JAVA

megapie2021@gmail.com

강사 및 학생의 인물 화면은 개인 초상권으로 보호 받으므로
임의로 동영상을 찍어서 유포시 민형사 책임이 따를 수 있습니다.

본 수업자료를 외부에 공개, 게시하는 것을 금지합니다.

megapie2021@gmail.com

Java

- 썬 마이크로 시스템즈에서 1995년 개발한 객체지향 프로그래밍 언어
- 창시자는 제임스 고슬링
- 2010년 오라클이 썬 마이크로시스템즈를 인수
- 2019년 1월부터 유료화 정책 강화
- OpenJDK : GPL2 , Oracle JDK :상업 라이선스
- 자바 에디션

  Java SE(Java Standard Edition): 표준 에디션

  Jakarta EE, 구 Java EE(Java Enterprise Edition) : 서버페이즈 특화

  Java ME(Java Micro Edition): PDA 셋톱박스, 센서 등 임베디드 시스템

  JavaFX : 데스크톱 애플리케이션 개발 및 배포위한 에디션 GUI라이브러리제공

https://namu.wiki/w/Java

megapie2021@gmail.com

Java에 관하여

◆ **JVM : java virtual Machine**
java 바이트 코드가 실행될 수 있는 runtime 환경을 제공하는 가상 머신
JVM = Java 바이트 코드를 실행하기 위한 유일한 런타임 환경.

◆ **JRE : java runtime environment**
JVM을 구현 한 Java 실행환경
JRE = JVM (Java Virtual Machine) + 애플리케이션 실행을 위한 라이브러리

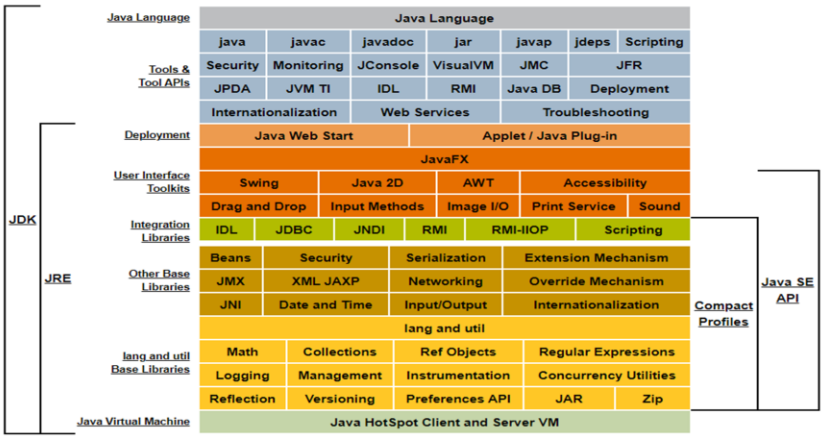
◆ **JDK** : Java Development Kit
Java 프로그램을 컴파일, 디버그 및 실행하는 데 필요한 모든 도구, 실행 파일 및 바이너리를 포함
JDK = JRE (Java Runtime Environment) + 개발 도구

◆ **https://www.linkedin.com/pulse/understanding-difference-between-jdk-jre-jvm-important-kumar**

◆ **https://www.geeksforgeeks.org/jvm-works-jvm-architecture/**

megapie2021@gmail.com

자바 플랫폼



Jdk11버전부터 JRE가 jlink 로 jdk안에 포함
(따로 설치 필요 없음)

https://docs.oracle.com/javase/8/docs/index.html

megapie2021@gmail.com

Dynamic Binding / Linking

◆ **Binding**

- Static Binding
  - 컴파일 시에 어떤 클래스의 어떤 메소드가 호출되는지 정한다.
- Dynamic Binding
  - 실행 중에 어떤 클래스의 어떤 메소드가 호출되는 지 정한다.

megapie2021@gmail.com

유연성과 가벼움

Dynamic Binding / Linking

◆ **Linking**

- Static Linking
  - 라이브러리나 다른 메소드 호출 시 실행파일에그부분을 합침.
  - 실행파일크기가 커짐 / 호출성능 향상.

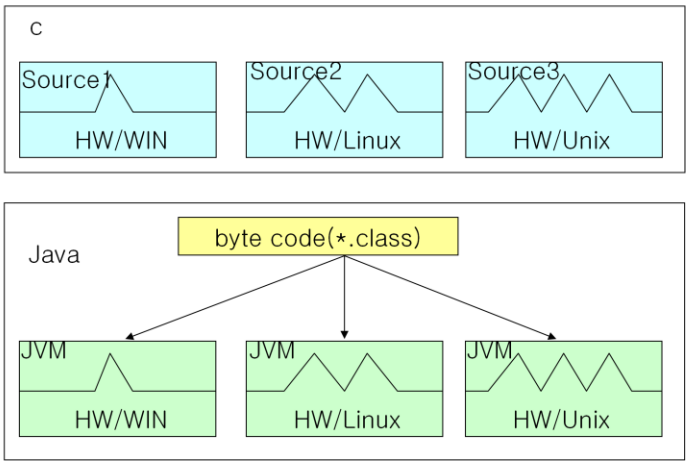
- Dynamic Linking
  - 호출하는 부분에 메소드를 호출한다는 표시만 한다. 실행되는 부분은 다른 파일에 저장.
  - 실행파일크기가 작아짐 / 실행 중 동적으로 찾아야 함으로 시간이 걸린다.
  - 실행파일 내부에 호출할 부분의 주소가 아닌 호출한 부분의 정확한 이름을 기재해야 이후에 찾을 수 있다.(주소는 무의미)

◆ **자바는 완벽한 Dynamic Binding / Dynamic Linking 지원**

megapie2021@gmail.com

플랫폼 독립

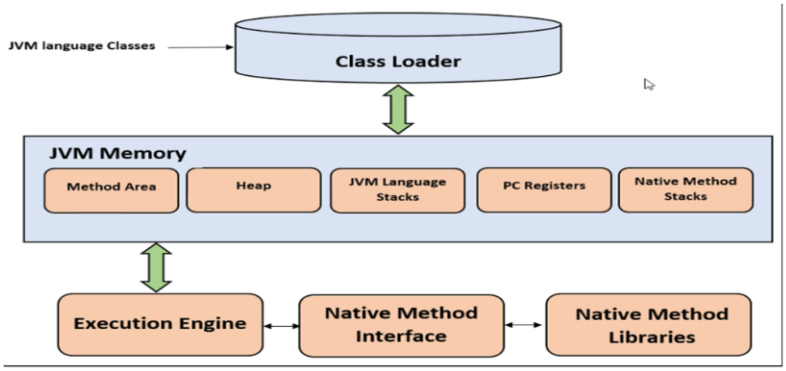
◆ 자바는 VM을 통해서 플랫폼 독립을 제공한다. (자바의 가장 큰 장점)



megapie2021@gmail.com

JVM의 구조와 메모리 모델

JVM의 내부 구조

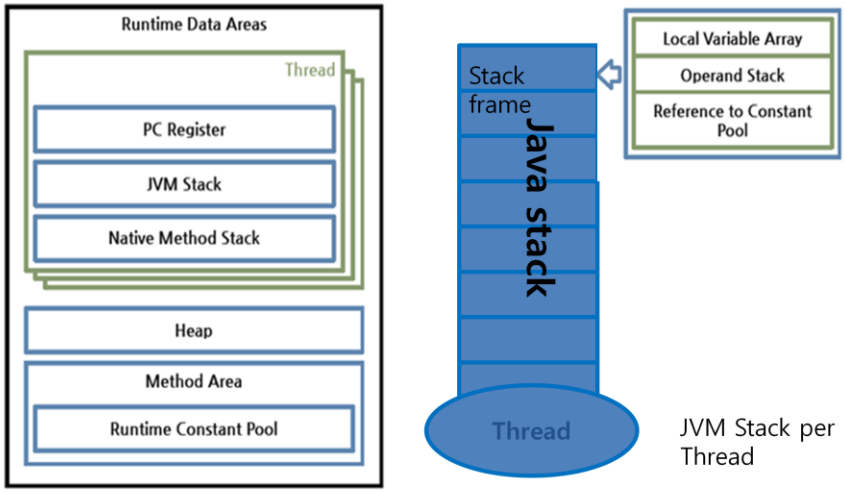


Java virtual machine stack :매개변수, 지역변수
Method Area : 메서드의 바이트코드, 클래스변수
Heap : 객체 Method영역과 Heap은 모든 쓰레드에 공유

https://howtodoinjava.com/wp-content/uploads/2018/05/JVM-Architecture.png

megapie2021@gmail.com

Runtime Data Area



https://www.cubrid.org/blog/3826357

megapie2021@gmail.com

수업을 위한 설치환경

- JDK8 or JDK11
  oracle.com -> products -> software:java
  oracle JDK , oracle Open JDK

- Eclipse 2019-12 R 4.14

megapie2021@gmail.com

다양한 출력문

System.out.print();
System.out.println();
System.out.printf()

```
System.out.printf("%s %d₩n","hong",200);
System.out.printf("%d %d₩n", 10,20);
System.out.printf("%.2f",12.1252);
```

megapie2021@gmail.com

변수

◆ 변수 (variable) – 값이 변하는 것이 가능함
◆ 데이터 저장 위함
◆ 데이터 저장 타입 변수명;

◆ 다양한 변수 표현
타입 변수명
타입 변수명1, 변수명2;
타입 변수명 =초기값;

```
int su1;
int su1, su2;
int su1=20;
int su1=20, su2, su3=30;
....
```

megapie2021@gmail.com

변수 명명 규칙

- 하나 이상의 글자로 이루어져야 한다.
- 첫 번째 글자는 문자이거나 '$', '_'여야 한다.
- 두 번째 이후의 글자는 숫자, 문자, '$', '_'여야 한다.
- '$', '_' 외의 특수 문자는 사용할 수 없다.
- 길이의 제한은 없다.
- 키워드는 식별자로 사용할 수 없다.
- 상수 값을 표현하는 단어 true, false, null은 식별자로 사용할 수 없다.

megapie2021@gmail.com

자바 프로그램 작성의 기초 - 데이터 타입

◆ 자바 데이터 타입의 분류 체계

primitive type(기본 타입)

숫자형(numeric type) - 정수형 : byte, short, int, long, char
부동소수점 : float, double

불리언(boolean type) -boolean

reference type (참조타입)

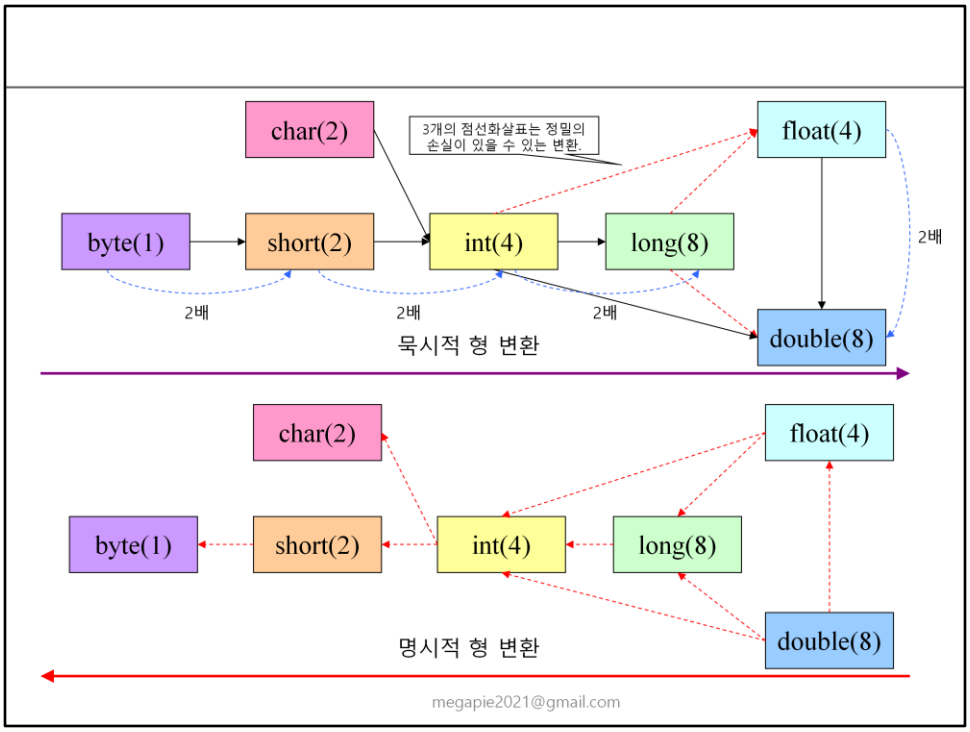
class

interface

배열(array)

열거형(enum type)

megapie2021@gmail.com


char(2)
3개의 점선화살표는 정밀의
손실이 있을 수 있는 변환.
float(4)
byte(1)
short(2)
int(4)
long(8)
2배
2배
2배
2배
double(8)
묵시적 형 변환
char(2)
float(4)
byte(1)
short(2)
int(4)
long(8)
double(8)
명시적 형 변환
megapie2021@gmail.com

연산자

◆ 캐스트 연산자(cast)
- 타입을 변환하는 연산
- 강제 형변환

사용 예
(float)10.2 , (int) 20.4

◆ 강제 형변환 vs 자동형변환

megapie2021@gmail.com

primitive type(프리미티브 타입) : 기본 타입

◆ 정수 타입의 표현 가능 범위

데이터 타입	크기	표현 범위
byte	1 바이트	-128 ~ 127
short	2 바이트	-32768 ~ 32767
int	4 바이트	-2147483648 ~ 2147483647
long	8 바이트	-9223372036854775808 ~ 9223372036854775807

megapie2021@gmail.com

로컬변수(local variable)

- 지역변수 or local variable(로컬변수)
- 일부 지역에서만 한정하여 사용하는 변수
- 메서드 안에서 선언 사용 – 그 지역공간 사용

```
class VariableTest1
  {
    int age;  // 지역변수가 아님
    public void prt( int su)   // su, su2 지역변수
    {
          int su2;
     }
  }
```

megapie2021@gmail.com

로컬변수(local variable)

◆ 로컬 변수 초기화 문제

```
class VariableTest2{
    public static void main(String args[]) {
        int num; //초기화를 하지 않음
        System.out.println(num);
    }
}
```

megapie2021@gmail.com

Local variable scope(지역변수의 범위)

```
class VariableTest3{
    public static void main(String args[]) {
        short su1= 12;
        System.out.println(su1);
        double su2 = 12.75;
        System.out.println(su2);
        char ch = 'A';
        System.out.println(ch);
    }
}
```

megapie2021@gmail.com

변수의 범위

```
class VariableTest4{
    public static void main(String args[]) {
      int su1=10;
        {
          int su2 = 10;
          System.out.println("su1:"+su1);
          System.out.println("su2 :" +su2);
        }

       System.out.println("su1: "+su1);
       System.out.println("su2 :"+su2);

    }
}
```

megapie2021@gmail.com

final 변수 : 상수

◆ **final 변수**

- final 변수 : 변수에 값을 딱 한번만 대입할 수 있는 변수
- 변할 수 없음 : 상수

```
class VariableTest5{
    public static void main(String args[]) {
    final int su=100;
     su=200;
     System.out.println("su :"+su);
}
}
```

```
class VariableTest6{
  public static void main(String args[])
{
    final int su;
     su=100;
     su=200;
     System.out.println("su :"+su);
}
}
```

megapie2021@gmail.com

주석

◆ **주석 (comment) : 프로그램의 실행에 영향을 미치지 않게 만들어진 텍스트**

```
/*
 *   클래스 이름  : Sample
 *   버전정보
 *    작성일 : 2021.01.01
 *    작성자 : 홍길동
 *    프로그램 설명 : 이름을 출력하는 프로그램
 */


public class Sample {

public static void main(String[] args) {
       String name="hong gil dong"; // 변수 name에 hong gil dong 값을 저장
        System.out.println("hello :"+name);  //출력
}
}
```

megapie2021@gmail.com

단순 대입연산자(assignment operator)

- **변수에 데이터를 담는 명령**
- **표현**
  - 변수 = 식
    jumsu=kor+eng;
  - 변수= 값
    jumsu=10;
  - num1=num2=10; // num1=(num2=10);

megapie2021@gmail.com

자바 프로그램 작성의 기초 - 연산자

◆ 단항 연산자 : 부호연산자 ,증가 연산자, 감소연산자

◆ 이항 연산자 : 사칙연산자, 비교연산자, 논리연산자, 대입연산, 복합대입연산
비트연산자

◆ 조건연산자

megapie2021@gmail.com

단일연산자- 부호연산자

```
int su=10;
int su2=-su;
System.out.println("su :"+su);
System.out.println("su2: "+su2);
`
```

megapie2021@gmail.com

단일 연산- 증감연산자

- ++ : 증가연산자 : ++ x == x=x+1
- -- : 감소연산자 : --x == x=x-1
- ++x x++ 과 --x x--

- ++x : 증가연산자 이면서 전위연산
  y=++x == x=x+1
  y=x
- x-- : 감소연산자 이면서 후위 연산자
  y=x-- == y=x
  x=x-1

```
int x=10;
int y=++x;
System.out.println("x :"+x);
System.out.println(" y:"+y);
```

```
int x=10;
int y=x++;
System.out.println("x :"+x);
System.out.println(" y:"+y);
```

megapie2021@gmail.com

증감연산 시 유의사항

◆ 복잡한 연산식에 ++, -- 연산자를 사용

```
int su1=10, su2=20;
int result;

result=++su1 + su1++ + su2--;
System.out.println("su1 = " + su1);
System.out.println("su2 = " + su2);
System.out.println("result = " + result);`
```

megapie2021@gmail.com

이항연산자 - 사칙연산자

- + (더하기)
- -(빼기)
- * (곱하기)
- / (나누기)
- % (나머지)

```
int su1=10, su2=2;
int tot=su1+su2;
int sub=su1-su2;
int mul=su1*su2;
int div=su1/su2;
int quo=su1%su2;

System.out.println(tot);
System.out.println(sub);
System.out.println( mul);
System.out.println(div);
System.out.println(quo);
```

megapie2021@gmail.com

형변환

◆ byte a1=5;
int a2=20;
System.out.println(a1+a2); // a1+a2 : 타입 int

◆ long s1=20; // int 값을 long으로 형변환

◆ int su1=20;
long su2=30L;
System.out.println(su1+su2); //su1+su2: long타입

◆ int s1=10;
double s2=20.2;
System.out.println(s1+s2) ; // s1+s2 : double타입

megapie2021@gmail.com

형변환에서 오류사항

- byte a=10;
  byte b=20;
  byte c=a+b;
  System.out.println(c);

- float c=20.2;

- byte a=20;
  byte b=-a;
  System.out.println(b);

megapie2021@gmail.com

연결연산자

◆ 연결연산자 : 문자열을 연결하여 새로운 문자열 만드는 연산

◆ 문자열 + 문자열 => 문자열
◆ 숫자 + 문자열 => 문자열
◆ 문자열+ 숫자 => 문자열

megapie2021@gmail.com

연결연산자

◆ 문자열 연결 연산자

```
System.out.println("결과는 "+3+4);
```

vs

```
System.out.println(3+4+"결과는 ");
```

megapie2021@gmail.com

비교 연산자

- 비교 연산자 : 두 수의 크기를 비교하는 연산자
- > (크다), < (작다) , >=(크거나 같다), <=(작거나 같다)
- ==(같다) !=(같지 않다)

```
int su=10;
float su2=10.2f;

System.out.println(su>su2);
System.out.println(su>su2);
System.out.println(su>=su2);
System.out.println(su<=su2);
System.out.println(su==su2);
System.out.println(su!=su2);
```

megapie2021@gmail.com

논리 연산자

- 논리 연산자 : boolean 값들을 가지고 논리식을 만드는 연산자
- &(and) |(or) ^(XOR) !(not)
- &&(and) || (or) : 최적화된 and , 최적화된 or

```
boolean t1=true;
boolean t2=false;

System.out.println(t1&t2);
System.out.println(t1&&t2);
System.out.println(t1|t2);
System.out.println(t1||t2);
System.out.println(t1^t2);
System.out.println(!t1);
```

megapie2021@gmail.com

논리연산자

- AND/OR 연산자
- 다음 코드의 작동방식에 대해 생각해 보자

```
1 == 2  &  3<4

1 == 2  && 3<4
```

megapie2021@gmail.com

비트연산자

◆ 데이터를 구성하는 각 비트를 가지고 and, or, xor, not 연산을 수행
◆ &(and) |(or) ~(not) ^(xor)

[비트 AND 연산]

피연산자1 \ 피연산자2	1	0
1	1	0
0	0	0

[비트 OR 연산]

피연산자1 \ 피연산자2	1	0
1	1	1
0	1	0

[비트 XOR 연산]

피연산자1 \ 피연산자2	1	0
1	1	1
0	1	0

[비트 NOT 연산]

피연산자	결과
1	0
0	1

megapie2021@gmail.com

비트 연산자 : 데이터 구성 비트를 가지고 AND, OR, XOR, NOT 연산을 수행하는 연산자

shift 연산자

- 각 데이터 비트를 왼쪽 또는 오른쪽으로 이동시키는 것
- 값 or 변수 << 비트수
  왼쪽으로 비트수 만큼 이동시키고 빈공간은 0으로
- 값 or 변수 >> 비트수
  오른쪽으로 비트수 만큼 이동시키고 빈 공간을 MSB(Most Significant Bit 최고값을 갖는 비트)은 비트로 채움
- 값 or 변수 >>> 비트수
  오른쪽으로 비트수 만큼 이동하고 빈 공간은 0으로 채움

```
int a=-20;
System.out.println(a>>3);
System.out.println(a<<3);
System.out.println(a>>>3);
```

megapie2021@gmail.com

복합대입 연산자

◆ 사칙연산과 대입을 함께 수행하는 연산자
◆ num+=3 => num=num+3

◆ num-=3 => num=num-3

◆ num*=3 => num=num*3

◆ num/=3 => num=num/3

◆ num%=3; => num=num%3

megapie2021@gmail.com

- 변수 +=식 , 변수 -=식, 변수 *=식, 변수 /=식, 변수 %=식
- 변수 &=식, 변수 |=식, 변수 ^=식
- 변수 <<=식 변수 >>=식, 변수 >>>=식

note!!

```
int a=20;
int b=10;
int c=5;

c*=a+b;

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

megapie2021@gmail.com

조건 연산자

◆ 조건식? 식1 or 값 1: 식2 or 값2

◆ 조건이 참(true)이면 식1 이나 값1을 선택 그렇지 않으면 식2 or 값2 선택

```
int jumsu=60;
String result=(jumsu>=60)? "60보다 크거나 같다 ":"60보다 작다";
System.out.println(result);
```

```
int jumsu=110; //10
int result=(jumsu>=100)? jumsu+100: jumsu+0;
System.out.println(result);
```

megapie2021@gmail.com

전체 연산자

구분	연산자	기능 설명
사칙 연산자	+ - * / %	사칙연산 및 나눗셈의 나머지 계산
부호 연산자	+ -	음수와 양수의 부호
문자열 연결 연산자	+	두 문자열을 연결
단순 대입 연산자	=	우변의 값을 좌변의 변수에 대입
증가/감소 연산자	++ --	변수 값을 1만큼 증가/감소
수치 비교 연산자	< > <= >=	수치의 크기 비교
동등 연산자	== !=	데이터의 동일 비교
논리 연산자	& | ^ !	논리적 AND, OR, XOR, NOT 연산
조건 AND/OR 연산자	&& ||	최적화된 논리적 AND, OR 연산
조건 연산자	?:	조건에 따라 두 값 중 하나를 택일
비트 연산자	& | ^ ~	비트 단위의 AND, OR, XOR, NOT 연산
쉬프트 연산자	<< >> >>>	비트를 좌측/우측으로 밀어서 이동
복합 대입 연산자	+= -= *= /= %= &= |= ^= <<= >>= >>>=	+ - * / % & | ^ << >> >>> 연산자와 = 의 기능을 함께 수행
캐스트 연산자	(타입 이름)	타입의 강제 변환

megapie2021@gmail.com

연산자 우선 순위

우선순위	연산자	처리 순서
높음 ↑	++ -- +(부호 연산자) -(부호 연산자) ~ ! 캐스트 연산자	←--
	* / %	--→
	+(덧셈 연산자, 문자열 연결 연산자) -(뺄셈 연산자)	--→
	<< >> >>>	--→
	< <= > >=	--→
	== !=	--→
	&	--→
	^	--→
	|	--→
	&&	--→
	||	--→
↓	? :	←--
낮음	= += -= *= /= %= &= |= ^= <= >>= >>>=	←--

megapie2021@gmail.com

입력테스트 - Scanner

```
import java.util.Scanner;

public class ScannerTest {
public static void main(String[] args) {
      Scanner sc=new Scanner(System.in);

       System.out.println("input name");
       String name=sc.nextLine();
      System.out.println("input age");
       int age=Integer.parseInt(sc.nextLine());
       System.out.println("input addr");
       String addr=sc.nextLine();

       System.out.println("name :"+name);
       System.out.println("age:"+age);
       System.out.println("addr :"+addr);
}
}
```

megapie2021@gmail.com

조건문 - if

◆ 조건문(conditional statement)

▪조건에 따라 주어진 일을 하는 명령문

◆ if 조건문

```
if(조건식)   // 조건이 참이면  실행문1 수행
      실행문 1
```

```
if(조건식) {        // 조건이 참인 경우 여러 문장 수행시 블록으로 묶음
   실행문1
   실행문 2
     ....
   실행문 n
}
```

```
 int a=20;

if(a>=20)  {
   System.out.println("a가 20보다 크거나 같습니다");
   System.out.println("a값은 :"+a);
}
```

megapie2021@gmail.com

if... else

◆ if... else

```
if(조건식)  // 조건이 참(true)이면 실행문1을 거짓이면(그렇지 않으면) 실행문2
      실행문1
else
      실행문2
```

```
int kor=90;
if(kor>=60)  {
       System.out.println("합격");
    }
else {
      System.out.println("불합격");
}
```

megapie2021@gmail.com

if... else

```
int a=70, b=50;

   if(a>=100)
       if(b>=40)
              System.out.println("test1");
   else
        System.out.println("test2");
```

Note: 여기서의 else는
if(a>=100)의 else?
if(b>=40)의 else?

megapie2021@gmail.com

여러 조건의 if 문 : if ..else if ....else

◆ if ..else if else

```
if(조건1)
        실행문1
else if(조건2)
        실행문2
else if (조건문3)
        실행문3
else
        실행문4
```

```
int kor=87;

 if(kor>=90)
        System.out.println("수");
 else if(kor>=80)
        System.out.println("우");
 else if(kor>=70)
        System.out.println("미");
 else if(kor>=60)
        System.out.println("양");
 else
        System.out.println("가");
```

megapie2021@gmail.com

조건문(switch)

◆ switch 조건문

```
switch(식 또는 변수)  //식 or 변수의 타입이 int형이 되거나 String(jdk7), enum
  {
      case 값1 :
          실행문1;
          break;
      case 값2:
          실행문2;
          break;
       case 값3:
           실행문3;
           break;
      default :
           실행문4;
  }
```

megapie2021@gmail.com

switch 수행 예-1

```
int a=3;

 switch(a)
{
      case 1:
            System.out.println("1입니다.");
            break;
      case 2:
            System.out.println("2입니다");
            break;
      case 3:
             System.out.println("3입니다");
             break;
      default:
             System.out.println("나머지");
}
```

megapie2021@gmail.com

switch 수행 예-2

```
String value="a";

switch(value)
 {
    case "a":
          System.out.println("a입니다");
          break;
    case "b":
          System.out.println("b입니다");
          break;
    default:
         System.out.println("나머지");
}
```

megapie2021@gmail.com

switch 수행 예-3

```
 enum Count
  {
   one, two, three, four, five
  }

public class SwitchTest{
     public static void main(String[] args) {
              switch(Count.two) {
                 case one:
                          System.out.println("하나");
                          break;
                 case two:
                          System.out.println("둘");
                          break;
                 default:
                         System.out.println("나머지");
              }
     }
}
```

megapie2021@gmail.com

반복문 -for

반복문 :
조건에 따라 반복하여 수행 하는 명령문
for, while, do..while등으로 표현이 가능함

수행순서

①　　②⑤⑧⑪⑭　　④⑦⑩⑬

for(초기값 ;　조건　　;　증감값)

{

실행문　③ ⑥ ⑨ ⑫

} ⑮

실행 순서 :　초기값 -> 조건 True-> 실행문 수행-> 증감-> 조건 True->실행문 .... 조건 False-> 끝

megapie2021@gmail.com

반복문 – for

```
for(int i=1; i<=3; i++)
  {
        System.out.println(i);
   }

 System.out.println("done");
```

```
for(int i=4; i>=2; i--)
{
     System.out.println(i);
}

 System.out.println("done");
```

megapie2021@gmail.com

반복문-for

```
for(int i=1; i<=10; i+=2)
 {
      System.out.println(i);
  }

 System.out.println("done");
```

```
 for(int i=10; i>=3; i-=2)
 {
      System.out.println(i);
  }

 System.out.println("done");
```

megapie2021@gmail.com

반복문(while)

◆ while(조건문) //조건이 true일 동안 반복수행

```
{
      실행문
 }
```

```
int i=1;
while(i<=10)
{
    System.out.println(i);
    i++;
 }
```

```
int i=1;
while(i<=10)
 {
     i++;
     System.out.println(i);
 }
```

반복문(do-while)

◆ do-while

```
do{
      실행문
  }while(조건식);
```

```
int i=1;
do {
      System.out.println(i);
      i++;

   } while(i<=10);
```

◆ while문과 do..while문의 차이

do.. while문 경우 조건을 실행문 후에 물어보기 때문에
조건이 false일 경우에도 한번은 실행문을 수행한다.

megapie2021@gmail.com

break 문

- for, while, do..while 문 등의 반복문을 빠져나옴
- switch문 안에 사용하여 switch문을 빠져 나옴

```
for(int i=0; i<10; i++)
{
        if(i>=5)
            break;
        System.out.println(i);
}
```

megapie2021@gmail.com

continue문

◆ 반복문시 다음 반복으로 진행

```
for(int i=1; i<=10; i++)
{
      if(i%3==0)
            continue;
      System.out.println(i);
  }
```

megapie2021@gmail.com

continue시 주의 사항

```
int i = 1;
while (i <= 10) {
      if ( i%3==0)
            continue;
      System.out.println(i);
      i++;
 }
```

```
int i=0;
while(i<10)  {
    i++;
    if(i%3==0)
        continue;
    System.out.println(i);

}
```

megapie2021@gmail.com

다중 반복문

for문이나 while문안에 for문이나 while을 작성할 수 있다.

```
for(int i=1; i<=3; i++)
{
    for(int j=1; j<=2; j++)
       {
          System.out.print(j +"  ");
          }
  System.out.println();
}
```

megapie2021@gmail.com

1차원 배열

◆ 배열선언

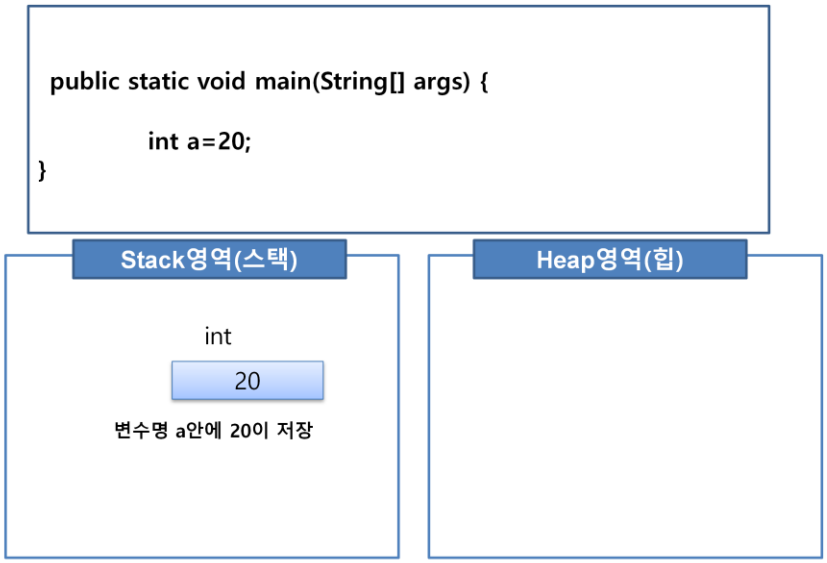
```
타입[]  변수명= new 타입[크기] ;

타입 []   변수명;
          변수명=new 타입[배열크기];
```

- int [] arr=new int[3];
- int [] arr2;
  arr2=new int[3];
- int[] arr2={10,20,30}; //초기값 부여

megapie2021@gmail.com

메모리 구조

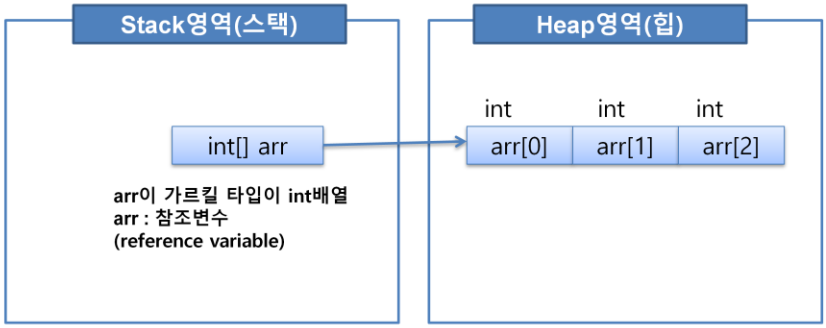


megapie2021@gmail.com

1차원 배열-메모리 구조

```
public static void main(String[] args) {

        int[] arr=new int[3];
        arr[0]=10;
        arr[1]=20;
        arr[2]=30;
}
```



megapie2021@gmail.com

2차원 배열

◆ 배열의 선언

```
타입[][]  변수명= new 타입[행의 수][열의 수] ;

타입 [][]   변수명;
    변수명=new 타입[행의 수][열의 수];
```

- int[][] arr=new int[3][4];
- float[][] arr2;
  arr2=new float[2][3];

- int[][] arr=new int[3];
  arr[0]=new int[2];
  arr[1]=new int[4];
  arr[2]=new int[3];

megapie2021@gmail.com

배열 예제

```
int[][] arr=new int[3][4];

System.out.println(arr.length);
System.out.println(arr[0].length);
System.out.println(arr[1].length);
System.out.println(arr[2].length);
System.out.println(arr[0][0]);
```

```
int[][] arr= {{10,10,10},{40,50,60},{30,30,30}};

for(int i=0; i<arr.length; i++)
{
for(int j=0; j<arr[i].length; j++)
{
System.out.print(arr[i][j]+"₩t");
}
System.out.println();
}
```

megapie2021@gmail.com

향상된 for문

- for(배열의 원소와 같은 타입 변수명 : 배열명)
  {
  실행문;
  }

```
int[] arr={10,20,30,40,50};

for(int index: arr)
    System.out.println(index);
```

megapie2021@gmail.com

열거형 타입

◆ 몇 개의 값으로만 한정적으로 이루어져 가지고 있는 타입

```
enum Week
{
   Sun, mon, Tue, Wed, Tur, Fri, Sat
}

public class WeekTest {
     public static void main(String[] args) {
          Week monday=Week.mon;
          System.out.println(monday);
          System.out.println(Week.mon);
    }
}
```

megapie2021@gmail.com

Q & A

megapie2021@gmail.com