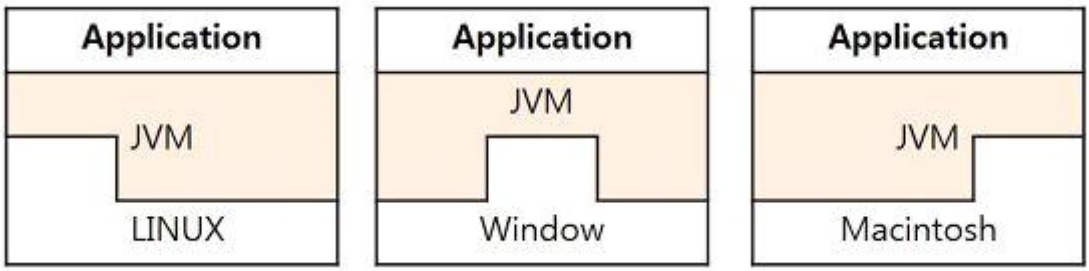
1주차(1)

자바에서의 계층적 구조를 쉽게 표현하려면 플로우차트를 이용하는 것 보다는 UML기법을 활용하는 것이 좋다. UML은 점선, 실선등으로 관계도를 표시하는 기법. 인터넷 찾아보던가

1주차: 기본문법(자료형,연산자,제어문 등), 클래스와객체(기본클래스), 상속
2주차: 추상화(추상클래스, 인터페이스, 내부클래스), 제네릭타입, 컬렉션

(List, Map, Set), 예외처리, 스레드
3주차: IO(File클래스, byte base, charater base)
4주차: 네트워크(채팅프로그램), JDBC

자바란?
썬 마이크로시스템즈 소속 제임스고슬링 등의 일부 연구진들은 '그린프로젝트'라는 이름으로 '오크(Oak)'라는 언어를 개발하고 있었다.
오크는 오디오, TV, 세탁기 등 각각의 가전제품을 제어하는 통합된 언어로써 개발중이었지만 결국 목적을 달성하지 못하고 실패로 돌아간다.
그 무렵 웹(www)이 급속도로 발전하게 되고, 이에 발맞추고자 썬에서는 오크의 명칭을 Java로 바꾼뒤 서로 다른 컴퓨터(OS - operating System(운영체제))사이에서 호환성과 이식률을 높인 언어로 발전시켰다.



JDK설치하기
SE = 스텐다드 에디션
EE = 엔터프라이즈 에디션
우리는 SE를 받는다.

www.oracle.com접속

DOWNLOAD탭에 마우스 오버 - 왼쪽에 Java for Developers클릭 -
Java Platform, Standard Edition에 JDK Download클릭 -
Java SE Development Kit 7u17의 권한 수락 - PC버전에 맞는 파일 다운로드

설치 후 C:\Program Files\Java\jdk1.7.0_17\bin(자바 설치경로)를 복사.
내 컴퓨터 우측클릭 - 속성 - 고급시스템 설정 - 환경변수 -
시스템변수(S)탭에서 Path찾아 더블클릭 - 변수값(V)의 내용 맨 뒤에 ;을 붙인 후 C:\Program Files\Java\jdk1.7.0_17\bin붙여넣기

PC시작버튼 - 실행 - cmd - (cd\하면 c드라이브 최상위로 이동. 안해도 됨ㅋ) - java -version입력 (자바 버전이 제대로 나오면 성공)확인 후 - javac 입력(뭔가 주르륵 나오면 패스설정까지 성공)

www.eclipse.org 접속해서 이클립스 다운로드ㅋ

자료형(기본자료형) valueType

논리형 : boolean - 1bit (기본값 false)
문자형 : char - 2byte (기본값 \u0000 or 0) ____ 0 ~ 65,535
정수형 : byte - 1byte (기본값 0) ____ -128 ~ 127
short - 2byte (기본값 0) ____ -32,768 ~ 32,767

int - 4byte (기본값 0) ____ -21,4748,3648 ~ 21,4748,3647(21억)

- 일반적으로 가장많이 쓰이는 자료형.

- but 주식시장이나 증권 관련 분야에서는 long을 사용하는 경우가 더 많음

long - 8byte (기본값 0) ____ -9,223,372,036,854,775,808 ~ (900경)
실수형 : float - 4byte (기본값 0.0)
double - 8byte(기본값 0.0)

위에서부터 한 개씩 알아봅시다~
그 전에!!!!! 자료형 사용에는 규칙이 필요하다.

변수선언 규칙 :
[자료형] 변수명; (선언)

변수명 = 값; (대입)
[자료형] 변수명 = 값(선언과 대입을 동시에. 초기화)

숫자가 먼저 들어가면 안된다.
_를 제외하고 특수기호가 포함될 수 없다.
예약어 금지(switch, while 등)
의미있는 단어로 이름짓자(숫자 num, 이름 name등)
한글은 사용하지 말 것.

[변수 표기법]
- 카멜 표기법
"단봉낙타" 표기법
각 단어의 첫문자를 대문자로 표기하고 붙여쓰되, 맨처음 문자는 소문자로 표기함
띄어쓰기 대신 대문자로 단어를 구분하는 표기 방식
예시: backgroundColor, typeName

- 파스칼 표기법
첫 단어를 대문자로 시작하는 표기법(클래스나 인터페이스등에 사용하는 형태)
예시: BackgroundColor, TypeName, PowerPoint

- 헝가리안 표기법
변수 선언시 접두어를 붙여 변수의 의미를 명확히 파악하게 하기 위한 표기법
예시 : boolean bTest = true;
int nTest = 0;
short sTest = 0;
float fTest = 0;
String sTest = “안녕”;

- 스네이크 표기법
여러단어로 이루어진 경우 단어 사이에 _를 붙여 구별하는 표기법
예시 : num_test, snake_case

논리형

논리형은 true, false 즉, 사실이다와 사실이 아니다의 두 가지 값만을 가진다.

```
boolean b = true;
System.out.println("b의 값 : " + b); //+기호는 더한다는 의미가 아닌 이어붙인다의 의미.
                                     //단, 숫자 사이의 + 기호는 더하기를 의미.
boolean b = 1;//자료형의 값이 올바르지 않아 오류
```

문자형

```
char ch = 'A'; //문자형은 홑따옴표 안에 넣어야 하며 한글자이상 넣을 수 없다.
System.out.println("ch = " + ch); //결과 : A
char ch2 = '\u0041'; //문자열이 들어간 것 같지만, 유니코드로 A를 의미하는 한글자가 들


//어간 것.
System.out.println("ch2 = " + ch2); //결과 : A

char ch3 = 65 + 1; //아스키코드 65에 + 1
System.out.println("ch3 = " + ch3); //결과 : B
```

정수형

```
byte b = 128; //byte자료형의 표현범위를 벗어나므로 오류가 난다.

byte b = 127;
short s = 32767;
int n = 550;

System.out.println("b = " + b); //결과 127
System.out.println("s = " + s); //결과 32767
System.out.println("n = " + n); //결과 550
```

실수형(소수)

```
float f = 3.14; //java에서 실수는 기본적으로 double형으로 인식하기 때문에 float자료형을

//사용한다는 것을 명시해줘야 한다. (3.14f)

float f1, f2;
f1 = 3.14f;
f2 = 150; //실수에도 정수 대입이 가능. 실수에 정수를 대입하면 자동 실수화 됨. 결과보자

System.out.println("f1 = " + f1); //결과 3.14
System.out.println("f2 = " + f2); //결과 150.0
```

기본자료형은 이것으로 끝.
다음에 배울 연산자를 포함하여, 자바 개발에서 몰라서는 안될 중요한 개념이 있다.
바로 캐스팅!! 일명 형변환이라 한다.

캐스팅(형변환)

1. 프로모션

- 큰 자료형에 작은 자료형을 대입하는 것(자동으로 이루어짐)

```
double d = 100.5; //8byte
int n = 200; //4byte
d = n;
System.out.println("d = " + d); //결과 : 200.0
```

---------- ex1

```
char c = 'A'; //2byte
long l = 100;//8byte
l = c;
System.out.println("l = " + l); //결과 65
```

---------- ex2

2. 디모션

- 작은 자료형에 큰 자료형을 대입하는 것(자동으로 이루어지지 않음)

```
char c = 'B'; //2byte
int n = c + 1; //여기까지는 프로모션 캐스팅
c = n; //c는 2byte, n은 4byte이므로 오류 발생
c = (char)n; //이렇게 수정
System.out.println("c = " + c);
```

---------- ex1

```
float f = 5.5f;
int n = 0;
n = (int)f; //같은 4byte여도 자료형이 일치하지 않으면 대입되지 않음. 고로 캐스팅
System.out.println("n = " + n);
//결과는 5 인데, float에서 int로 캐스팅되면서 소수점 이하 자리를 상실함
```

---------- ex2

짚고 넘어갈 자바의 장점(신기함)

```
byte b1 = 100;
byte b2 = 20;
byte b3 = b1 + b2; //오류남.

int b3 = b1 + b2; //이렇게 수정
```

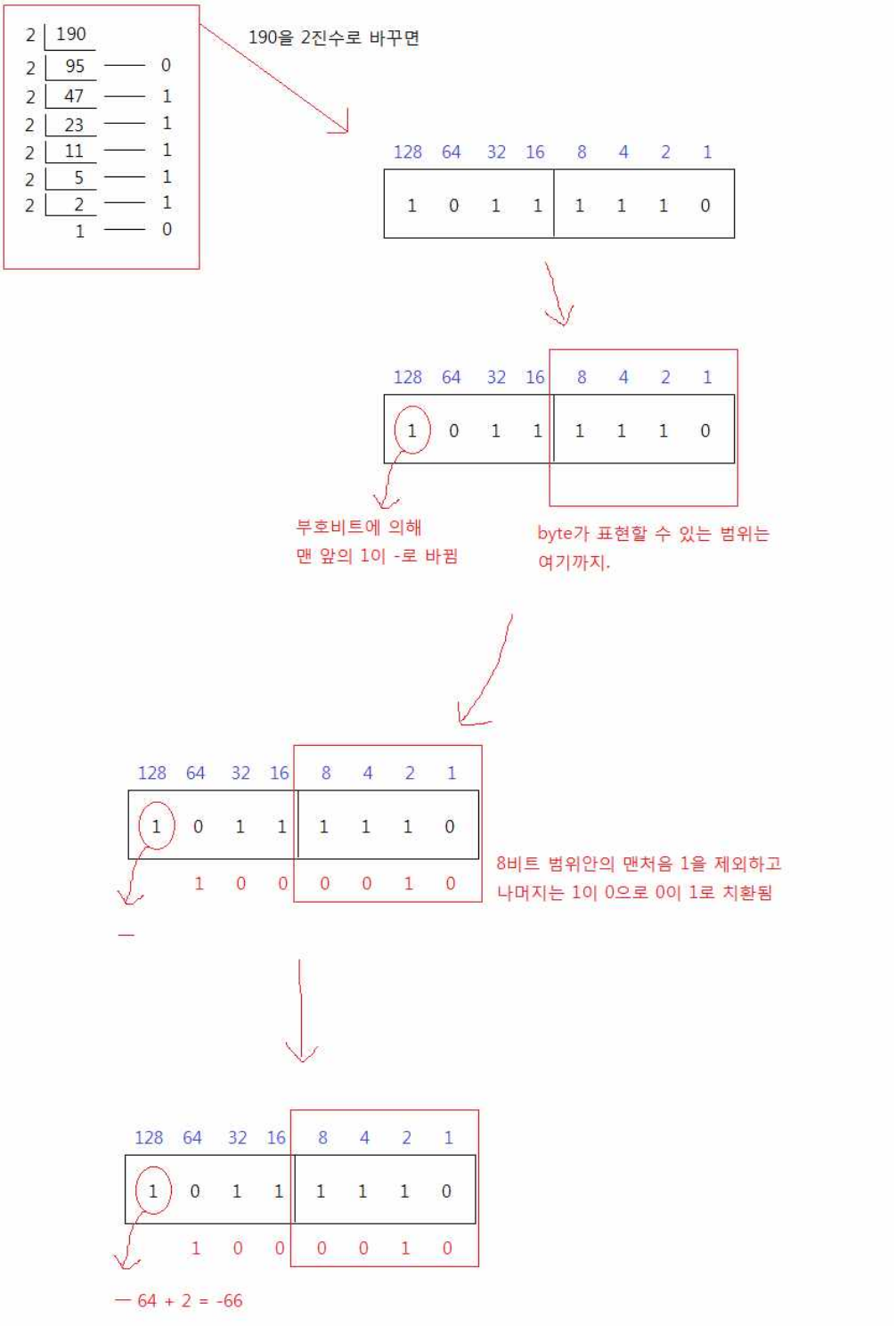
byte의 표현 범위가 127까지 밖에 되지 않다보니, byte끼리의 연산은 127을 넘어가버릴 가능성이 높다.
이런 상황을 대비하여 java개발자들은 byte끼리의 연산이 수행될 때, int형 변수로 값을 받도록 만듦.

---

byte범위를 넘어가는 연산임에도 byte로 캐스팅 한 경우

```
byte b1 = 100;
byte b2 = 90;
byte b3 = (byte) (b1 + b2);
System.out.println(b3);//결과 -66
```

이유는 아래쪽 그림으로 설명(필요시)


2 | 190
2 | 95 —— 0
2 | 47 —— 1
2 | 23 —— 1
2 | 11 —— 1
2 | 5 —— 1
2 | 2 —— 1
1 —— 0
190을 2진수로 바꾸면
128 64 32 16 8 4 2 1
1 0 1 1 1 1 1 0
부호비트에 의해
맨 앞의 1이 -로 바뀜
byte가 표현할 수 있는 범위는
여기까지.
1 0 0 0 0 1 0
8비트 범위안의 맨처음 1을 제외하고
나머지는 1이 0으로 0이 1로 치환됨
—
— 64 + 2 = -66

연산자(Operator)

1. 최고연산자 : . , ()
2. 증감연산자 : ++ , --
3. 산술연산자 : + , - , * , / , %(모듈러. 나머지 연산자)
   (10 / 3 = 3 <-- 요놈은 몫을 구한다.
     10 % 3 = 1 <-- 요놈은 나머지를 구한다.)
4. 시프트 연산자 : >> , << , >>>
5. 비교연산자 : > , < , >= , <= , == , !=
6. 비트연산자 : & , | , ^ , ~
7. 논리연산자 : && , || , !
8. 조건(삼항)연산자 : ? , :
9. 대입연산자 : = , *=, /= , %= , += , -=

산술 연산자.

산술 연산자는 4칙연산과 나머지 값을 구하는 연산자로 나뉜다.

```
int n1, n2, n3;
n1 = 20; //n1에20을대입
n2 = 7; //n2에7을대입
n3 = n1 + n2; //n1 + n2의값을n3에대입
System.out.println("n3 = " + n3); //결과27

n3 = n1 - n2;
System.out.println("n3 = " + n3); //결과 13

n3 = n1 / n2;
System.out.println("n3 = " + n3); //결과 2 - 몫 출력

n3 = n1 % n2;
System.out.println("n3 = " + n3) //결과 6 - 나머지 출력
```

대입 연산자.

앞에서 많이 사용했듯이 특정 값을 변수에 전달하여 기억시킬 때 사용하는 연산자.

```
int n1 = 10; //n1이라는 int형 변수에 10이라는 정수를 대입함.
int n2 = 7;
System.out.println("=연산자: n1 = " + n1 + ", n2 = " + n2);



int n3 = 13;
int n4 = 15;
System.out.println("+=연산자: n3 += n4 = " + (n3 += n4)); //n3 = n3 + n4을 줄여서 씀
```

```
int n5 = 10;
int n6 = 3;
System.out.println("/=연산자: n5 /= n6 = " + (n5 /= n6)); //n5 = n5 / n6를 줄여서 씀
```

비교 연산자.

비교 연산자는 변수나 상수의 값을 비교하여 참과 거짓을 판단하는 연산자.

그러므로 결과값은 반드시 true나 false로만 반환된다.

그럼 결과를 확인하려면 어떤 자료형으로 받아야 할까요??

```
int n1 = 10;
int n2 = 20;
boolean result;
result = n1 < n2;
//한줄로하면boolean result = n1 < n2;
System.out.println("n1 < n2 : " + result);

result = n1 == n2;
System.out.println("n1 == n2 : " + result);

result = n1 != n2;
System.out.println("n1 != n2 : " + result);
```

논리 연산자.

비교 연산자를 통한 연산이 2개 이상 필요할 때 사용한다.

```
int myAge = 30;
int limit = 35;

//&&은 앞쪽의 연산이 false일때 뒤쪽연산을 수행하지 않고 넘어간다.
//&&는and의 뜻. '~하고'라는 의미로 이해하면 도움이 된다.
//둘 다 참일때만 참
boolean result = (limit - myAge) >= 5 && myAge > 30;
System.out.println("&&연산결과: " + result);
```

```
int n1 = 10;
int n2 = 20;
//||은앞쪽의연산이false여도 뒤쪽연산을수행한다.
//||는or의 뜻. '~거나'라는의미로이해하면도움이된다.
//한쪽만 참이어도 참
boolean result2 = (n1 += 10) > 20 || n2 - 10 == 11;
System.out.println("||연산결과: " + result2);

//! 는not의 뜻. true는 false로, false는 true로 바꿔서나타낸다.
System.out.println("!연산결과: " + !result2);
```

비트 연산자.

논리 연산자와 유사하지만 bit단위(2진수)의 연산만 가능하다.

일반적으로 다음에 배울 시프트 연산자와 더불어 암호화, 복호화 작업에 사용되며.

나는 아직까지 실무에서 써본 적이 없다.

그러나 몰라서 못하는 것과, 아는데 안하는 것은 다르니 일단 알고 넘어가자.

```
int a = 10; //1010
int b = 7; //0111
int c = a & b; //논리곱(and) - 2진수로바꿨을 때 두값이모두1 일때만결과가1. 나머지는0
System.out.println("c : " + c);

int a2 = 12;
int b2 = 8;
int c2 = a2 | b2; //논리합(or) - 2진수로바꿨을 때 두값이모두0일때만결과가0. 나머지는1
System.out.println("c2 : " + c2);

int a3 = 9;
int b3 = 11;
int c3 = a3 ^ b3; //배타적or(xor) - 2진수로바꿨을때 두값이 서로같을때는0.서로다를때는1
System.out.println("c3 : " + c3);
```

시프트 연산자.

역시 bit단위의 연산을 수행하지만 오른쪽 또는 왼쪽으로 이동시켜 값에 대한 변화를 준다.

```
int a = 12; //1100
int b = 2;



int c = a >> b; //b만큼 오른쪽으로 이동
System.out.println("c : " + c);



int d = c << b;
System.out.println("d : " + d);



char ch = 'F'; //1000110
int num = 1;
char ch_result = (char)(ch >> num);
System.out.println(ch_result);
//F는 아스키 코드로 70. 인터넷으로 아스키 코드표 보면서 설명 ㄱㄱ
//시프트 연산으로 1만큼 bit를 이동시키면 35에 해당하는 #이 출력된다.



//시프트 연산을 통해 오른쪽 혹은 왼쪽으로 이동시키면
//정보가 잘려나간 곳은 공백으로 표시된다.
//이런 공백을 처리하는 것도 시프트 연산자인데,
//솔직히 나도 실전에서 써본적이 없어 잘 모르겠다;;
```

증감 연산자.
1씩 증가시키거나 1씩 감소시키는 연산자.
비교적 쉽다.
선행증감과 후행증감의 차이점만 알아두자

```
int a = 10;
System.out.println("a : " + ++a); //결과 11


int b = 10;
System.out.println("b : " + b++); //결과 10
//여기까지 일단 결과 보여주고 아래쪽 System.out.println()써서 확인시켜주자.


//여기서는 값이 증가되어 있다.
//b++연산을 수행 한뒤 대기하다가 다시 b를 만났을때 증가된 값을 출력했기 때문.
System.out.println("b++ : " + b); //결과 11

//이렇게 보면 쓸모 없어보이는 것 같아도 정말 많이 쓰이는 연산자 중 하나.
//반복문 들어가면 쓸 일 많아진다.
```

삼항 연산자(조건 연산자).
하나의 조건을 정의하여 그 조건이 참일 경우 true를, 거짓일 경우 false값을 얻어내기 위한 연산자.

```
int a = 10;
int b = 15;
boolean result;

result = ++a >= b ? true : false;
System.out.println("result :" + result);


int n1 = 10;
int n2 = 20;
char result2;
result2 = (n1 += n1) == n2 ? 'O' : 'X';
System.out.println("result2 : " + result2);
//삼항연산의 값을 받을 변수의 자료형과 ?뒤의 결과값의 타입이 같아야 한다.
```

자료형, 연산자 문제

```
int a = 10;
int b = 12;
//++a >= b || 2 + 7 <= b && 13 - b >= 0 && (a += b) - (a % b) > 10;
```

오류나므로 주석처리 하고 코드 작성없이 결과 도출해 보라 하자

```
//풀이
int a = 10;
int b = 12;
boolean result;
result = ++a >= b || 2 + 7 <= b && 13 - b >= 0 && (a += b) - (b % a) > 10;
System.out.println(result);
```

결과 = true

문제1

```
/*
* 과수원이 있다.
*
* 배, 사과, 오렌지를 키우고 있는데 하루에 생산되는 양은 각각
* 5,  7,   5개.
*
* 과수원에서 하루에 생산되는 총 개수를 출력하고, 시간당
* 전체 과일의 평균 생산 갯수를 출력.
* 평균값을 담는 변수는 float으로 할 것.
*/

//풀이
int pear = 5; //배
int apple = 7; //사과
int orange = 5; //오렌지


int fruitTotal = pear + apple + orange; //하루생산량
System.out.println("하루에 생산되는 과일 수 : " + fruitTotal + "개");


float average = fruitTotal / 24f; //시간당 평균
```

```
System.out.println("시간당 평균 생산 갯수 : " + average + "개");
```

-- 문제2