



자바 프로그램의 기본 구조



학습 목표

- * 자바 애플리케이션을 작성하여 실행하기까지의 과정을 학습합니다.
- * 자바 애플리케이션의 기본적인 구조에 대하여 학습합니다.
- * 자바의 주석문, 예약어, 식별자에 대하여 학습합니다.
- * 자바의 기본형 변수의 종류와 특징에 대하여 학습합니다.
- * 자바에서 사용 가능한 연산자의 종류와 형태에 대하여 학습합니다.



자바 프로그램의 기본 구조

- * 자바 애플리케이션 작성, 컴파일, 실행 과정
- * 자바 애플리케이션 기본 구조 분석
- * 자바 주석문
- * 자바 식별자와 키워드
- * 자바의 기본형 변수
- * 자바의 연산자



자바 애플리케이션 작성, 컴파일, 실행 과정

- * 자바 애플리케이션 작성
- * 자바 애플리케이션 컴파일과 실행



자바 어플리케이션 작성

- * 프로그램 작성, 컴파일 후에 독립적 실행 가능한 자바 프로그램
- * main 메소드 정의 필요

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

- * main 메소드에 정의된 작업 수행



자바 어플리케이션 작성

* 자바 프로그램의 구조

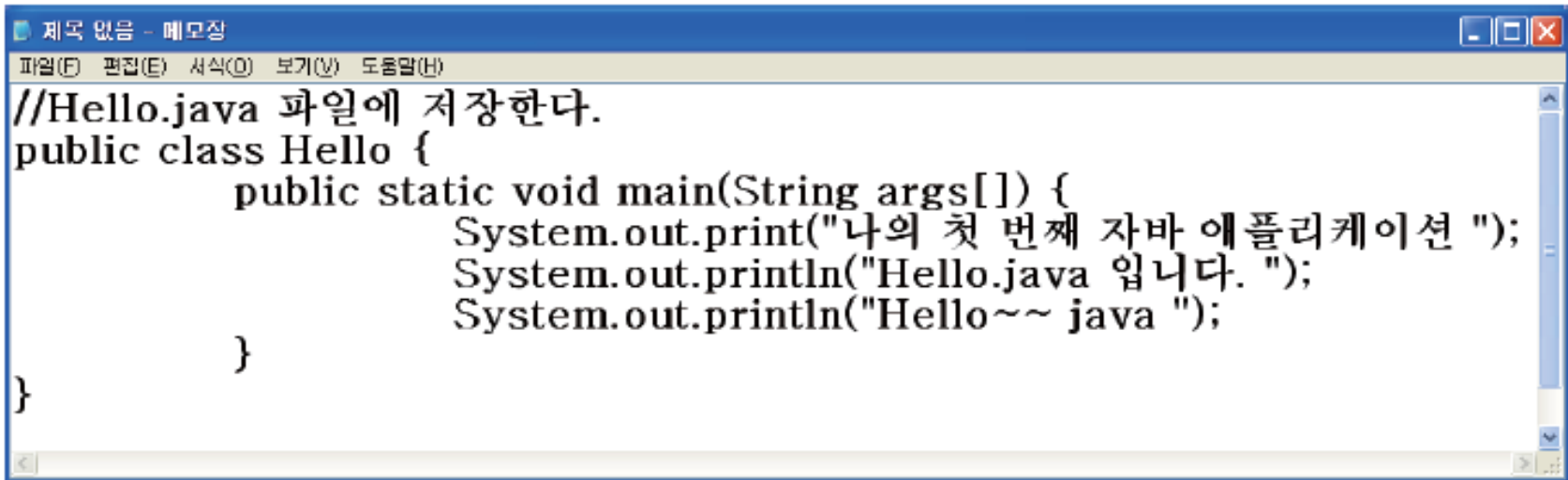
```
public class 클래스이름 {  
    // 변수 정의  
    // 메소드 정의  
}
```

```
class 클래스이름 {  
    // 변수 정의  
    // 메소드 정의  
}
```

* 클래스이름.java 로 저장

자바 어플리케이션 작성

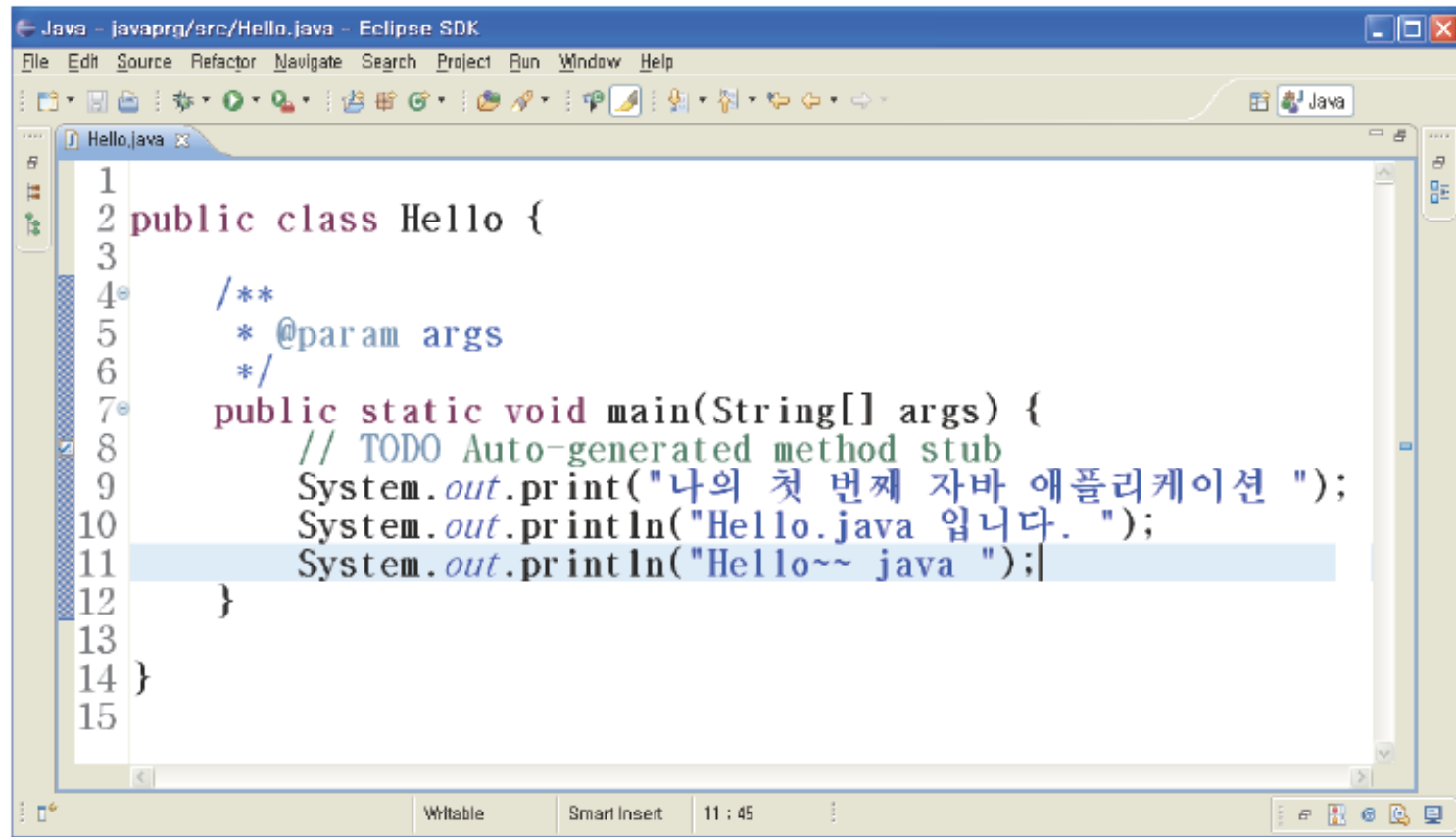
- * 자바 프로그램 작성은 메모장 또는 이클립스 등의 IDE 사용
- * 메모장



```
//Hello.java 파일에 저장한다.  
public class Hello {  
    public static void main(String args[]) {  
        System.out.print("나의 첫 번째 자바 어플리케이션 ");  
        System.out.println("Hello.java 입니다. ");  
        System.out.println("Hello~~ java ");  
    }  
}
```

자바 어플리케이션 작성

* 이클립스



```
Java - javaprg/src/Hello.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Hello.java x
1
2 public class Hello {
3
4     /**
5     * @param args
6     */
7     public static void main(String[] args) {
8         // TODO Auto-generated method stub
9         System.out.print("나의 첫 번째 자바 어플리케이션 ");
10        System.out.println("Hello.java 입니다. ");
11        System.out.println("Hello~~ java ");
12    }
13
14 }
15
Writeable Smart Insert 11 : 45
```




자바 어플리케이션 컴파일과 실행

- * 작성한 자바 소스는 컴파일 거쳐 실행

- * 컴파일 명령

- * javac 클래스명.java

```
javac Hello.java
```

- * 컴파일 후 클래스명.class 파일 자동 생성

- * 인터프리트 명령

- * java 클래스명

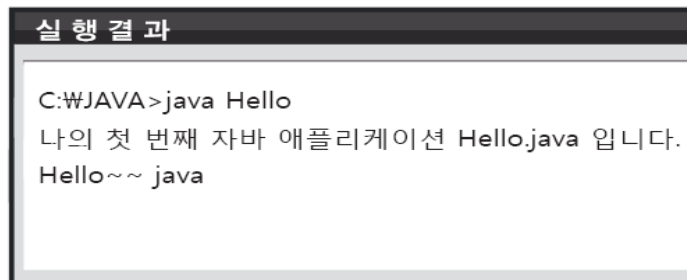
```
java Hello
```

- * main 메소드 실행

자바 어플리케이션 컴파일과 실행

* 프로그램 3-1 실습

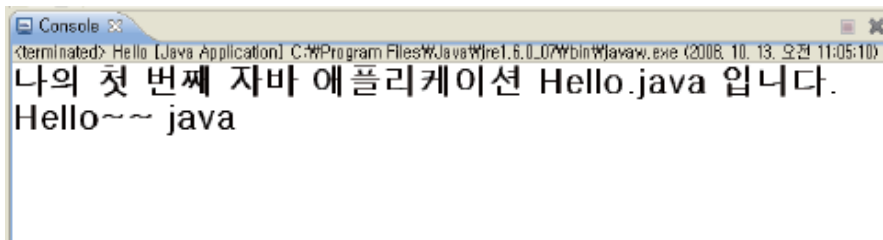
- * Hello.java 작성, 컴파일, 인터프리트
- * 메모장 이용



```
실행 결과

C:\WJAVA>java Hello
나의 첫 번째 자바 애플리케이션 Hello.java 입니다.
Hello~~ java
```

* 이클립스 이용



```
Console 52
terminated> Hello [Java Application] C:\Program Files\Java\jre1.6.0_07\bin\javaw.exe (2008. 10. 13. 오전 11:05:10)
나의 첫 번째 자바 애플리케이션 Hello.java 입니다.
Hello~~ java
```



자바 애플리케이션 기본 구조 분석

- * 문장, 메소드, 클래스 정의
- * 자바 표준 출력 문장



문장, 메소드, 클래스 정의

- * 하나 또는 여러 개의 클래스로 구성
- * 하나의 클래스 내에 변수와 메소드 정의하여 하나의 영역({})으로 표시
- * 하나의 메소드는 여러 개의 문장으로 구성되어 하나의 영역({})으로 표시
- * 하나의 문장은 세미콜론(;)으로 표시



문장, 메소드, 클래스 정의

* Hello 클래스 구조

```
public class Hello { ○———— 클래스 영역의 시작

    public static void main(String args[ ]) { ○———— 메소드 영역의 시작
        System.out.print("나의 첫 번째 자바 애플리케이션 ");

        System.out.println("Hello.java 입니다. "); ○———— 문장의 끝

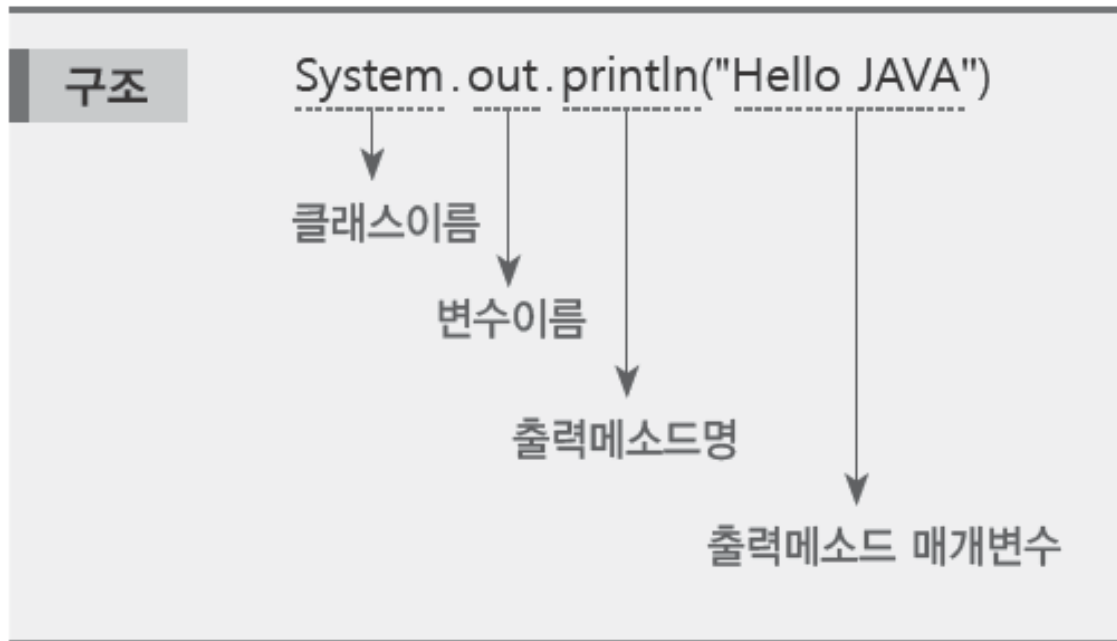
        System.out.println("Hello~~ java ");

    } ○———— 메소드 영역의 끝

} ○———— 클래스 영역의 끝
```

자바 표준 출력 문장

- * 화면에 프로그램 관련 내용 출력
- * `System.out.print()`
- * `System.out.println()`





자바 주석문

- * C++ 스타일 주석문
- * JAVA DOC 주석문



C++ 스타일 주석문

- * 적절한 주석은 프로그램 가독성을 높임
- * // : 단일 라인 주석 처리
- * /* */ : 여러 라인 주석 처리

```
//두 번째 줄에 내용을 출력하는 메소드 선언
static String getComment() {
    return "주석 테스트 프로그램입니다.";
}
```

```
/* 다음 클래스는 다음과 같은 두 줄의 내용을 출력하는
   자바 프로그램입니다.
   "출력될 내용은 다음과 같습니다."
   "주석 테스트 프로그램입니다."
*/
public class CommentTest {
```




JAVA DOC 주석문

- * 자바 소스 파일과 별도의 html 파일로 주석 문서화
- * `/** */` 내부에 주석 표시
- * 단일의 또는 멀티 라인의 주석 처리
- * javadoc 명령으로 html 파일 생성
- * html 파일 내에는 `/** */` 포함 내용 및 클래스, 변수, 메소드, 생성자 정보 포함



JAVA DOC 주석문

* 주석 작성 javadoc 명령 실행

```
/**  
 * 파일명 : CommentTest.java  
 * 작성일 : 작성일 : 2009 / 3 / 21  
 * 작성자 : 조성희  
 */
```

```
javadoc CommentTest.java
```

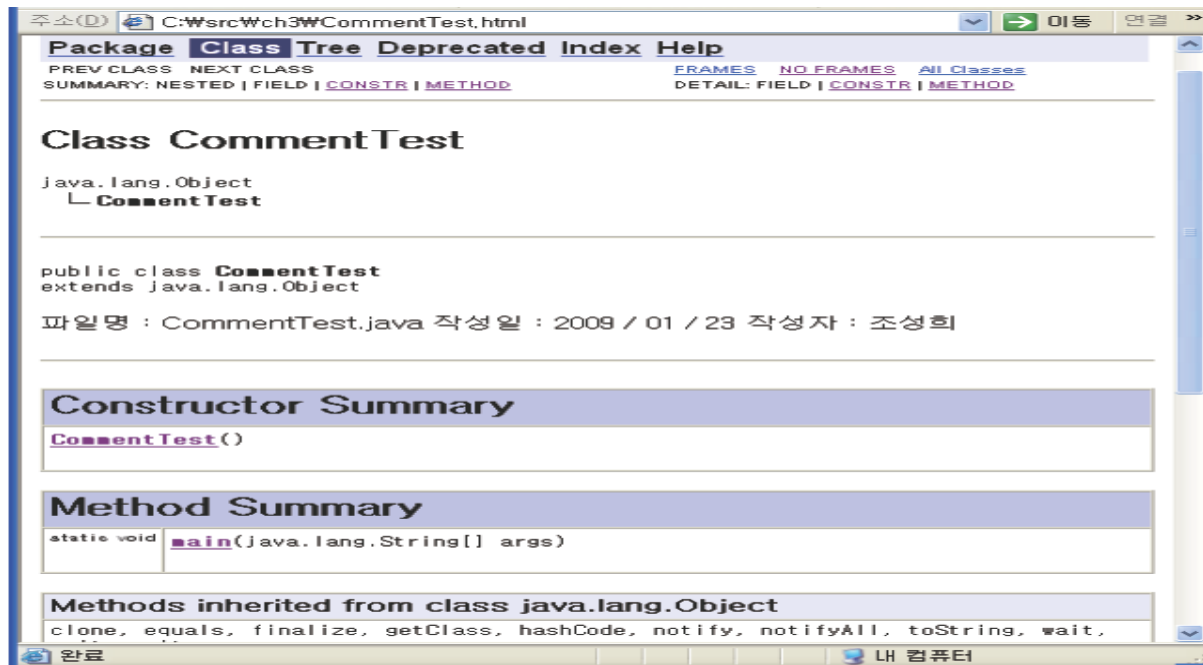
* 생성된 일부 html 파일들

실행 결과

```
C:\WJAVA>javadoc CommentTest.java  
Loading source file commentTest.java...  
Construction Javadoc information...  
Standard Doclet version 1.6.0_07  
Building tree for all the packages and classes...  
Generating CommentTest.html...  
Generating package-frame.html...
```

JAVA DOC 주석문

- * 프로그램 3-2 실습
 - * CommentTest.java 작성, 주석 문서화
 - * CommentTest.html 생성 확인





자바 식별자와 키워드

- * 자바 식별자 규칙과 관례
- * 자바 키워드 리스트




자바 식별자 규칙과 관례

- * 식별자 : 변수, 상수, 메소드, 클래스 이름들
- * 식별자 규칙
 - * 첫문자는 일반문자, _, \$만 가능
 - * 첫문자 아니면 일반문자, _, \$, 숫자 가능
 - * 키워드 사용 불가
 - * 대소문자 구분
 - * 공백 포함 불가
- * 규칙 어기면 컴파일 오류

자바 식별자 규칙과 관례

IDENTIFIER	사용 여부	오류 원인
index	ok	
3_grade	error	숫자로 시작
_Variable	ok	
Test100	ok	

\$testVar	ok	
class	error	KEYWORD
이름	ok	
thisName	ok	
this	ok	KEYWORD
blank name	error	IDENTIFIER 내에 공백을 포함
#abc	error	'_'나 '\$' 이외의 특수 문자 불가




자바 식별자 규칙과 관례

- * 식별자 관례를 따르면 가독성 높임
- * 클래스와 인터페이스 이름 관례
 - * 첫문자 대문자로 시작
 - * 두 단어 이상 결합되면 단어사이 대문자로 연결
 - * 명사 이름 사용
 - * 클래스 이름 예 : Button, Frame, Applet, Thread, MenuBar
 - * 인터페이스 이름 예 : Runnable, LayoutManager, AppletContext



자바 식별자 규칙과 관례

- * 변수와 메소드 이름 관례
 - * 첫문자 소문자로 시작
 - * 두 단어 이상 결합되면 단어사이 대문자로 연결
 - * 변수 이름은 명사, 메소드 이름은 동사 이름 사용
 - * 메소드 이름 예 : getName(), getPointSize(), setBackground()
 - * 변수 이름 예 : name, age, deptName, pointSize



자바 식별자 규칙과 관례

- * 상수 이름 관례
 - * 기본형 변수는 모두 대문자 사용
 - * 단어와 단어 사이는 _(underscore) 으로 연결
 - * 참조형 변수는 대소문자 모두 사용
 - * 기본형 상수 예 : MAX_VALUE, PI, CROSSHAIR_CURSOR
 - * 참조형 상수 예 : black, darkGray, UndefinedProperty

자바 키워드 리스트

* 모든 키워드 소문자로 표시

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



자바 키워드 리스트

- * 자바 키워드 사용시 주의점
- * `const`, `goto`는 현재 사용되지 않는 키워드로 식별자로 사용 불가능
- * 대문자 `TRUE`, `FALSE`, `NULL`은 자바 키워드 아님
- * `C`, `C++` 언어의 키워드 `sizeof` 자바 언어의 키워드 아님



자바의 기본형 변수

- * 자바의 변수들과 메모리 구조
- * 자바의 기본형 변수 종류와 특징



자바의 변수들과 메모리 구조

- * 자바에서 변수 사용하기
 - * 메모리상에 데이터를 저장하고 프로그램에서 사용
 - * 변수 사용 전에 선언 필요

```
데이터타입(data type) 변수이름;
```

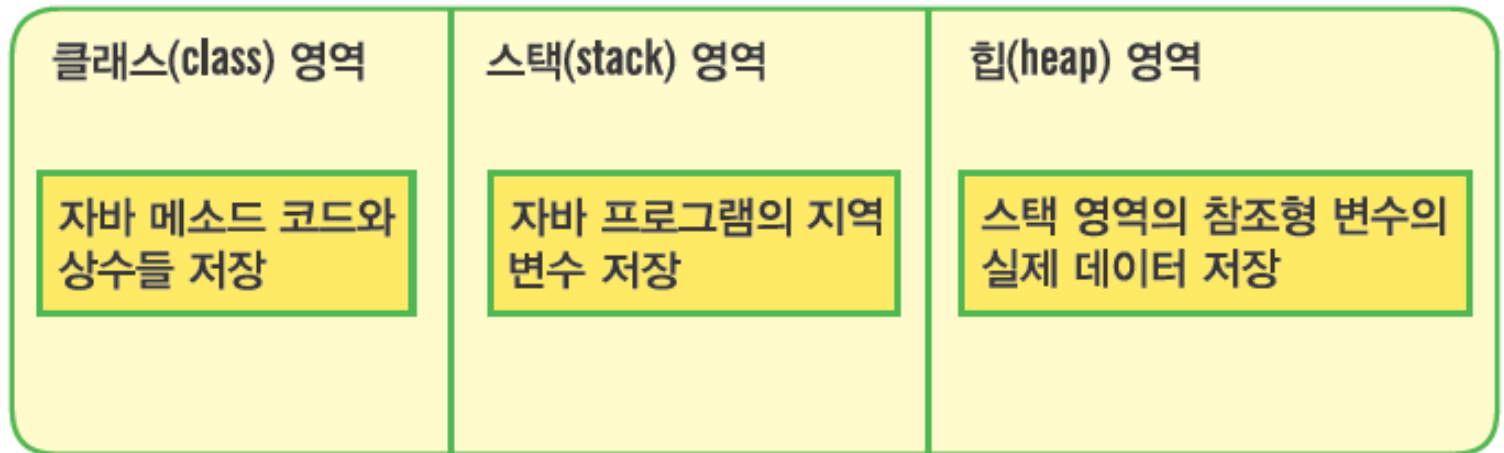
- * 논리형 데이터 flag 선언의 예

```
boolean flag;
```



자바의 변수들과 메모리 구조

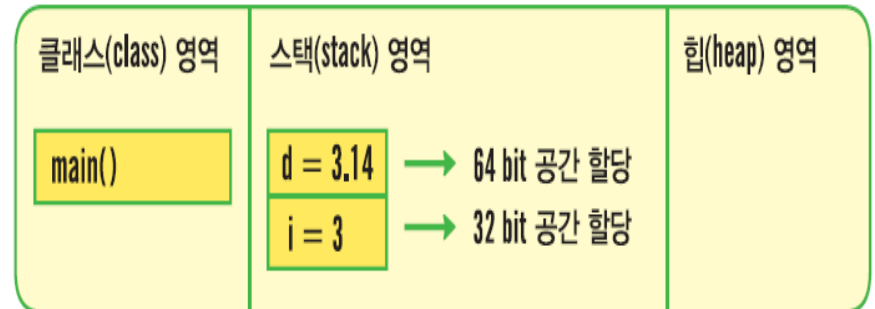
- * 변수와 메모리 구조
 - * 클래스, 스택, 힙 영역으로 구분



자바의 변수들과 메모리 구조

* 기본형 변수의 메모리 구조

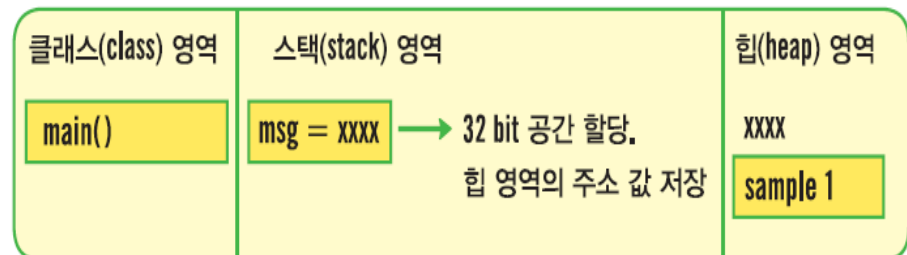
```
class TestVar {  
    public static void main(String arg[]) {  
        int    i; // 라인 1  
        double d; // 라인 2  
  
        i = 3; // 라인 3  
        d = 3.14; // 라인 4  
        System.out.println("i=" + i + "d=" + d);  
    }  
}
```



자바의 변수들과 메모리 구조

- * 참조형 변수의 메모리 구조
- * new 연산자 이용하여 힙 영역에 데이터 할당하고 할당된 주소 참조
- * 배열, 클래스, 인터페이스 타입

```
class TestVar {  
    public static void main(String[] args) {  
        String msg; // 라인 1  
        msg = new String("sample 1"); // 라인 2  
    }  
}
```



자바의 기본형 변수 종류와 특징

표현 형태	데이터타입	설 명
논리값	boolean	참이나 거짓을 나타내는 값
단일 문자	char	16비트의 유니코드 문자 데이터
정수	byte	부호가 있는 8비트의 정수
	short	부호가 있는 16비트의 정수
	int	부호가 있는 32비트의 정수
	long	부호가 있는 64비트의 정수
실수	float	부호가 있는 32비트의 부동소수점 실수
	double	부호가 있는 64비트의 부동소수점 실수

데이터 타입	크 기	표현 범위
boolean	1바이트	true 또는 false
char	2바이트	'\u0000' ~ '\uffff'
byte	1바이트	$-2^7 \sim 2^7-1$ (-128 ~ 127)
short	2바이트	$-2^{15} \sim 2^{15}-1$ (-32768 ~ 32767)
int	4바이트	$-2^{31} \sim 2^{31}-1$ (-2147483648 ~ 2147483647)
long	8바이트	$-2^{63} \sim 2^{63}-1$ (-9223372036854775808 ~ 9223372036854775807)
float	4바이트	1.4E-45 ~ 3.4028235E38
double	8바이트	4.9E-324 ~ 1.7976931348623157E308



자바의 기본형 변수 종류와 특징

- * byte 타입의 최대값 : `Byte.MAX_VALUE`
- * byte 타입의 최소값 : `Byte.MIN_VALUE`
- * short 타입의 최대값 : `Short.MAX_VALUE`
- * short 타입의 최소값 : `Short.MIN_VALUE`
- * int 타입의 최대값 : `Integer.MAX_VALUE`
- * int 타입의 최소값 : `Integer.MIN_VALUE`
- * long 타입의 최대값 : `Long.MAX_VALUE`
- * long 타입의 최소값 : `Long.MIN_VALUE`
- * float 타입의 최대값 : `Float.MAX_VALUE`
- * float 타입의 최소값 : `Float.MIN_VALUE`
- * double 타입의 최대값 : `Double.MAX_VALUE`
- * double 타입의 최소값 : `Double.MIN_VALUE`



자바의 기본형 변수 종류와 특징

* boolean

- * 논리값을 1바이트로 표현
- * true와 false 두 가지 값 표현

```
boolean    isFull; // boolean 타입 변수 선언  
isFull = true;  // boolean 타입 변수 값 초기화  
boolean    beOk = false; // boolean 타입 변수 선언과 초기화
```



자바의 기본형 변수 종류와 특징

* char

- * 단일 문자 2바이트 표현
- * 2^{16} 개의 문자 표현
- * 단일 따옴표(' ') 내부에 문자로 표현
- * 특수 문자는 'W'로 시작하여 표현

```
char    ch1 = 'Wn'; // new line
char    ch2 = 'Wr'; //carriage return
char    ch3 = 'Wt'; //tab
char    ch4 = 'Wu88ab'; // Unicode 표현
char    ch5;
ch5 = 'a';
```



자바의 기본형 변수 종류와 특징

- * 정수 타입 : byte, short, int, long
 - * 부호 있는 정수 표현
 - * 음수는 2의 보수로 표현
 - * 메모리 크기

정수	byte	부호가 있는 8비트의 정수
	short	부호가 있는 16비트의 정수
	int	부호가 있는 32비트의 정수
	long	부호가 있는 64비트의 정수



자바의 기본형 변수 종류와 특징

- * 정수 타입 : byte, short, int, long
 - * 정수 리터럴은 10진수, 8진수, 16진수 표현 가능

```
int exam1 = 1; //10진수
int exam2 = 076; //8진수. 0으로 시작
int exam3 = 0x1A;
//16진수. 0x나 0X로 시작. 16진수를 표현하는 A, B, C, D, E, F는 소문자로도 사용 가능
```

- * long 타입은 숫자 뒤에 l 또는 L 삽입

```
long exam4 = 1L; // 10진수 (long type)
long exam5 = 076L; // 8진수 (long type)
long exam6 = 0x886aL; // 16진수 (long type)
```



자바의 기본형 변수 종류와 특징

- * 실수 타입 : float, double

- * IEEE 754 규격 준수

- * 메모리 크기

실수	float	부호가 있는 32비트의 부동소수점 실수
	double	부호가 있는 64비트의 부동소수점 실수

- * 플랫폼 독립적



자바의 기본형 변수 종류와 특징

- * 실수 타입 : float, double
 - * 소수점 또는 지수 표현식 사용
 - * double은 숫자 리터럴만 있거나 또는 숫자 뒤에 D나 d 삽입
 - * float는 숫자 뒤에 F나 f 삽입

```
double exam7 = 0.1 ;  
double exam8 = 1.34e10 ;  
float exam9 = 0.1F ;  
float exam10 = 1.34E5F ;  
double exam11 = 1.34e10D ;
```




자바의 기본형 변수 종류와 특징

- * 프로그램 3-3 실습
 - * PrimitiveDataTest.java 작성
 - * 각 타입별 데이터 출력

```
C:\WJAVA>java PrimitiveDataTest  
boolean b = true  
char c = 9  
int i = 20  
double d = 5.24  
float f = 3.14  
long l = 10
```



자바의 기본형 변수 종류와 특징

- * 프로그램 3-4 실습
 - * MaxMinTest.java 작성
 - * 정수와 실수 타입별 최대값, 최소값 출력

```
C:\WJAVA>java MaxMinTest  
byte의 최소값 = -128  
byte의 최대값 = 127  
short의 최소값 = -32768  
short의 최대값 = 32768  
int의 최소값 = -2147483648  
int의 최대값 = 2147483648  
long의 최소값 = -9223372036854775808  
long의 최대값 = 9223372036854775808  
float의 최소값 = 1.4E-45  
float의 최대값 = 3.4028235E38  
double의 최소값 = 4.9E-234  
double의 최대값 = 1.7976931348623157E308
```



자바의 연산자

* 자바 연산자의 종류



자바의 연산자

- * 자바 연산자의 종류
 - * 형변환 연산자
 - * 산술 연산자
 - * 비교 연산자
 - * 논리 연산자
 - * 비트 연산자
 - * 대입 연산자
 - * 조건 삼항 연산자



자바 연산자의 종류

* 형변환 연산자

- * 데이터 타입을 명확하게 변경
- * 연산자 사용 형태

() 안에 변환하고 싶은 데이터 타입을 넣어서
변환하고자 하는 변수나 데이터 앞에 기술합니다.

* 형변환 규칙

- * boolean 타입 제외
- * 자동 형변환(묵시적 형변환, 확대 형변환)
- * 명시적 형변환(축소 형변환)

자바 연산자의 종류

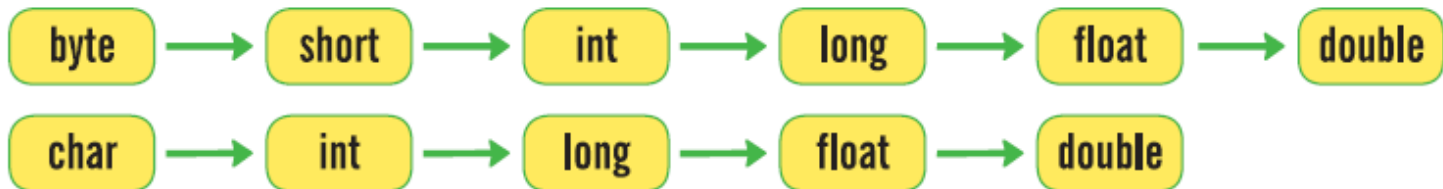
* 형변환 연산자

* 자동 형변환(묵시적 형변환, 확대 형변환)

* 형변환 연산자 생략 가능

```
byte b = 10;  
int i=b;
```

* 표현 범위가 더 큰 범위를 표시할 수 있는 데이터 타입으로 변환



자바 연산자의 종류

* 형변환 연산자

* 명시적 형변환(축소 형변환)

- * 작은 범위를 표시할 수 있는 데이터 타입으로 변환
- * 형변환 연산자 반드시 명시
- * 형변환시 메모리 구조

```
int i=10;  
byte b = (byte) i;
```

```
int i=128;  
i
```

0	00000000	00000000	00000000	10000000
---	----------	----------	----------	----------

부호(양수는 0) (10진수 128 을 이진수로 변환한 2^7 을 컴퓨터 내에 비트 단위로 표시)

```
byte b = (byte)i;  
b
```

10000000



자바 연산자의 종류

- * 프로그램 3-5 실습
 - * CastingTest.java 작성
 - * 자동 형변화와 명시적 형변환

```
C:\WJAVA>java CastingTest  
byte 97을 int로 자동 형변환 = 97  
int 97을 float로 자동 형변환 = 97.0  
float 97.0F를 double로 자동 형변환 = 97.0  
int 97을 char로 명시적 형변환 = a  
double 97.0을 int로 명시적 형변환 = 97
```


자바 연산자의 종류

* 산술 연산자

++	++값	값 1 증가 후에 값 취함	단항
	값++	현재 값을 취한 후 값 1 증가	
--	--값	값 2 감소 후에 값 취함	
	값--	현재 값을 취한 후 값 1 감소	
+	값1 + 값2	덧셈	이항
-	값1 - 값2	뺄셈	
*	값1 * 값2	곱셈	
/	값1 / 값2	나눗셈	
%	값1 % 값2	정수의 몫을 취한 후 나머지 구하는 연산	

자바 연산자의 종류

- * 산술 연산자 : ++(증가), --(감소) 연산자

- * 현재의 정수 값을 1 증가 또는 감소

- * 단항 연산자

- * 변수의 왼쪽, 오른쪽 모두 가능

- *

```
int i = 2;  
int r = i++; // 라인 1.
```

변수 r에는 2, 변수 i에는 3 저장. 즉, 변수 i의 값을 변수 r에 대입시킨 후 변수 i의 값 1 증가

- *

```
int i = 2;  
int r = ++i; // 라인 1.
```

변수 r에는 3, 변수 i에도 3 저장. 즉, 변수 i의 값을 먼저 1 증가시켜 변환한 후 변수 r에 대입



자바 연산자의 종류

- * 산술 연산자 : %(나머지) 연산자

- * 정수의 몫을 구한 나머지

- * 이항 연산자

- * boolean 타입 적용 불가능

- *

```
int i = 9 % 2;
```

i 변수의 결과는 1



자바 연산자의 종류

- * 프로그램 3-6 실습
 - * IncrementOpTest.java 작성
 - * ++(증가), --(감소) 연산자 이용

```
C:\W\JAVA>java IncrementOpTest
```

```
before i = 5      after i = 5
```

```
before i = 6      after i = 7
```

```
before i = 7      after i = 6
```

```
before i = 6      after i = 6
```



자바 연산자의 종류

- * 프로그램 3-7 실습
 - * ArithmeticOpTest.java 작성
 - * *(곱셈), /(나눗셈), %(나머지) 연산자 이용

```
C:\WJAVA>java ArithmeticOpTest  
28 * 5 = 140  
28 / 5 = 5  
28 % 5 = 3
```

자바 연산자의 종류

* 비교 연산자

- * 값의 동일성이나 대소 비교, 객체 타입 비교
- * 결과는 true 또는 false 중 하나 리턴

>	값 1 > 값 2	값 1이 값 2보다 큰 경우 true
>=	값 1 >= 값 2	값 1이 값 2보다 크거나 같은 경우 true
<	값 1 < 값 2	값 1이 값 2보다 작은 경우 true
<=	값 1 <= 값 2	값 1이 값 2보다 작거나 같은 경우 true
==	값 1 == 값 2	값 1과 값 2가 같은 경우 true
!=	값 1 != 값 2	값 1과 값 2가 같지 않은 경우 true
instanceof	값 1 instanceof 값 2	값 1이 값 2 데이터형의 객체인 경우 true



자바 연산자의 종류

- * 비교 연산자 : `||`(OR), `&&`(AND) 연산자
 - * `&&`(AND) 연산자는 양쪽 값이 모두 true인 경우에만 true 리턴
 - * `||`(OR) 연산자는 최소 한쪽의 값만 true인 경우에 true 리턴
 - * `||`, `&&` 연산자 모두 연산을 완전히 수행하지 않아도 연산 결과 결정 가능



자바 연산자의 종류

* 비교 연산자 : ||(OR), &&(AND) 연산자



```
String    str1 = null;  
int    i = 0;  
if (( i != 0 ) && (str1.length() > 5))
```

i != 0의 결과는 false가 결정되어 더 이상 str1.length() > 5의 연산 수행할 필요없이 && 연산 결과 false 결정



먼저 연산한 항의 결과가 true일 경우 다른 항도 연산



먼저 계산한 항의 결과가 false라면 나머지 항은 연산할 필요없이 결과 false 리턴



자바 연산자의 종류

* 비교 연산자 : ||(OR), &&(AND) 연산자

*

```
String    str1 = null;
int    i = 0;
if (( i != 0 ) || (str1.length() > 5))
```

i != 0의 연산 결과가 true이면 다른 한쪽 항의 결과에 관계없이 true 결정

i != 0 은 false이면 str1.length() > 5 수행하므로 NullPointerException 발생



자바 연산자의 종류

- * 프로그램 3-8 실습
 - * ShortAndOpTest.java 작성
 - * &, && 연산자 이용하여 결과 출력

```
C:\WJAVA>java ShortAndOpTest
&&거짓
Exception in thread "main" java.lang.NullPointerException
    at ShortAndOpTest.main(ShortAndOpTest.java:9)
```



자바 연산자의 종류

- * 프로그램 3-9 실습
 - * ShortOrOpTest.java 작성
 - * |, || 연산자 이용하여 결과 출력

```
C:\WJAVA>java ShortOrOpTest
```

```
|| 참
```

```
Exception in thread "main" java.lang.NullPointerException  
    at ShortOrOpTest.main(ShortOrOpTest.java:9)
```



자바 연산자의 종류

* 비트 연산자

&	값 1 & 값 2	값 1과 값 2의 비트 단위 논리곱(and) 연산
	값 1 값 2	값 1과 값 2의 비트 단위 논리합(or) 연산
^	값 1 ^ 값 2	값 1과 값 2의 비트 단위 배타 논리합(exclusive or) 연산
~	~값	값의 비트 단위 보수(not) 연산
>>	값 1 >> 값 2	값 1을 비트 단위로 값 2의 비트 수만큼 오른쪽으로 쉬프트
>>>	값 1 >>> 값 2	값 1을 비트 단위로 값 2의 비트 수만큼 오른쪽으로 쉬프트, 왼쪽에는 0이 채워진다.
<<	값 1 << 값 2	값 1을 비트 단위로 값 2의 비트 수만큼 왼쪽으로 쉬프트



자바 연산자의 종류

- * 대입 연산자
 - * = 연산자 사용
 - * 연산자 오른쪽의 값을 왼쪽에 대입
 - * == (동등 비교) 연산자와 구분 필요
 - * 산술, 논리, 비트 연산자 등의 다른 연산자와 같이 사용하여 축약 연산자로 이용 가능

자바 연산자의 종류

* 대입 연산자

=	값 1 = 값 2	값 2를 값 1에 대입
+=	값 1 += 값 2	값 1 = 값 1 + 값 2
-=	값 1 -= 값 2	값 1 = 값 1 - 값 2
*=	값 1 *= 값 2	값 1 = 값 1 * 값 2
/=	값 1 /= 값 2	값 1 = 값 1 / 값 2
%=	값 1 %= 값 2	값 1 = 값 1 % 값 2
&=	값 1 &= 값 2	값 1 = 값 1 & 값 2
=	값 1 = 값 2	값 1 = 값 1 값 2
^=	값 1 ^= 값 2	값 1 = 값 1 ^ 값 2
<<=	값 1 <<= 값 2	값 1 = 값 1 << 값 2
<<<=	값 1 <<<= 값 2	값 1 = 값 1 <<< 값 2
>>=	값 1 >>= 값 2	값 1 = 값 1 >> 값 2

자바 연산자의 종류

* 조건 삼항 연산자



형식 조건식?값1:값2

조건식 : **boolean** 결과 리턴.

값1 : 조건식의 결과가 **true**인 경우의 연산 결과

값2 : 조건식의 결과가 **false**인 경우의 연산 결과



연산자 사용 예

```
int jumsu = 50;  
String s = jumsu >= 0 && jumsu <= 100 ? "적합" : "비적합" ;  
// jumsu 는 50 이므로 조건식은 true, s 변수에 "적합" 대입
```

```
int value = -1;  
int flag = value >= 0 ? 0 : 1 ;  
// value 변수는 -1 이므로 조건식은 false. flag 변수에 1 대입.
```



자바 연산자의 종류

- * 프로그램 3-10 실습
 - * ConditionAndOpTest.java 작성
 - * ?:(조건 삼항) 연산자 이용

```
C:\W\JAVA>java ConditionOpTest  
50 : 적합한 데이터입니다.
```




자바 연산자의 종류

- * 연산자의 우선 순위
 - * $()$, $[]$ 표현식 최우선 순위
 - * 단항, 이항, 삼항 연산자 순서
 - * 대입 연산자 최하위 순위

자바 연산자의 종류

* 연산자의 우선 순위

1	() [] .
2	++ -- ~ ! (데이터 타입)
3	* / %
4	+ -
5	>> >>> <<
6	> >= < <=
7	== !=
8	&
9	^
10	
11	&&
12	
13	?:
14	= 및 각종 축약된 대입 연산자



정리

- * 자바 애플리케이션 프로그램은 소스 작성, 저장, 컴파일, 실행의 순서로 결과를 확인합니다.
- * 자바 언어에서는 `System.out.println();` 이나 `System.out.print();` 문장으로 결과를 출력합니다.
- * 자바 언어에서는 `//`, `/* */`, `/** */` 형태의 주석문을 제공합니다.
- * 자바의 식별자(IDENTIFIER)는 대소문자를 구별하여 예약어가 아닌 일반 문자들과 숫자, `_`, `$` 으로 구성되며 이중에서 숫자로 시작할 수 없습니다.
- * 자바의 키워드는 50여 개 정도가 존재하며 모두 소문자로만 구성되어 있습니다.
- * 자바 언어의 변수는 메모리에 저장되는 값의 성격에 따라서 실제 값을 저장하는 기본형 변수와 주소 값을 저장하는 참조형 변수로 구분합니다.



정리

* 자바의 기본형 변수들은 다음과 같습니다.

논리값	boolean	참이나 거짓을 나타내는 값	1byte
단일 문자	char	16비트의 유니코드 문자 데이터	2byte
정수	byte	부호가 있는 8비트의 정수	1byte
	short	부호가 있는 16비트의 정수	2byte
	int	부호가 있는 32비트의 정수	4byte
	long	부호가 있는 64비트의 정수	8byte
실수	float	부호가 있는 32비트의 부동소수점 실수	4byte
	double	부호가 있는 64비트의 부동소수점 실수	8byte



정리

- * boolean 타입을 제외한 기본형 변수 간에는 표현 범위가 커지는 쪽으로의 변환은 자동으로, 그 반대의 경우에는 명시적 형변환을 할 수 있습니다.
- * 산술 연산자로는 ++, --, +, -, *, /, % 연산자를 제공합니다.
- * 비교 연산자는 대소 비교나 객체의 타입 비교 등에 사용되고, true나 false boolean 결과를 리턴합니다.
- * 논리 연산자로는 &&(AND), ||(OR) 연산자를 제공하여 단축 연산이 가능합니다.
- * 대입 연산자는 '=' 의 형태로 표현하며 연산자 간의 우선순위가 가장 낮아서 가장 나중에 연산됩니다.
- * '?:'(조건 삼항) 연산자를 제공하여 주어진 조건에 따라 서로 다른 결과를 가져옵니다.