값으로 변경하는 메서드

```
class Circle {
    
    private double radius;
    private final double PI = 3.14;
    
    public Circle(double radius) {
        this.radius = radius;
    }
    
    public double getRadius() {
        return radius;
    }
    
    public void setRadius(double radius) {
        this.radius = radius;
    }
    
    public double computeArea(double radius) {
        double area = (double) (radius * radius * PI);
        return area;
    }
    
    public double computePerimeter(double radius) {
        double Perimeter = (double) (2 * PI * radius);
        return Perimeter;
    }
    public String toString() {
        return "반지름 : " + getRadius() + " 원의 면적 : " + computeArea(radius) + " 원의 둘레 : " + computePerimeter(radius);
    }
    
}
```

4. 3의 클래스를 시험하는 CircleDriver 클래스를 작성하라.
 > 반지름이 4.5인 Circle 객체를 만들어라.
 > 그 객체의 면적과 둘레를 계산하여 출력하라.
 > 그 객체의 반지름을 5로 변경하라.

```
3  import java.util.Scanner;
4
5  public class CircleDriver {
6
7      public static void main(String[] args) {
8
9          Scanner scan = new Scanner(System.in);
10         double radius;
11
12         System.out.print("반지름을 입력하세요 : ");
13         radius = scan.nextDouble();
14         Circle c1 = new Circle(radius);
15         System.out.println(c1.toString());
16
17     }
18
19 }
```

```
Console
<terminated> CircleDriver [Java Application] C:₩Program Files₩Java₩jre1.8.0_201₩bin₩
반지름을 입력하세요 : 3
반지름 : 3.0  원의 면적 : 28.26  원의 둘레 : 18.84
```

※참고

왜 main()메서드에 static을 붙이는 걸까?

일반적으로 static이 붙지 않은 메서드의 경우 메서드 호출이 실행될 때 메모리를 할당받는다. 그리고 실행이 종료되면 메모리를 반환한다. 그에 반해 static이 붙은 변수나 메서드는 호출전에 메모리를 먼저 할당받고 프로그램이 실행되는 동안 계속 상주한다. 따라서 main() 메서드는 자바 프로그램이 실행되는 동안, 호출하지 않더라도 메모리를 할당받고 항상 상주하고 있어야 하므로 static을 붙인다.

7. 메서드와 오버로딩

1. this(1)

>> 현재 생성되어 사용중인 인스턴스 객체 자신을 의미

>> 생성자메서드나 메소드의 매개변수 이름이 클래스의 멤버 변수 이름과 동일할 경우 사용

```
Class Student{
  String name;
  int grade;
  int class;
  public Student(String name,int grade){
          this.name = name;
          this.grade = grade;
  }
```

=> 좌측의 this.name은 현재의 객체 참조 변수 name 을 의미

```
Class Student{
  String name;
  int grade;
  int class;
  public Student(String name int grade){
          name = name;
          grade = grade;
  }
```

=> 생성자의 매개변수와 멤버 변수가 동일 한 경우

1. this를 사용함으로써 의미를 명확하게 하여 가독성을 높인다.
2. 객체 변수나 매개변수의 이름으로 같은 이름을 사용할 수 있다는 장점이 생긴다.

1. this(2)

>> 현재 **같은 클래스 내의 다른 생성자 메서드를 호출**하는 경우도 사용 가능하다.

```
Class  Box{
  int width;
  int  height;
  int depth;
  public Box(){ //1번생성자
    this(1,1,1); //3번호출
  }
  public Box(int w,int h){ //2번생성자
    this(w,h,1);//3번호출
  }
  public Box(int w,int h, int d){ //3번생성자
    width = w;
    height = h;
    depth = d;
  }
}
```

> 어떤 형태의 객체가 생성되더라도 결국 마지막의 생성자가 실행되게 된다. 왜냐하면 위 두 개의 this는 세 번째의 생성자 함수를 호출하도록 되어있기 때문이다.

> 생성자 내에서 this구문은 반드시 첫 라인에 위치하여야 한다. 그렇지 않은 경우 오류를 발생시킨다.

▶ 교과서 69p 예제

```
Class  Student{
  String name;
  int grade;
  int class;

  public Student(){}

  public Student(String name){
     this.name = name;
  }

  public Student(String name,int grade){
     this(name);
     this.grade = grade;
  }
```

> 세번째 형태의 객체가 생성될 경우, 두 번째 생성자 메서드도 호출 된다.

2, **메소드와 오버로딩**

>> 클래스가 가지는 동적인 특성인 행위
>> 메서드는 일반적으로 소문자로 시작
>> 생성자메서드는 리턴타입이 없으나, 일반적인 메소드는 반드시 리턴타입 (void, int, double, String .....) 이 있다.

- 접근제어자
[접근제어자] [활용방법] 리턴타입 메서드이름 (매개변수)

> 접근제어자 : public / default / private / protected
> 활용방법 : static / final / abstract / synchronized

public : 모든 클래스에서 접근 가능
protected : 동일 패키지 또는 하위 클래스에서만 접근가능
default : 동일 패키지내의 클래스에서만 접근가능
private : 자신의 클래스 안에서만 사용 가능 (외부에서 접근 불가)
 > 정보은닉, 보안

접근제어자	같은 클래스의 멤버	같은 패키지의 멤버	자식 클래스의 멤버	그 외의 영역
public	O	O	O	O
protected	O	O	O	X
default	O	O	X	X
private	O	X	X	X

- 활용방법

final : 하위 클래스에서 오버라이딩 될 수 없음

abstract : 추상메소드로 추상클래스는 선언부분만 가지고 몸체는 가질 수 없다. 몸체는 서브 클래스에서 오버라이딩됨. 따라서 추상클래스를 통해서 인스턴스 객체를 생성할 수 없다.

synchronized : 스레드를 동기화할 수 있는 기법을 제공하기 위해 사용됨

캡슐화(Encapsulation)

>> 멤버변수는 접근권한을 private으로 해서 외부에서는 숨겨진 형태로 만들고 public으로 지정한 메소드를 통해서만 멤버 변수에 접근 가능하도록 한다.

3. **클래스 메서드**

>> 멤버변수에 static으로 선언되는 클래스 변수가 있듯이 메서드에도 클래스 메서드가 있다.

1. 클래스를 로딩할 때 생긴다.
2. 클래스 이름을 통해 접근한다.
3. 클래스로부터 생성된 모든 객체가 공유한다.
4. 일반 객체 변수를 사용할 수 없고 클래스 변수만 사용한다.

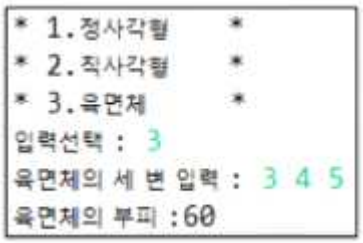
4. **메소드 오버로딩**

>> 생성자의 오버로딩과 동일한 개념

1. 같은 클래스 안에 매개변수의 개수, 타입, 순서를 달리하는 동일한 이름의 메소드가 여러 개 존재
2. 다형성을 구현할 수 있음
3. 리턴타입, 접근제어자가 다른 것은 상관 없다.

실습예제

1. 키보드를 통해 입력 받은 매개변수가 1개이면 정사각형의 넓이를, 2개이면 직사각형의 넓이를, 3개이면 육면체의 부피를 구하는 cals() 메소드를 오버로딩을 이용하여 구현하라. (Moverloading.java)




```
import java.util.Scanner;
class Test{
    int cal(int length) {
        int area = length * length;
        return area;
    }
    int cal(int width, int height) {
        int area = width * height;
        return area;
    }
    int cal(int length, int width, int height) {
        int volumn = length * width * height;
        return volumn;
    }
}
```

// Test 클래스와 cal 메소드

```java
public class Moverloading {

    public static void main(String[] args) {

        int option;

        Scanner scan = new Scanner(System.in);
        System.out.println("* 1. 정사각형      *");
        System.out.println("* 2. 직사각형      *");
        System.out.println("* 3. 육면체        *");
        System.out.print("입력선택 : ");
        option = scan.nextInt();

        Test t1 = new Test();

        if(option == 1) {
            int length;
            Scanner sc = new Scanner(System.in);
            System.out.print("정사각형의 변 입력 : ");
            length = sc.nextInt();
            System.out.println("정사각형의 넓이 : " + t1.cal(length));
        }

        else if (option == 2) {
            int width, height;
            Scanner sc = new Scanner(System.in);
            System.out.print("직사각형의 두 변 입력 : ");
            width = sc.nextInt();
            height = sc.nextInt();
            System.out.println("직사각형의 넓이 : " + t1.cal(width, height));
        }

        else if (option == 3) {
            int length, width, height;
            Scanner sc = new Scanner(System.in);
            System.out.print("육면체의 세 변 입력 : ");
            length = scan.nextInt();
            width = scan.nextInt();
            height = scan.nextInt();
            System.out.println("육면체의 부피 : " + t1.cal(length, width, height));
        }

        else {
            System.out.println("1부터 3 사이의 숫자를 입력하세요.");
        }

    }

}
```

Console

<terminated> OverLodingTest [Java Application] C:₩Program Files₩Java₩jre1.8.0_201₩bin₩javaw.exe (2020. 3.

```
* 1. 정사각형      *
* 2. 직사각형      *
* 3. 육면체        *
입력선택 : 3
육면체의 세 변 입력 : 3 4 5
육면체의 부피 : 60
```

2. 인터넷 쇼핑몰에서 item을 구입하는 고객 Client클래스를 설계하고 구현하라. (Client.java)

1) 고객은 이름(name)과 사이버머니(c_money)를 입력받아 초기화도 하면서 대응하는 매개변수로 변경까지도 가능한 생성자 메서드를 갖는다.

2) item을 일정한 수량만큼 구입한다. 구입시 사이버머니가 구입금액보다 커야한다.

3) 고객의 구입 현황과 사이버머니의 잔액을 출력한다.

4) 이외 필요한 내용은 각자 필요에 따라 추가하여 작성한다.

```
3  public class Client {
4      private String name;
5      private int c_money;
6      private String itemName;
7
8      public Client(String name, int c_money) {
9          super();
10         this.setName(name);
11         this.setC_money(c_money);
12     }
13
14     public String getName() {
15         return name;
16     }
17
18     public void setName(String name) {
19         this.name = name;
20     }
21
22     public int getC_money() {
23         return c_money;
24     }
25
26     public void setC_money(int c_money) {
27         this.c_money = c_money;
28     }
29
30     public void buyItem(Item1 i, int n) {
31         if(i.getPrice() * n <= this.c_money) {
32             c_money -= i.getPrice() * n;
33             i.calculateSales(n);
34             itemName = i.getName();
35         }
36         else {
37             System.out.println("잔액부족!! 구입불가!!");
38         }
39     }
40
41     public void printClient() {
42         System.out.println("==================");
43         System.out.println("이름 : " + name);
44         System.out.println("사이버머니 : " + c_money);
45         System.out.println("구매품목 : " + itemName);
46         System.out.println("==================");
47
48     }
49
50 }
```

2-1. 홍길동이 mouse 10개를 구입한 이후 홍길동의 상태와 아이템의 상태를 출력하라. (ItemDriver.java)

1) 고객 (“홍길동”, “100000)을 생성하라.

2) 아이템(“mouse”, 7000, 50)을 생성하라.

```
3  public class ItemDriver {
4
5      public static void main(String[] args) {
6          Client c1 = new Client("홍길동", 100000);
7          Item1 i1 = new Item1("마우스", 7000, 50);
8          c1.buyItem(i1, 10);
9          c1.printClient();
10         i1.printItem();
11
12     }
13
14 }
```

Console

<terminated> ItemDriver [Java Application] C:₩Program Files₩Java₩jre1.8.0_201₩bin₩javaw

```
==================
이름 : 홍길동
사이버머니 : 30000
구매품목 : 마우스
==================
==================
제품명 : 마우스
가격 : 7000
재고량 : 40
매출액 : 70000
==================
```

3. 한 출판사의 도서목록에 들어가는 도서를 모델링하는 Book클래스를 설계하고 구현하라. 도서는 도서명(title), 저자(author), 가격(price), 재고량(stock)을 가진다.

1) 도서명, 저자, 가격, 재고량을 넘겨받아 초기화도 하면서 대응하는 매개변수로 변경까지도 가능한 생성자 메서드를 갖는다.

2) 도서명, 저자, 가격을 알 수 있어야 하고 값을 변경할 수 있어야 하나 외부클래스에서 직접 접근은 불가능하다.

3) 판매량(salesVolume)을 입력 받으면 판매량에 따른 매출액을 계산하고, 재고량을 조절한다. 이 때 재고량이 0개 이하이면 신판을 추가 주문한다.

4) 각 도서의 상태를 출력한다.

```java
import java.util.Scanner;

public class Book {

    private String title; // 도서명
    private String author; // 저자
    private int price; // 가격
    private int stock; // 재고량
    private int salesVolume;
    private int sales;

    public Book(String title, String author, int price, int stock) {
        super();
        this.setTitle(title);
        this.setAuthor(author);
        this.setPrice(price);
        this.setStock(stock);
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getAuthor() {
        return author;
    }

    public void setAuthor(String author) {
        this.author = author;
    }

    public int getPrice() {
        return price;
    }

    public void setPrice(int price) {
        this.price = price;
    }

    public int getStock() {
        return stock;
    }

    public void setStock(int stock) {
        this.stock = stock;
    }

    public int getSalesVolume() {
        return salesVolume;
    }

    public void setSalesVolume(int salesVolume) {
        this.salesVolume = salesVolume;
    }
```

```
62      public void buyBook(int salesVolume) {
63          Scanner scan = new Scanner(System.in);
64          System.out.println("판매량을 입력하세요 : ");
65          this.salesVolume = salesVolume;
66          salesVolume = scan.nextInt();
67          if(stock >= salesVolume) {
68              stock -= salesVolume;
69              sales += salesVolume * price;
70              if(stock == 0) {
71                  this.add();
72              }
73          }
74          else {
75              System.out.println("재고가 부족합니다.");
76          }
77
78      }
79
80      public void add() {
81          Scanner scan = new Scanner(System.in);
82          System.out.println("추가 주문량을 입력하세요 : ");
83          int n = scan.nextInt();
84          stock += n;
85      }
86
87      public void printBook() {
88          System.out.println("==========");
89          System.out.println("도서명 : " + title);
90          System.out.println("저자 : " + author);
91          System.out.println("가격 : " + price);
92          System.out.println("재고량 : " + stock);
93          System.out.println("==========");
94      }
95
96  }
```

3-1. (BookDriver.java)

1) (독서법, 채석용, 8400, 12) 인스턴스객체를 생성하고 각 값을 출력하라.

2) 가격을 10200원으로 변경하고 출력하라.

3) 책의 판매량을 10개로 설정하고 매출액을 계산하라.

```
3  public class BookEx {
4
5      public static void main(String[] args) {
6          Book b1 = new Book("독서법", "채석용", 8400, 12);
7          b1.setPrice(10200);
8          b1.printBook();
9          b1.buyBook(10);
10         b1.printBook();
11     }
12
13 }
```

Console

<terminated> BookEx [Java Application] C:₩Program Files₩Java₩jre1.8.0_201₩bi

```
==========
도서명 : 독서법
저자 : 채석용
가격 : 10200
재고량 : 12
==========
판매량을 입력하세요 : 10
매출액 : 102000
==========
도서명 : 독서법
저자 : 채석용
가격 : 10200
재고량 : 2
==========
```

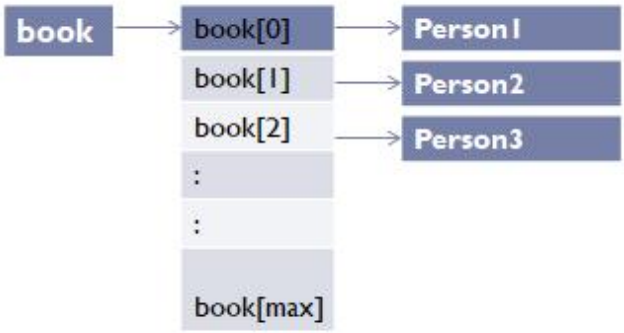
※참고

객체들의 배열

>> 지금까지의 배열은 배열 원소가 정수 혹은 실수와 같은 기본 타입이었으나, 기본값의 배열뿐만 아니라 객체의 배열도 제공한다.

```
Class Phonebook{
  private Person[] book;

  Phonebook(int max){
    book = new Person[max];
 }
```

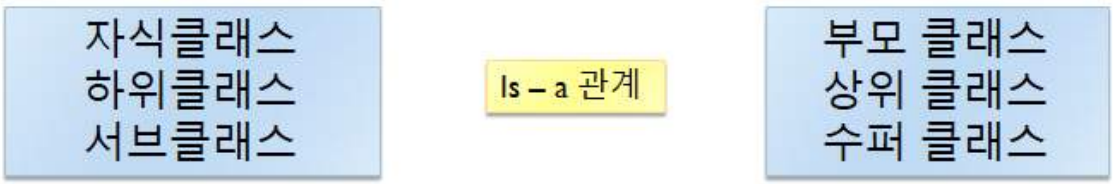


8. 상속

상속

>> 기존에 있는 클래스의 멤버 변수나 메서드를 물려받아 새로운 클래스를 만드는 것

> 코드의 재 사용성



ex) 택시는 자동차이다. (택시는 자동차를 상속받는다.)
팩스 전화기는 전화기이다. (팩스 전화기는 전화기를 상속받는다.)

> 택시는 자동차의 모든 속성과 메서드를 가지며 추가적으로 미터기 변수와 미터기를 올리고 내리는 메서드를 갖는다.

클래스명	자동차	택시(is a 자동차)	트럭(is a 자동차)
멤버변수	차기종 색상 속도 제조회사	미터기	무게
메소드	속도를 올리다 속도를 내리다 출발하다 정지하다	미터기를 올리다 미터기를 내리다	짐을 싣다 짐을 내리다

1. 상속방법

[접근제어자] class 클래스이름 extends 상위 클래스명

>> 단일 상속을 지원 : 상위 클래스는 하나만 지정한다.

>> 모든 클래스는 extends Object를 기술하지 않아도 Object클래스를 자동으로 상속받는다.

예)
```
class Car{
        String carname;
        String color="검정색";
        int velocity;
        void speedUp( ){  velocity += 5;}
        void speedDown( ){  velocity -= 5;}
}
class Truck extends Car{
        int ton;
}
```

2. 접근제어자

>> 상위 클래스의 접근제어자가 public인 경우 하위 클래스가 상속받는 데는 아무 문제가 없다. protected인 경우 같은 패키지가 아니더라도 상속된 클래스에서 접근 가능하다. 그러나 private인 경우 상속 관계라 해도 상위

클래스에 접근이 불가능하다.

상속과범위

> 메소드 호출의 기본 원칙 : 한 클래스에서 한 메소드가 호출되면

1. 클래스 내에 정의된 메소드인지 확인(있으면 클래스 내의 것을 실행)

2. 아니라면 상위 메소드에 정의된 메소드인지 확인

3. 최상위에 도달할 때까지 계속된다.

> 상위 클래스의 생성자메서드는 하위클래스에 자동으로 상속되지 않는다.

> 어떤 클래스에 생성자메서드를 정의하지 않는다면 컴파일러는 기본 생성자메서드 메소드를 추가한다.

4. 따라서 다음의 두 코드는 같은 결과이다.

super();는 상위 클래스의 생성자메서드 메소드를 호출하며, 상위 클래스의 생성자메서드 메소드가 없다면 최상위인 Object 클래스의 메소드를 호출한다.

```
class MyClass{
  public void printHello( ){
    System.out.println("Hi");
  }
}
```

```
class MyClass{
  MyClass( ){
    super( );
  }
public void printHello( ){
    System.out.println("Hi");
  }
}
```

컴파일러가 추가한 문장

상속과 생성자메서드

>> 상속과정에서 생성자 메서드는 상속되지 않는다.

>> 하위 클래스에서 객체가 생성되면 자동으로 상위 클래스의 인자없는 생성자메서드(티폴트)실행 > 상위 클래스에 인자 없는 생성자가 없으면 에러 발생

>> 상위 클래스의 생성자메서드 먼저 수행, 초기화 > 하위 클래스 초기화를 순서대로 진행

▸ 다음의 결과는?

```
class Car_1{
 Car_1(){ }
 Car_1(String name){
   System.out.println("Car 이름이 있는 생성자메서드");
}}
class Truck extends Car_1{
 Truck(){
 System.out.println("Truck 생성자메서드");
}
 public static void main(String[] args){
     Truck mytruck = new Truck();
 }
```

▸ 수정한 결과

```
class Car_1{
 Car_1(String name){
  System.out.println(name + "Car 이름이 있는 생성자");
 }
}
class Truck extends Car_1{
 Truck(){
 super("sm3");
  System.out.println("Truck 생성자");
 }
  public static void main(String[] args){
    Truck mytruck = new Truck();
  }
}
```

1. Car_1(){ }를 추가하거나
2. Car_1(){ }를 추가하지 않더라도 Super(); 메소드를 이용하여 상위 클래스의 생성자메서드를 명시적으로 호출한다.

예약어 super

>> 상위 클래스의 변수나 메소드를 참조하기 위해 사용하는 예약어

ex) super.x : 상위 클래스의 변수 x를 나타냄

super.x() : 상위 클래스의 x라는 메소드를 호출

super 메소드

>> 상위 클래스의 생성자메서드를 명시적으로 호출할 때 사용하는 거승로 하위 클래스의 생성자메서드에서 제일 먼저 호출한다.

메소드 오버라이딩

>> 상위 클래스의 메소드를 하위 클래스에서 재정의하여 사용하는 것

>> 상위 클래스 메소드의 이름, 인자, 반환형에 대해서 완전히 같아야 한다.

>> static, final, private 메소드의 경우 오버라이딩 할 수 없다.

실습예제

1. 은행계좌 Account.java를 모델링한다.
 1) 멤버변수로 계좌번호(a_number)와 잔고(balance)를 갖는다.
 2) 생성자 함수는 계좌번호와 은행 잔고로 만들어진다.
 3) 잔고를 반환한다. (getBalance)
 4) 주어진 금액을 입금한다. (deposit)
 5) 주어진 금액을 출금한다. (withdraw)
 6) 객체의 현 상태를 문자열로 반환한다. (toString)

```
3  public class Account {
4
5      protected String a_number; // 계좌번호
6      protected int balance; // 잔고
7
8      public Account(String a_number, int balance) {
9          super();
10         this.a_number = a_number;
11         this.balance = balance;
12     }
13
14     public int getBalance() {
15         return balance;
16     }
17
18     public void deposit(int deposit) {
19         balance += deposit;
20     }
21
22     public void withdraw(int withdraw) {
23         if(balance > withdraw) {
24             balance -= withdraw;
25         }
26         else {
27             System.out.println("잔액이 부족합니다.");
28         }
29     }
30
31     public String toString() {
32         return "계좌번호 : " + a_number + "잔고 : " + balance;
33     }
34
35 }
```

2. 은행계좌 (Account.java)를 상속한 저축예금 (SavingAccount.java)를 모델링하라.

1) 멤버변수로 이율을 갖는다.

2) 상위 생성자메서드를 호출하여 잔고와 이율값을 set하는 생성자메서드를 갖는다.

3) 이율을 반환한다. (getInterest)

4) 이율을 주어진 값으로 변환한다. (setInterst)

5) 객체의 현 상태를 문자열로 반환한다. (toString)

```
3  public class SavingAccount extends Account {
4
5      private double annualInterest; // 이율
6
7      public SavingAccount(String a_number, int balance, double annualInterest) {
8          super(a_number, balance);
9          this.annualInterest = annualInterest;
10     }
11
12     public double getAnnualInterest() {
13         return annualInterest;
14     }
15
16     public void setAnnualInterest(double annualInterest) {
17         this.annualInterest = annualInterest;
18     }
19
20     public String toString() {
21         return "계좌번호 : " + a_number + "잔고 : " + balance + "이율 : " + annualInterest;
22     }
23
24 }
```

실습예제

1. 어느 회사 직원들의 봉급을 계산하는 프로그램을 작성하라.

1) 이 회사에는 일반 직원과 매니저가 있고, 매니저 중에는 임원이 있다.
2) 모든 직원은 이름과 사번, 봉급을 받는다. 매니저는 봉급 외에 보너스를 받을 수 있다. 임원은 봉급과 보너스 외에 스톡옵션 (stock_option)을 받는다.
3) 직원, 매니저, 임원의 초봉은 각각 10만원, 20만원, 40만원이며 매니저, 임원의 보너스는 각각 5만원, 10만원이다. 임원의 스톡옵션은 10만원 상당의 주식이다.
4) 직급에 따라 매년 봉급 인상율이 다르며 직원, 매니저, 임원의 봉급 인상율은 각각 30%, 20%, 10%이다.

```
class Employee{
    
    protected String name; //해당 클래스와 하위클래스
    protected int salary; //급여
    protected int basic; //기본급
    
    public Employee(String name, int salary, int basic) {
        this.name = name;
        this.salary = salary;
        this.basic = basic;
    }
    
    public void getSalary() { //급여계산
        salary = basic; //모든 직원이 기본급
    }
    
    public void raiseSalary() { //급여인상 (추상메서드)
        System.out.println("급여를 인상합니다.");
    }
}
```

// 직원 클래스

```
class Staff extends Employee{
    
    private String team; //소속
    private int exe_salary; //성과금
    //메서드 오버라이딩
    
    public void getSalary() { //급여계산
        salary = basic + exe_salary;
    }
    
    public void raiseSalary() { //급여인상
        salary = (int) (basic * 1.05 + exe_salary);
    }
    
    @Override
    public String toString() {
        return "사원 [team=" + team + ", exe_salary=" + exe_salary + ", name=" + name + ", salary=" + salary
                + ", basic=" + basic + "]";
    }

    public Staff(String name, int salary, int basic) {
        super(name, salary, basic);
    }
    
    public void setTeam(String team) {
        this.team = team;
    }

    public void setExe_salary(int exe_salary) {
        this.exe_salary = exe_salary;
    }

}
```

// Staff 클래스

```
class Excutive extends Employee{
    
    private int bonus;
    //메서드 오버라이딩
    
    public void getSalary() { //급여계산
        salary = basic + bonus;
    }
    
    public void raiseSalary() { //급여인상
        salary = (int) (basic * 1.02 + bonus);
    }
    
    @Override
    public String toString() {
        return "임원 [bonus=" + bonus + ", name=" + name + ", salary=" + salary + ", basic=" + basic + "]";
    }

    public Excutive(String name, int salary, int basic) {
        super(name, salary, basic);
    }
    
    public void setBonus(int bonus) {
        this.bonus = bonus;
    }
}
```

// Excutive 클래스

```java
public class EmployeeTest {

    public static void main(String[] args) {

        /*사원생성 (나미림, 기획팀, 500000, 200000, 500000) */
        Staff s1 = new Staff("나미림", 500000, 500000);
        s1.setExe_salary(200000);
        s1.setTeam("기획팀");
        System.out.println(s1.toString());
        s1.raiseSalary();
        System.out.println(s1.toString());

        /*임원생성 (신잎새, 1000000, 1000000, 5000000)*/
        Excutive e1 = new Excutive("신잎새", 1000000, 1000000);
        e1.getSalary();
        e1.setBonus(100000);
        System.out.println(e1.toString());
        e1.raiseSalary();
        System.out.println(e1.toString());
    }

```

// main 클래스

2. 2차원 공간의 점을 나타내는 Point 클래스를 설계하고 작성하라.

1) 멤버변수로 x, y좌표를 갖는다. (x_coordinate, y_coordinate)
2) 생성자 함수는 (0,0)으로 초기화하거나, 특정 (x, y)로 초기화하는 2가지 종류를 갖는다.
3) x좌표를 반환하는 메소드 (getX)를 갖는다.
4) y좌표를 반환하는 메소드 (getY)를 갖는다.
5) x좌표를 주어진 값으로 변경하는 메소드 (setX)를 갖는다.
6) y좌표를 주어진 값으로 변경하는 메소드 (setY)를 갖는다.
7) 좌표값을 (x, y)의 형태로 출력하는 메소드 (printAttr)를 갖는다.

```
30 class Point{
31     
32     private double x_coordinate;
33     private double y_coordinate;
34     
35     public Point(double num1, double num2) {
36         x_coordinate = num1;
37         y_coordinate = num2;
38     }
39     
40     public Point() {
41         x_coordinate = 0;
42         y_coordinate = 0;
43     }
44     
45     public void setX(double x_coordinate) {
46         this.x_coordinate = x_coordinate;
47     }
48 
49     public void setY(double y_coordinate) {
50         this.y_coordinate = y_coordinate;
51     }
52     
53     public double getX() {
54         return x_coordinate;
55     }
56     
57     public double getY() {
58         return y_coordinate;
59     }
60     
61     public void printAttr(double x, double y){
62         System.out.println("(" + x +" , " + y + ") ");
63     }
64 
65 //  public void move(double num1, double num2) {
66 //      x_coordinate += num1;
67 //      y_coordinate += num2;
68 //  }
69     
70 }
```

// Point 클래스

```java
import java.util.Scanner;

public class PointEx {

    public static void main(String[] args) {

        double x_coordinate, y_coordinate;
//      double move_x, move_y;
        Scanner scan = new Scanner(System.in);

        System.out.print(" x좌표와 y좌표의 값을 입력하세요 : ");
        x_coordinate = scan.nextDouble();
        y_coordinate = scan.nextDouble();

        Point p1 = new Point(x_coordinate, y_coordinate);
        p1.printAttr(x_coordinate, y_coordinate);

//      System.out.print("증가시킬 x값와 y의 값을 입력하세요 : ");
//      move_x = scan.nextDouble();
//      move_y = scan.nextDouble();

//      p1.move(move_x, move_y);

    }
}
```

// main 클래스

9. 추상클래스와 인터페이스

추상클래스

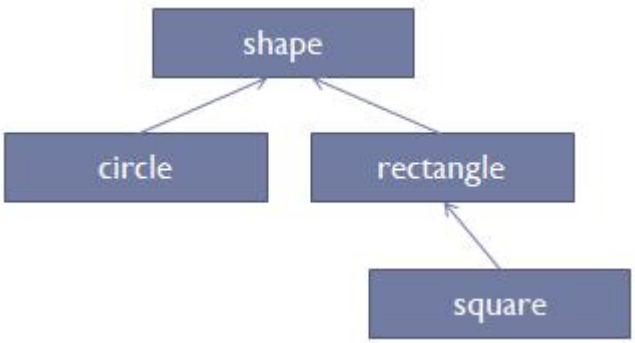
>> 클래스 계층구조에서 일반적인 개념을 나타내는 클래스로 하나 이상의 추상메서드를 포함한다. (abstract 키워드를 사용하여 표시한다.)

>> 추상 메서드는 메서드 머리부분만 몸체는 없는 메서드이다.

>> 추상 메서드를 가진 클래스로 new 연산자를 이용해 객체를 생성할 수 없다.

>> 추상 클래스의 추상메서드는 반드시 오버라이딩 되어야하기 때문에 하위 클래스들이 특정 메서드를 반드시 구현하도록 강제할 수 있다.

>> 추상 메서드를 오버라이딩 하지 않으면, 상속받는 클래스도 자동으로 추상 클래스가 된다.



> 2차원 도형의 일반적인 클래스
> 면적과 둘레를 가지나 그 값들을 구할 수 없다. (상속받는 도형들마다 구하는 공식이 다르기 때문)
> 멤버변수 : 도형의 이름
> 추상메서드 : getArea()

면적을 구하는 메서드가 있다는 정도만 기술하고 구체적인 방법은 자식, 손자 클래스에서 구현한다.

ex) shape 예제

```
3  public abstract class Shape { //추상클래스 만들기
4      String name; //도형의 이름
5      static int cnt; //몇개 만들었는지
6
7      public Shape() {
8          super();
9          cnt++;
10     }
11
12     public Shape(String name) {
13         super();
14         this.name = name;
15         cnt++;
16     }
17
18     public String getName() {
19         return name;
20     }
21
22     public void setName(String name) {
23         this.name = name;
24     }
25
26     public static int getCnt() {
27         return cnt;
28     }
29
30     public abstract void getArea();
31
32 }
```

// Shape 클래스

```java
public class Circle extends Shape {

    private final double PI = 3.14;
    private int radius;

    public Circle(String name, int radius) {
        super(name);
        this.radius = radius;
    }

    public void getArea() { // 오버라이딩 해야 함
        double area;
        area = radius * radius * PI;
        System.out.println(this.name + "의 넓이 : " + area + "cm^2");
    }

}
```

// Circle 클래스

```java
public class Rectangle extends Shape {

    private int width;
    private int height;

    public Rectangle(String name, int width, int height) {
        super(name);
        this.width = width;
        this.height = height;
    }

    public void getArea() {
        int area;
        area = width * height;
        System.out.println(this.name + "의 넓이 : " + area + "cm^2");
    }
}
```

// Rectangle 클래스

```
public class ShapeTest {

    public static void main(String[] args) { // Shape로는 객체 생성 불가능
    //  Shape s1 = new Shape();
    //  Shape s2 = new Shape("Triangle");
        Circle c1 = new Circle("원", 5);
        c1.getArea();
        printCnt();
        Circle c2 = new Circle("원", 7);
        c2.getArea();
        printCnt();
        Rectangle r1 = new Rectangle("사각형", 5, 4);
        r1.getArea();
        printCnt();
        Triangle t1 = new Triangle("삼각형", 5, 4);
        t1.getArea();
        printCnt();

    }

    public static void printCnt() {
        System.out.println("만들어진 도형의 개수 : " + Shape.getCnt());
    }

}
```

Console

<terminated> ShapeTest [Java Application] C:₩Program Files₩Java₩jre1.8.0_201₩bin₩javaw.exe (2(

```
원의 넓이 : 78.5cm^2
만들어진 도형의 개수 : 1
원의 넓이 : 153.86cm^2
만들어진 도형의 개수 : 2
사각형의 넓이 : 20cm^2
만들어진 도형의 개수 : 3
삼각형의 넓이 : 10.0cm^2
만들어진 도형의 개수 : 4
```

// main 클래스

1. 클래스 계층 구조에서 일반적인 개념을 나타내는 클래스
2. 하나 이상의 추상메서드를 포함한다.
3. 추상클래스로부터 new 연산자를 이용해 객체를 만들 수 없다.
4. 추상클래스는 보통의 메서드와 변수를 포함 할 수 있다.
5. 추상클래스의 추상메서드는 반드시 오버라이딩 된다.
6. 만약 추상메서드를 오버라이딩하지 않으면 상속받는 클래스는 자동으로 추상클래스가 된다.
7. 추상메서드 형식

[접근제어자] abstract 리턴타입 메서드이름(); // { }가 없다.

인터페이스

>> 상수(final static)와 추상메소드 선언의 집합 > 추상클래스의 부분집합
>> 추상 클래스보다 완벽한 추상화 제공 > 추상클래스는 추상 메소드, 멤버변수, 일반 메소드를 갖지만 인터페이스는 추상 메서드만 갖는다.
>> 클래스라고 부르지 않을 뿐 클래스로 이해하자

왜 언제 사용하나?
> 협업시 사용, 다중 상속의 유사한 기능을 흉내낼 수 있으며 현재의 클래스가 이미 다른 클래스로부터 상속을 받고 있는 상태이면서, 또 다른 클래스의 요소들이 필요할 때 사용 -> loose coupling(느슨한결합)

loose coupling(느슨한결합) : 다른 클래스를 직접적으로 사용하는 클래스의 의존성을 줄인 결합이다. 코드의 재사용성과 유연성을 위해 강한 결합보다는 느슨한 결합이 좋다.
- 하위 클래스의 코딩내용(바디)에 대해서는 간섭하지 않겠다. > 느슨한
- 단, 결합으로 인해 메서드 명과 파라미터는 동일하게 두겠다. > 결합

형식
> pubilc interface 인터페이스이름 [extends 인터페이스이름] {
    상수선언 ...
    pubilc 메서드이름(인자들);
  }

인터페이스를 클래스에 구현하는 방법

형식
> class 클래스이름 implements 인터페이스명 { }

인터페이스에 정의된 접근제어자는 public을 사용하여야 하는 이유
> 인터페이스에 정의된 메서드는 반드시 상속받은 하위 클래스에서 오버라이딩되어 사용되기 때문

실습예제

TV와 Radio 클래스가 Sound 인터페이스를 implements했기 때문에 SoundUp() 메소드와 SoundDown() 메소드를 오버라이딩하였다.

교과서 96p SoundExam.java

```
3  public interface Sound {
4      public void SoundUp(int level);
5      public void SoundDown(int level);
6  }
7
8  class TV implements Sound{
9      private int SndLevel;
10     public TV(){
11         SndLevel = 0;
12     }
13     public void SoundUp(int level) {
14         SndLevel += level;
15     }
16     public void SoundDown(int level) {
17         SndLevel -= level;
18         if(SndLevel < 0) SndLevel = 0;
19     }
20 }
21
22 class Radio implements Sound{
23     private int SndLevel;
24     public Radio(){
25         SndLevel = 0;
26     }
27     public void SoundUp(int level) {
28         SndLevel += level;
29     }
30     public void SoundDown(int level) {
31         SndLevel -= level;
32         if(SndLevel < 0) SndLevel = 0;
33     }
34 }
35
36 class SoundExam{
37     public static void main(String[] args) {
38         Sound radio = new Radio();
39         Sound tv = new TV();
40         radio.SoundUp(5);
41         tv.SoundUp(5);
42     }
43 }
```

교과서 97p Advanced.java

인터페이스도 일반 클래스처럼 또 다른 인터페이스를 상속받을 수 있다.

```
3  interface Sound {
4      public void SoundUp(int level);
5      public void SoundDown(int level);
6  }
7
8  interface AdvancedSound extends Sound {
9      public void SoundOff();
10 }
```

실습예제

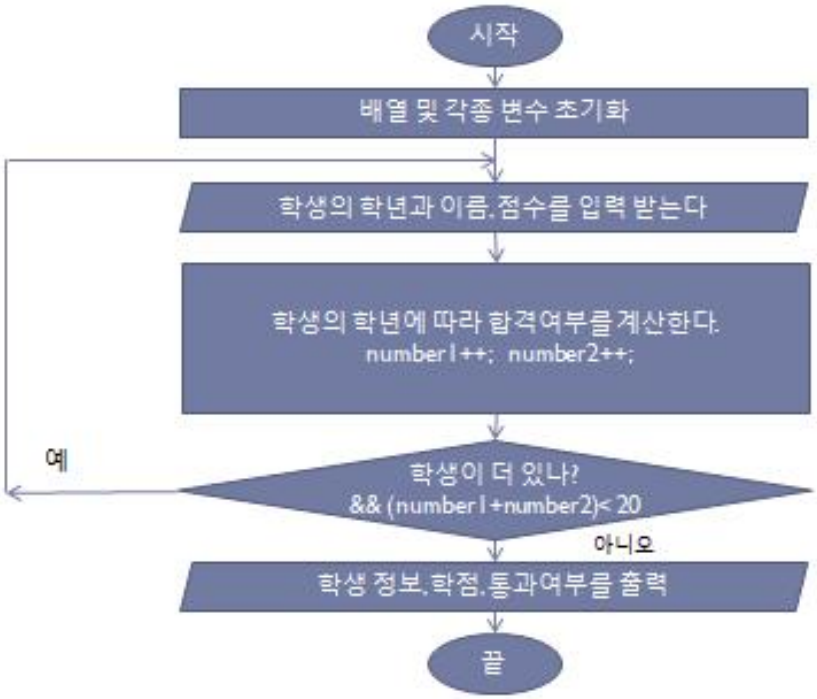
1. 자바 교과목의 성적을 처리하고자 한다. (GradeTest.java)
 1) 교과목을 수강하는 학생은 1,2학년이다.
 2) 각 학생은 이름, 점수, 학점을 갖는다.
 3) 1학년은 70점 이상이면 통과, 2학년은 80점 이상이면 통과이다.
 4) 최대 수강인원은 20명이다. 각 학생의 점수와 이름을 입력받아 학점을 계산 후 출력한다.
 5) 3개의 클래스와 1개의 인터페이스로 설계하라.

인터페이스 : ComputeGrade
부모클래스 : Student
자식클래스 : FirstGrade, SecondGrade
Student class : Student 객체 생성, 이름을 주어진 값으로 변경, 점수를 주어진 값으로 변경, 이름을 반환, 점수를 반환
FirstGrade class : 학점을 계산한다, 통과 여부를 출력한다.
SecondGrade class : 학점을 계산한다, 통과 여부를 출력한다.

학생 배열 : roster1, roster2
1학년 학생 수 : number1
2학년 학생 수 : number2
최대 수강생 수 : max
입력점수 : score
입력이름 : name

학년 : grade
추가처리여부 : answer

GradeTest class (main)의 흐름도



```java
package student.interface_test;
import java.util.ArrayList;

class Student{
    protected String name; // 이름
    protected int score; // 점수
    protected String grade; // 학점
}

interface ComputeGrade{
    void input(); // 이름, 점수 입력
    void calGrade(int score); // 학점 계산
    void print_grade(); // 학점 출력
    void print_all(); // 전체 학생 명단 출력
    void print_passer(); // 통과 학생 명단 출력
}

class FirstGrade extends Student implements ComputeGrade{
    
    Scanner scan = new Scanner(System.in);
    static ArrayList<String> roster1 = new ArrayList<>(); // 전체 학생 명단
    static ArrayList<String> pass1 = new ArrayList<>(); // 통과 학생 명단

    public void input() {
        
        System.out.println("\n최대 수강인원은 20명 입니다.");
        System.out.println("입력을 종료하시려면 0번을 입력해주세요. \n");
        
        for(int i = 0; i < 20; i++) {
            System.out.print(i+1 +"번 학생 이름을 입력하세요 : ");
            name = scan.next();
            roster1.add(name); // 1학년 학생 명단 추가
            
            if(name.equals("0")) {
                System.out.print("입력을 종료합니다. \n"); break;
                
            } else {
                System.out.print("점수를 입력하세요 : ");
                score = scan.nextInt();
                calGrade(score);
                
                if(score >= 70) {
                    pass1.add(name); // 통과 명단 추가
                }
                
                print_grade();
            }
            
        }

    }

}
```

```
54⊖    @Override
55     public void calGrade(int score) {
56         if(score >= 70) {
57             grade = "통과";
58             
59         } else {
60             grade = "불통과";
61         }
62     }
63 
64⊖    @Override
65     public void print_grade() {
66         System.out.println("========성적처리결과========");
67         System.out.println("1학년 " + name + "학생");
68         System.out.println("점수 : " + score + "점");
69         System.out.println("통과여부 : " + grade);
70         System.out.println("========================");
71     }
72     
73⊖    @Override
74     public void print_all() {
75         System.out.println("========1학년 전체 학생 명단========");
76         
77         for(int i = 0; i < roster1.size(); i++) {
78             System.out.println(roster1.get(i));
79         }
80         
81         System.out.println();
82         
83     }
84 
85⊖    @Override
86     public void print_passer() {
87         System.out.println("========1학년 통과 학생 명단========");
88         
89         for(int i = 0; i < pass1.size(); i++) {
90             System.out.println(pass1.get(i)); // 통과 명단 출력
91         }
92         
93         System.out.println();
94     }
95 
96     
97 }
```

```
class SecondGrade extends Student implements ComputeGrade{
    
    Scanner scan = new Scanner(System.in);
    static ArrayList<String> roster2 = new ArrayList<>(); // 전체 학생 명단
    static ArrayList<String> pass2 = new ArrayList<>(); // 통과 학생 명단

    public void input() {
        
        System.out.println("\n최대 수강인원은 20명 입니다.");
        System.out.println("입력을 종료하시려면 0번을 입력해주세요. \n");
        
        for(int i = 0; i < 20; i++) {
            System.out.print(i+1 +"번 학생 이름을 입력하세요 : ");
            name = scan.next();
            roster2.add(name); // 2학년 학생 명단 추가
            
            if(name.equals("0")) {
                System.out.print("입력을 종료합니다. \n"); break;
                
            } else {
                System.out.print("점수를 입력하세요 : ");
                score = scan.nextInt();
                calGrade(score);
                
                if(score >= 80) {
                    pass2.add(name); // 통과 명단 추가
                }
                print_grade();
            }
            
        }
        
    }

    @Override
    public void calGrade(int score) {
        if(score >= 80) {
            grade = "통과";
            
        } else {
            grade = "불통과";
        }
    }

    @Override
    public void print_grade() {
        System.out.println("========성적처리결과========");
        System.out.println("2학년 " + name + "학생");
        System.out.println("점수 : " + score + "점");
        System.out.println("통과여부 : " + grade);
        System.out.println("========================");
    }
```

```
151 ¤¶
152⊖ »   @Override¤¶
△153 »   public void print_all() {¤¶
154 »   »   System.out.println("========1학년 전체 학생 명단========");¤¶
155 »   »   ¤¶
156 »   »   for(int i = 0; i < roster2.size(); i++) {¤¶
157 »   »   »   System.out.println(roster2.get(i));¤¶
158 »   »   }¤¶
159 »   »   ¤¶
160 »   »   System.out.println();¤¶
161 »   »   ¤¶
162 »   }¤¶
163 ¤¶
164⊖ »   @Override¤¶
△165 »   public void print_passer() {¤¶
166 »   »   System.out.println("========2학년 통과 학생 명단========");¤¶
167 »   »   ¤¶
168 »   »   for(int i = 0; i < pass2.size(); i++) {¤¶
169 »   »   »   System.out.println(pass2.get(i)); // 통과 명단 출력¤¶
170 »   »   }¤¶
171 »   »   ¤¶
172 »   »   System.out.println();¤¶
173 »   }¤¶
174 »   ¤¶
175 »   ¤¶
176 }¤¶
```

```java
public class GradeTest {

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int grade_option;

        while(true) {

            System.out.println("---------성적처리프로그램---------");
            System.out.println("성적을 처리할 학생의 학년을 입력해주세요.");
            System.out.println("종료하시려면 0번을 입력해주세요.");
            System.out.println("-----------------------------");
            System.out.println("1. 1학년");
            System.out.println("2. 2학년");
            System.out.println("3. 1학년 통과한 학생 명단 확인하기 ");
            System.out.println("4. 2학년 통과한 학생 명단 확인하기 ");
            System.out.println("5. 1학년 전체학생 명단 확인하기 ");
            System.out.println("6. 2학년 전체학생 명단 확인하기 ");
            System.out.println("0. 종료하기");
            System.out.print(">> ");
            grade_option = scan.nextInt();

            if(grade_option == 1 || grade_option == 3 || grade_option == 5) {
                FirstGrade f1 = new FirstGrade();

                if(grade_option == 1) {
                    f1.input();

                } else if(grade_option == 3) {
                    f1.print_passer();

                } else {

                }

            } else if(grade_option == 2 || grade_option == 4 || grade_option == 6) {
                SecondGrade s1 = new SecondGrade();

                if(grade_option == 2) {
                    s1.input();

                } else if(grade_option == 4){
                    s1.print_passer();

                } else {

                }

            } else if(grade_option == 0) {
                System.out.println("성적처리 프로그램을 종료합니다.");
                break;

            } else {
                System.out.println("숫자를 잘못입력하셨습니다. ");
                System.out.println("숫자를 다시 입력해주세요. \n");
            }

        }

    }

}
```