chapter 1.자바를 시작하기 전에

ch1-1,2 자바란?

- 컴퓨터 프로그램을 만드는데 사용
- JRE + JDK + API
- 객체지향 + 함수형

ch1-3 자바의 특징

-객체지향 언어, 자동메모리 관리,
멀티스레드(프로그램 동시진행), 운영체제에 독립적

ch1-4 JVM

-자바 프로그램이 실행되는 가장 컴퓨터
JVM 설치만 되어있으면 어디서든 실행가능

ch1-5 JDK 자바 개발도구

-자바로 프로그램을 개발하기위해서는 JDK가 필요하다.

ch1-6 JAVA API

-Java API란? java로 프로그램을 만드는데 필요한 주요 기능을 미리 만들어서 제공
-Java API 문서란? javaAPI가 제공하는 기능에 대한 상세한 정보 제공(html)
-Java API 문서의 설치
oracle.com에서 압축파일을 다운받아서 압축해제

ch1-7이클립스 단축키

ctrl+D 한줄 삭제
ctrl+alt+down 행단위 복사
alt+up,down 행단위 이동
ctrl+i 자동들여쓰기 맞춤

ch1-8 git, gitHub

git : 프로그램 등의 소스 코드 관리를 위한 분산 버전 관리 시스템
svn은 보통 저장소가 서버에 있으며, git은 저장소가 자신의 컴퓨터에 있다.
github : git을 호스팅해주는 웹 서비스이며, git 저장소 서버를 대신 유지 및 관리해주는 서비스다.

chapter 2.변수

ch2-1변수란?하나의 값을 저장할 수 있는 메모리 공간
-하나의 값을 저장할 수 있는 메모리 공간
1. 변수 선언 이유 - 값을 저장할 공간 마련
2. 변수의 선언 방법 - 변수타입 변수이름;
3. 변수에 값 저장 - int age = 25;
4. 변수의 초기화 - 변수에 처음으로 값을 저장하는 것
(지역변수는 반드시 읽기전에 반드시 초기화 해야함)

ch2-2 변수의 타입 (기본형과 참조형)

-변수의 타입은 저장할 값의 타입에 의해 결정된다.
<기본형>
기본형은 실제 값을 저장
참조형은 메모리 주소를 저장

자료형	데이터	메모리 크기	표현 가능 범위
boolean	참과 거짓	1 바이트	true, false
char	문자	2 바이트	모든 유니코드 문자
byte	정수	1 바이트	-128 ~ 127
short		2 바이트	-32768 ~ 32767
int		4 바이트	-2147483648 ~ 2147483647
long		8 바이트	-9223372036854775808 ~ 9223372036854775807
float	실수	4 바이트	$\pm(1.40\times10^{-45} \sim 3.40\times10^{38})$
double		8 바이트	$\pm(4.94\times10^{-324} \sim 1.79\times10^{308})$

ch2-3 변수, 상수, 리터럴

상수 - 한번만 값을 저장 가능한 변수
리터럴(값) - 그 자체로 값을 의미하는 것

ch2-3 리터럴의 접두사와 접미사

정수형 long타입은 L , 실수형 float타입은 f를 붙인다.

ch2-4 문자와 문자열

- String 타입은 클래스라 객체생성을 해야 하지만 자주 쓰이기 때문에 바로 쓰는 것을 허용한다.
- String = " "; //빈 문자열, char ch = ''; //에러

ch2-5 두 변수의 값 교환하기

빈 컵이 필요.. int x = 10, y = 20; int tmp; //빈컵
tmp = x; x=y; y = tmp;

ch2-6 printf()의 지시자

지시자	설명
%b	불리언(boolean) 형식으로 출력
%d	10진(decimal) 정수의 형식으로 출력
%o	8진(octal) 정수의 형식으로 출력
%x, %X	16진(hexa-decimal) 정수의 형식으로 출력
%f	부동 소수점(floating-point)의 형식으로 출력
%e, %E	지수(exponent) 표현식의 형식으로 출력
%c	문자(character)로 출력
%s	문자열(string)로 출력

%n = |n (둘다 개행기능, %n이 보편적)

ch2-7 화면에서 입력받기 - Scanner

Scanner란? 화면으로부터 데이터를 입력받는 기능제공
1.import
2.객체생성 Scanner scanner = enw Scanner (System.in);
3.객체사용 int num = scanner.nextInt();
(화면에서 입력받은 정수를 num에 저장)
String input = scanner.nextLine(); //한행을 input에 저장
int num = Integer.parseInt(input)//문자열 -> 숫자

ch2-17 타입간의 변환방법

1.문자와 숫자간의 변환 -> 3 + '0' = '3' , '3'-'0' = 3
2.문자열로 변환 -> 3 + "" = "3"
3.문자열을 숫자로 -> Integer.parseInt("3");
4.문자열을 Double로 -> Double.parseDouble("3.4")
5.문자열 문자로-> "3".charAt(0) - '0'

chapter 3. 연산자

ch3-1 연산자 우선순위와 결합법칙

1. 산술 > 비교 > 논리 > 대입. 대입은 제일 마지막에 수행된다.
2. 단항(1) > 이항(2) > 삼항(3). 단항 연산자의 우선순위가 이항 연산자보다 높다.
3. 단항 연산자와 대입 연산자를 제외한 모든 연산의 진행방향은 왼쪽에서 오른쪽이다.

ch3-2 증감 연산자

타입	설명	사용예
전위형	값이 참조되기 **전에** 증가시킨다.	j = ++i;
후위형	값이 참조된 **후에** 증가시킨다.	j = i++;

증감 연산자가 독립적으로 쓰였을 경우 차이가 없다.

<전위형>

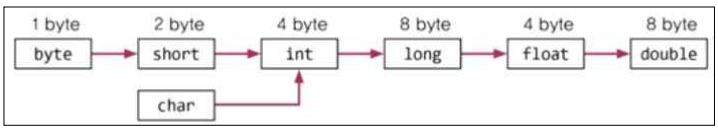
```
j = ++i; // 전위형   ⟶   ++i;   // 증가 후에
                         j = i; // 참조하여 대입
```

<후위형>

```
j = i++; // 후위형   ⟶   j = i; // 참조하여 대입 후에
                         i++;   // 증가
```

ch3-4 자동 형변환

큰 값에서 작은 값으로 변환할 때 수동 형 변환 필요



ch3-5 산술 변환

-연산 전에 피연산자의 타입을 일치시키는 것

1.두 피연산자의 타입을 같게 일치 시킴

2.피연산자의 타입이 int보다 작은 타입이면 int로 변환

ch3-6 반올림 -Math.round()

실수를 소수점 첫 째 자리에서 반올림해 정수로 반환

```
long result = Math.round(4.52);  // result에 5가 저장된다.
```

Math.round()함수에 10의제곱을 이용.. 곱하고 나누워 원하는 자릿수까지 반올림한 수를 구할 수 있다.

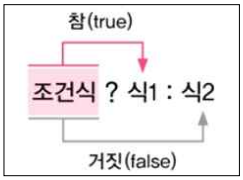
```
(Math.round(pi*1000)/1000.0); // 3.142
```

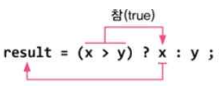
ch3-7 비교연산자

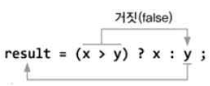
비교연산자	연산결과
>	좌변 값이 크면, true 아니면 false
<	좌변 값이 작으면, true 아니면 false
>=	좌변 값이 크거나 같으면, true 아니면 false
<=	좌변 값이 작거나 같으면, true 아니면 false

문자열 비교에는 == 대신 equals()를 사용한다.

ch3-8 조건연산자 (삼항 연산자 if문을 간단히..)







ch3-9 복합 대입 연산자

i +=3;	i = i + 3;
i -= 3;	i = i - 3;

chapter 4. 조건문과 반복문 (제어문)

ch4-1 조건문과 반복문

조건문(if, switch)조건을 만족 할 때만 {}을 수행

반복문(for, while)조건을 만족하는 동안 {}을 수행

ch4-2 조건식의 다양한 예

조건식	조건식이 참일 조건
90 <= x && x <= 100	정수 x가 90이상 100이하일 때
x < 0 || x > 100	정수 x가 0보다 작거나 100보다 클 때
x%3==0 && x%2!=0	정수 x가 3의 배수지만, 2의 배수는 아닐 때
ch=='y' || ch=='Y'	문자 ch가 'y' 또는 'Y'일 때
ch==' ' || ch=='\t' || ch=='\n'	문자 ch가 공백이거나 탭 또는 개행 문자일 때
'A' <= ch && ch <= 'Z'	문자 ch가 대문자일 때
'a' <= ch && ch <= 'z'	문자 ch가 소문자일 때
'0' <= ch && ch <= '9'	문자 ch가 숫자일 때
str.equals("yes")	문자열 str의 내용이 "yes"일 때(대소문자 구분)
str.equalsIgnoreCase("yes")	문자열 str의 내용이 "yes"일 때(대소문자 구분안함)

ch4-3 if-else문, if-else-if문, 중첩 if문

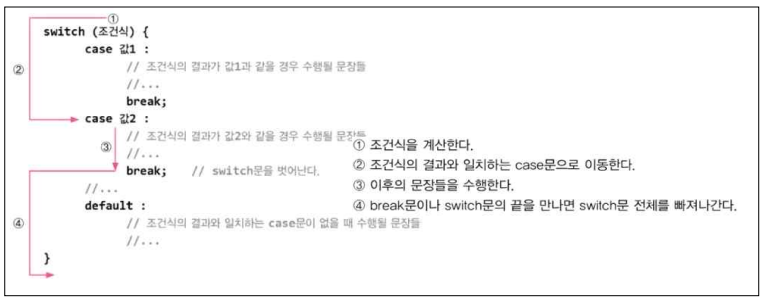
if-else문 - 둘 중 하나가 참일 때와 거짓일 때로 나눠서 처리

if-else-if문 - 여러 개중 하나 참일 때 하나만 실행 하고 if-else-if 문을 벗어난다.

중첩 if문 - if문 안의 if문(조건 세분화)

ch4-4 switch문

처리에야 하는 경우의 수가 많을 때 유용한 조건문



<switch와 if-else문 비교>

1.switch 문은 조건식이 정수, 문자열 이지만 if-else문의 조건식은 t/f이다.

2.switch 문은 조건식이 하나, if-else문은 조건식이 여러 개이다.

3.switch 문은 항상 if-else문으로 바꿀 수 있지만 if-else는 항상 바꿀 수 있는 것은 아니다. ()

ch4-5 switch문의 제약조건

1.switch문의 조건식 결과는 정수, 문자열 이어야한다.

2.case문의 값은 정수 상수(문자포함),문자열만 가능하며 중복되지 않아야 한다. (변수X,실수X)

ch4-6 임의의 정수(난수) 만들기 Math.random()

math,random()-0.0과 1.0사이의 임의의 double값 반환

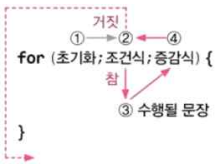
0.0 <= Math.random() < 1.0

1부터 10까지 난수 만들기 (10개)

1. 각 변에 10을 곱한다.(구하고 싶은 숫자 개수를 곱함)
2. 각 변을 int로 형 변환 한다.
3. 각 변에 1을 더한다.

ch4-7 for문 (반복 횟수 알 때)

조건을 만족하는 동안 {}를 반복




```
        1부터   5까지   1씩 증가
for(int i=1;i<=5;i++) {
    System.out.println("I can do it.");
}
```

ch4-8 중첩 for문

for문안에 for문을 넣을 수 있다. (구구단, 별찍기)

ch4-9 while문(반복 횟수 모를 때)

조건을 만족하는 동안 {}를 반복, 초기화 필요

```
while (조건식) {
    // 조건식의 연산결과가 참(true)인 동안, 반복될 문장들을 적는다.
}
```

for문과 while문은 서로 100% 바꿔 쓸 수 있다.

ch4-9 do-while문

while문의 블록을 앞으로 뺀 것.(사용자 입력을 받을 때 유용)

입력을 받고 맞는지 확인

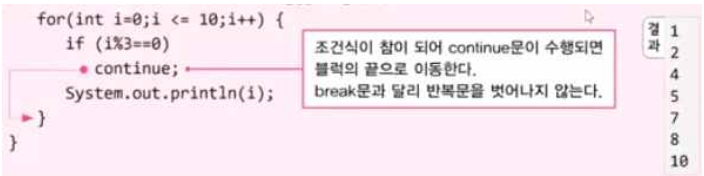
```
do {
    // 조건식의 연산결과가 참일 때 수행될 문장들을 적는다.(처음 한 번은 무조건 실행)
} while (조건식); ← 끝에 ';'을 잊지 않도록 주의
```

ch4-10 break문

자신이 포함된 하나의 반복문을 벗어난다.

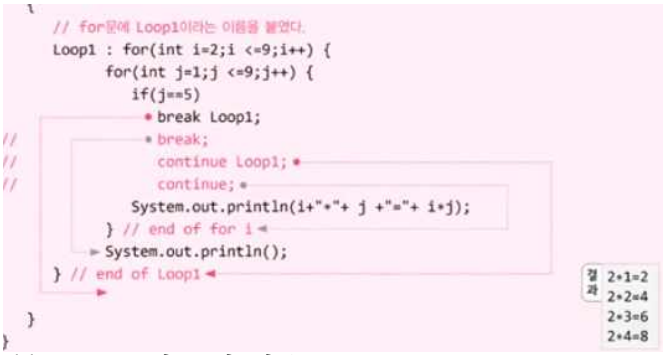
ch4-11 continue문

자신이 포함된 반복문의 끝으로 이동->다시 반복문으로
전체 반복 중에서 특정 조건시 반복을 건너 뛸 때 유용
특정 조건이 만족시 continue이후 실행X



ch4-12 이름붙은 반복문

반복문에 이름을 붙여서 하나 이상의 반복문을 빠져나옴
(break는 하나의 반복문만 빠져나올 수 있음)



//continue에는 잘 안씀