

# 변수(variable)





변수



# 변수

## 변수

→ 메모리 공간(RAM)에 한 개의 값을 기록하기 위한 장소(공간)

## 변수 사용 이유

메모장에 “안녕하세요” 메시지를 100번 작성해야 한다면?

1. “안녕하세요”를 100번 직접 작성한다.
2. “안녕하세요”를 한번 작성 후 copy(Ctrl+c) & paste(Ctrl+v)를 이용한다.  
→ 최초에 작성된 “안녕하세요”라는 메시지를 컴퓨터의 메모리공간에 저장한 후(copy) 필요한 경우에 꺼내서 사용(paste)하는 방식



# 변수

## 변수의 선언

→ 메모리 공간(RAM)에 데이터를 저장할 수 있는 공간을 할당하는 것

**자료형** **변수명** ;

마침

저장할 데이터의 타입(정수, 실수, 문자, 문자열, 등등....)

해당 값을 기억하기 위한 변수의 이름

**int number;**

number라는 이름의 정수형 변수 선언

# 자료형

## 변수의 자료형

### 1. 기본형(Primitive Type)

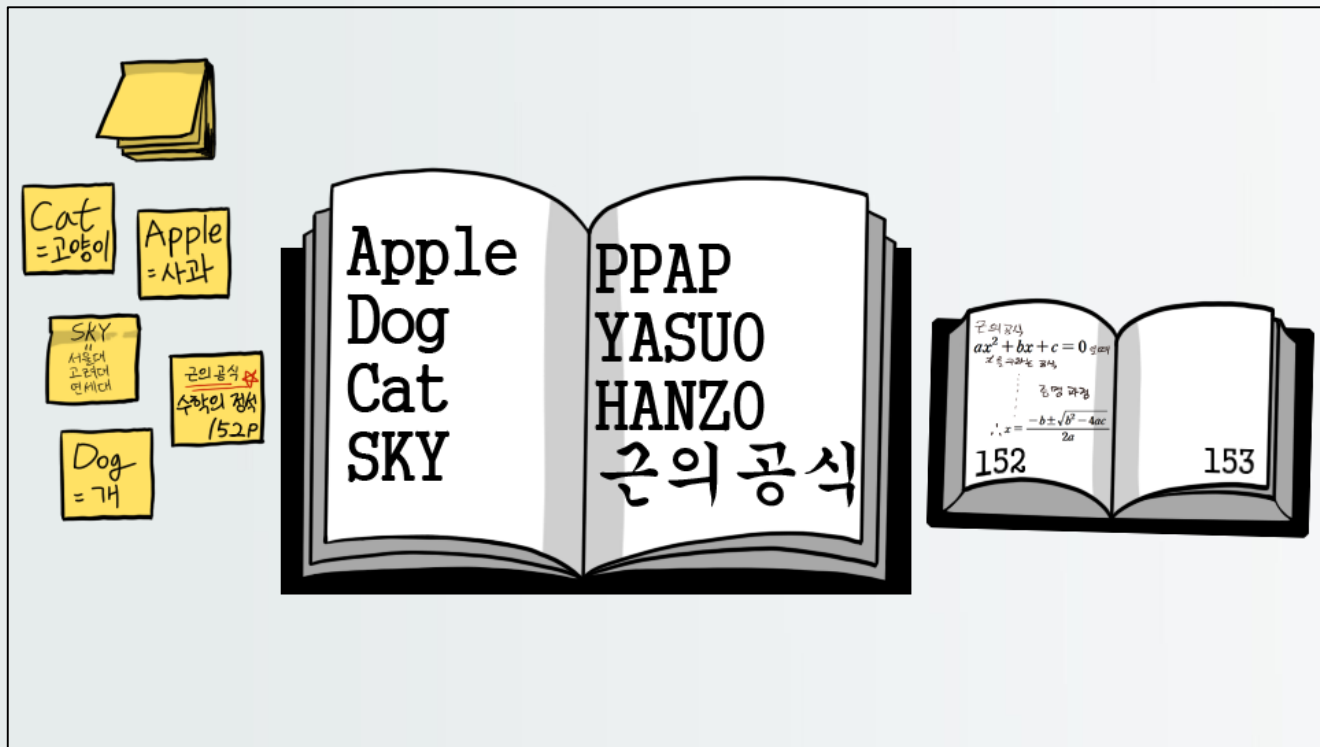
- **실제 데이터(값)를 저장**
- 논리형, 문자형, 정수형, 실수형으로 나뉘지고, 8개의 자료형이 있음
- 각 자료형 별 데이터의 저장 크기가 다름

### 2. 참조형(Reference Type)

- 데이터가 저장되어 있는 **주소를 저장**(객체의 주소)
- 기본형을 제외한 나머지(String 등), 사용자정의 자료형
- 주소값만 저장하기 때문에 4byte로 저장크기가 일정함



# 자료형



# 기본 자료형

분류	종류	크기	표현범위	내용
논리형	boolean	1byte	true / false	- true, false 중 하나의 값 - 조건식이나 논리적 계산에 사용
문자형	char	2byte	u0000 ~ uffff	- 한 개의 문자를 저장하는데 사용(Unicode)
정수형	byte	1byte	-128 ~ 127	- 이진데이터를 다루는데 사용
	short	2byte	-32768 ~ 32767	- C언어와 호환
	int	4byte	$-2^{31} \sim 2^{31}-1$	- 정수를 저장하는데 사용
	long	8byte	$-2^{63} \sim 2^{63}-1$	- 정수의 기본 자료형은 int이며, 범위 초과시 long을 사용
실수형	float	4byte	1.4E-45 ~ 3.4E38	- 실수를 저장하는데 사용
	double	8byte	4.9E-324 ~ 1.8E308	- 실수는 근사치를 구하는 방식으로 처리되어, 오차발생의 여지가 있어 기본 자료형은 오차 발생률이 적은 double을 사용

※ 정수 및 실수의 기본자료형의 의미는 일반적으로 정수 또는 실수를 사용했을 때 자동으로 처리되는 자료형을 의미

ex)

System.out.println(1); → 이때 사용 된 1은 정수 자료형 4개 중 int형

System.out.println(1.1); → 이때 사용 된 1.1은 실수 자료형 2개 중 double형



# 기본 자료형

## 정수의 양수/음수 표현방법

정수는 양수/음수가 존재하기 때문에 양수/음수를 표현하기 위해 제일 앞의 bit를 부호비트(msb)로 사용

byte -> 1byte = 8bit -> 1+7 -> 부호표현 1bit 값 표현 7bit



short -> 2byte = 16bit -> 1+15 -> 부호표현 1bit 값 표현 15bit



int -> 4byte = 32bit -> 1+31 -> 부호표현 1bit 값 표현 31bit



long -> 8byte = ??





# 기본 자료형

## 실수의 데이터 표현

IEEE754 표준방식에 따라서 표현

float 4byte = 32bit = 1+8+23



double 8byte = 64bit = 1+11+52



→ 실수표현방식 :  $\pm M \times 2^E$

2진수로 실수를 모두 표현하기에 한계가 있어 근사값을 구해서 표현하는 것(부동소수점)

※ 실수형의 경우 오차가 적어 정확성이 높은 double을 기본형으로 사용



# 참조 자료형

## String

- 하나의 문자열을 저장하는 데 사용
- 문자는 한 글자만을 의미하지만 문자열은 한 글자 이상의 문자를 의미(단어, 문장, 문단 등...)
- String은 문자열을 저장하기 위해 자바에서 만든 특수한 형태 → 사용자정의 → 참조형



# 변수

## 변수의 선언

**자료형** **변수명** ;

마침

저장할 데이터의 타입(정수, 실수, 문자, 문자열, 등등....)

해당 값을 기억하기 위한 변수의 이름

논리형 변수 선언 : `boolean bool;`

문자형 변수 선언 : `char ch;`

문자열 변수 선언 : `String str;`

정수형 변수 선언

- `byte bNum;`
- `short sNum;`
- `int num;`
- `long lNum;`

실수형 변수 선언

- `float fNum;`
- `double dNum;`



# 변수

## 변수 명명규칙

1. 대소문자가 구분되며 길이 제한이 없다.
2. 예약어를 사용하면 안 된다.
3. 숫자로 시작하면 안 된다.
4. 특수문자는 '\_'와 '\$'만을 허용한다.



반드시 지켜져야 하며,  
그렇지 않은 경우 에러발생

5. 카멜 표기법을 사용한다.
6. 한글입력이 가능하지만 사용을 지양한다.
7. 의미있는 변수명을 사용하는 것을 권고한다.



지키지 않아도 사용은 가능하지만,  
프로그래머 간의 약속임  
→ 결국 지켜야 함



# 변수

## 자바 주요 예약어

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



# 변수

## 변수의 초기화

- 변수를 사용하기 전에 처음으로 값을 저장하는 것
- 변수를 사용하기 위해서는 반드시 초기화 해야함

## 변수 초기화 방법

1. 변수 선언 이후 값을 대입  
`int age;`  
`age = 20;`
2. 변수 선언과 동시에 초기화  
`int age = 20;`

※ = 기호는 등호가 아니라 대입연산자

→ 대입연산자를 기준으로 오른쪽의 값을 왼쪽변수에 대입



# 변수

## Literal

→ 변수에 대입되는 값 자체를 의미

## Literal을 이용한 변수 초기화

논리형 변수 : `boolean bool = true;`

문자형 변수 : `char ch = 'A';`

문자열 변수 : `String str = "Hello";`

정수형 변수

- `byte bNum = 100;`
- `short sNum = 200;`
- `int num = 2000;`
- `long lNum = 100000000000L;`

실수형 변수

- `float fNum = 1.123f;`
- `double dNum = 3.14;`

※ 정수형 `long`과 실수형 `float`에는 뒤에 각각 해당 자료형을 표현하는 문자표기 (대소문자 상관없음)



# 변수

## 상수

→ 수학에서 변하지 않는 값, 프로그래밍에서는 한번만 저장할 수 있는 공간으로 다른 변수와의 차이점은 초기화 이후 다른 값을 대입할 수 없음

## 상수선언

**final double PI = 3.14**

1. **Final** : 해당 변수가 상수임을 선언하기 위한 시스템 예약어  
→ 자료형 앞에 **final**이란 키워드를 붙이면 해당 변수는 상수형 변수로 선언되며 바로 초기화해야함
2. **PI** : 변수명은 카멜표기법을 사용하는 것을 규약으로 하지만 상수형 변수는 모두 대문자표기가 관례임  
→ 변수명만 보더라도 다른 일반 변수와 구분하기 위해서 모두 대문자로 표기함



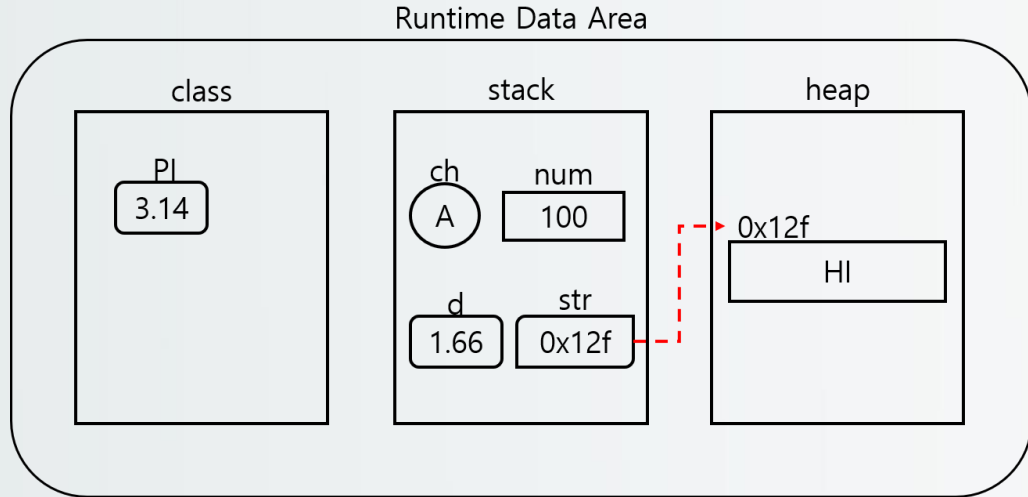


# 변수

## 변수의 메모리 저장

→ 변수는 메모리에 저장하는 공간이며 기본형, 참조형, 상수형에 따라 저장되는 메모리의 영역이 다름

변수 선언  
`char ch = 'A';`  
`int num = 100;`  
`double d = 1.66;`  
`String str = "HI";`  
`final double PI = 3.14;`





형변환



# 형변환

## 컴퓨터의 동작 원칙

1. 반드시 같은 데이터 타입 간의 연산을 한다.
2. 반드시 동일한 데이터 타입의 값을 동일한 데이터타입의 변수에 대입한다.
3. 결과값 또한 동일한 데이터 타입의 값을 갖는다.

※ 컴퓨터의 동작원칙이 지켜지지 않을 경우 프로그래밍언어가 자동으로, 혹은 사용자에 의해 강제로 **형변환**을 해야함



# 형변환

## 자동 형변환이 되는 경우

1. 자료형이 다른 값에 대입될 때(작은 자료형 → 큰 자료형)

byte b = 100;                      → byte형 변수 b에 100이라는 값 대입  
int i = b;                            → int형 변수 i에 변수b에 있는 값(100)대입

2. 자료형이 다른 값이 계산될 때

int i = 3;                            → int형 변수 i에 3이라는 값 대입  
double d = 9.9;                    → double형 변수 d에 9.9라는 값 대입  
System.out.println(i+d);  
→ i의 값인 3이 double형 3.0으로 자동 형 변환 된 후 연산하여 결과는 12.9가 출력



# 형변환

## 강제 형변환을 해야 하는 경우

→ 자동형변환을 사용 할 수 없거나 원하는 결과를 얻을 수 없는 경우 강제형변환을 해야 함

→ 강제 형변환 방법은 변환대상 데이터 값 앞에 (변환할 자료형)을 작성

```
int number = 1;
```

```
byte bNum = (byte)number;
```

```
int num = 2147483647;
```

```
int result = num + 1;
```

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

→ 정수형 변수 num에 2147483647이라는 값 대입

→ 정수형 변수 result에 num의 값에 +1한 값을 대입

→ 출력결과는?



```
int num = 2147483647;
```

```
long result = (long)num + 1;
```

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

→ int형 변수 num에 2147483647이라는 값 대입

→ long형 변수 result에 num의 값을 long으로 형변환 후 +1한 값을 대입

→ 2147483648 출력



# 형변환

## Data Overflow

→ 데이터가 허용된 범위 이상의 비트를 침범하는 것

1byte 기준으로 확인

byte 변수에서 01111111 → 127

128을 만들기 위해서는 127+1

2진수 연산으로 127+1을 하면

10000000

정수의 가장 앞 bit는 부호비트이므로

10000000 → -128이 됨

이러한 상황을 Data Overflow라 함



+  
1



# 형변환

## byte와 short의 연산

- 자바에서 사용하는 정수형 기본자료형은 int
- byte와 short를 연산하는 경우 모두 int로 변환해서 연산을 하고, 결과를 int로 반환
- 이 때, 결과값을 byte나 short로 처리하기 위해서는 강제로 형변환을 해주어야 함

byte num1 = 10, num2 = 20;      → byte형 변수 num1에 10대입, byte형 변수 num2에 20대입  
byte result = num1 + num2;      → 더하기 연산 시 int로 변환되어 연산 후 int값을 반환하여 에러  
에러 해결방안

1. 결과값을 int형 변수에 대입한다.  
→ int result = num1 + num2;
2. byte로 강제 형변환을 한다.  
→ byte result = (byte)(num1+num2);      →()를 통해서 연산을 먼저 한 후 결과값을 형변환



# 형변환

## 강제 형변환 시 주의사항

→ 데이터가 큰 자료형에서 작은 자료형으로 변경 시 데이터 손실이 발생 할 수 있으므로 주의

```
int iNum = 290;
```

```
byte bNum = (byte)iNum;
```







# 입·출력 메소드



# 출력 메소드

## 1. System.out.println()

- ()안의 변수, 문자, 숫자, 논리값을 출력해 주는 메소드로 해당 내용을 출력한 후 자동으로 줄바꿈
- ex) `System.out.println("hello");`, `System.out.println(1111);`, `System.out.println(변수명);`

## 2. System.out.print()

- `System.out.println()`과 거의 동일하지만 출력한 후 줄 바꿈을 하지 않음
- ex) `System.out.print("hello");`, `System.out.print(1111);`, `System.out.print(변수명);`



# 출력 메소드

## 1. System.out.printf()

- 정해져 있는 형식에 맞춰서 그 형식에 맞는 값(변수)을 줄 바꿈 하지 않고 출력
- 여러 자료형을 한번에 사용할 때 편리

ex) System.out.printf("%s", "hello");, System.out.printf("%d", 1111);

### 정수 표현

- %d : 정수형, %o : 8진수, %x : 16진수

### 실수 표현

- %f : 실수(소수점 아래 6자리), %e : 지수형태표현
- %g : %e, %f 중 더 짧은 표현을 사용
- %A : 16진수 실수

### 문자/문자열 표현

- %c : 문자
- %s : 문자열

### 논리형 표현

- %b : 논리형

### 너비 및 정렬 방법

- %5d : 5칸을 확보하고 오른쪽 정렬
- %-5d : 5칸을 확보하고 왼쪽 정렬
- %.2f : 소수점 아래 2자리까지만 표시



# 출력 메소드

## escape 문자

- 출력 메소드 내부에서 기능이 있는 문자(“,’,%,등등)을 사용하는 방법

특수문자	문자 리터럴	비고
Tab	\t	정해진 공간만큼 띄어쓰기
New line	\n	출력하고 다음라인으로 옮김
역슬래쉬	\\	기능이 있는 문자 사용 시 역슬래쉬를 넣고 특수문자를 넣으면 기능문자가 아닌 문자로 인지함
작은 따옴표	\'	
큰 따옴표	\"	
유니코드	\u	유니코드 표시할 때 사용



# 출력 메소드

## 출력 메소드 실습

### 문제 1

아래 주어진 값에 해당하는 변수를 만들고 주어진 값으로 초기화 한 후 그 값을 출력하는 코드 작성

- 정수형 100
- 정수형 9999억
- 실수형 486.520(float)
- 실수형 5697.890123
- 문자 A
- 문자열 Hello JAVA
- 논리 true

※ 모든 값은 변수에 저장하고 변수를 출력

### 문제 2

자신의 신상정보를 저장 할 변수를 만들고 정보를 변수에 대입, 출력하는 프로그램 작성

- 이름, 나이, 성별, 주소, 전화번호, 이메일
- 본인정보 출력이 완성되었으면, 옆 사람 신상 정보 또한 변수에 대입하여 출력하는 프로그램으로 수정



이름	나이	성별	지역	전화번호	이메일
홍두깨	22	남	서울	01012345678	myemail@email.com
고길동	50	남	경기도	01098765432	mrgogo@email.com

# 입력 메소드

## Scanner Class

→ 사용자로부터 입력되는 정수, 실수, 문자열을 처리하는 클래스

## Scanner Class 사용 방법

→ 기존 기능제공 클래스와 사용법은 동일하며, 기능이 제공되는 메소드명을 기억하면 됨

→ JAVA 설치 시 기본적으로 제공되는 기능제공 클래스임

### 1. Import

```
import java.util.Scanner;
```

### 2. Scanner 생성

```
Scanner sc = new Scanner(System.in); → 클래스명 약어 = new 클래스명(System.in);
```

### 3. 키보드 입력 값을 받는 메소드

i) 정수 : `sc.nextInt();`

ii) 실수 : `sc.nextFloat();` 또는 `sc.nextDouble();`

iii) 문자열 : `sc.next();` 또는 `sc.nextLine();`

→ `sc.next();`는 띄어쓰기 입력 불가, `sc.nextLine();`은 띄어쓰기 가능

iv) 문자 : `sc.next().charAt(0);`





# 실습문제



# 변수를 통한 입·출력 실습

## 문제 1

자신의 이름(String), 나이(int), 주소(String), 키(double), 몸무게(double), 성별(char)을 입력 받고 출력하는 프로그램을 작성 하시오.

- 키는 정수로 변환하여 출력
- 몸무게는 첫째 자리까지 출력

## 문제 2

점수를 입력 받고 아래와 같이 출력 될 수 있는 프로그램 만들기



국어점수 입력 : 67

수학점수 입력 : 56

영어점수 입력 : 73

당신 성적의 총 합은 196점이고 평균은 65.33 입니다!





# 변수를 통한 입·출력 실습

## 문제 3

새 프로젝트 생성(프로젝트 명 자유롭게 대신 표기법은 확인)

### 1. 실행용 클래스

- 패키지명 : kh.java.var
- 클래스명 : Run
- 내용 : 기능제공 클래스의 exam1() 메소드 실행

### 2. 기능제공 클래스

- 패키지명 : kh.java.function
- 클래스명 : Example
- 메소드 명 : public void exam1(){}
- 내용 : 정수 두 개를 입력 받아 두수의 합(+), 차(-), 곱(\*), 나눗셈(/), 나눈 나머지(%)를 출력 ※ ( )안의 기호가 연산 기호임

## 실행결과 예시

첫번 째 정수 입력 : 23

두번 째 정수 입력 : 7

===== 결과 =====

더하기 결과 : 30

빼기 결과 : 16

곱하기 결과 : 161

나누기 몫 : 3

나누기 나머지 : 2



# 변수를 통한 입·출력 실습

## 문제 4

문제 3의 기능제공 클래스에 메소드 추가

- 메소드 명 : `public void exam2(){}`
- 내용 : 가로, 세로 값을 실수형으로 입력 받아 사각형의 면적과 둘레를 계산하여 출력
- 면적 :  $\text{가로} * \text{세로}$
- 둘레 :  $2 * (\text{가로} + \text{세로})$

## 실행결과 예시

```
가로 길이 입력 : 13.5
세로 길이 입력 : 41.7
===== 결과 =====
면적 : 562.95
둘레 : 110.4
```



# 변수를 통한 입·출력 실습

## 문제 5

문제 3의 기능제공 클래스에 메소드 추가

- 메소드 명 : `public void exam3(){}`
- 내용 : 영어 문자열을 입력 받고 첫번째, 두번째, 세번째 자리의 문자를 각각 출력

※ Hint : 문자를 입력 받는 메소드의 응용

## 실행결과 예시

영어단어 입력 : **apple**

첫번째 문자 : **a**

두번째 문자 : **p**

세번째 문자 : **p**



# 변수를 통한 입·출력 실습

## 문제 8

새 프로젝트 생성(프로젝트 명 자유롭게 대신 표 기법은 확인)

### 1. 실행용 클래스

- 패키지명 : kh.java.run
- 클래스명 : Run
- 내용 : 기능제공 클래스의 sample1() 메소드 실행

### 2. 기능제공 클래스

- 패키지명 : kh.java.practice
- 클래스명 : Sample
- 메소드 명 : public void sample1(){}  
- 내용 : 문자 하나를 입력 받아 그 문자의 유니코드 값 출력
- 메소드 명 : public void sample2(){}  
- 내용 : 실수형으로 국어, 수학, 영어 점수를 입력 받아 3과목의 총점과 평균 출력  
총점과 평균은 정수로 출력

## 실행결과 예시

sample1() 메소드 실행 결과

문자 입력 : A  
A의 유니코드 값 : 65

sample2() 메소드 실행 결과

국어 점수 입력 : 84  
수학 점수 입력 : 99  
영어 점수 입력 : 76  
===== 결과 =====  
총점 : 259  
평균 : 86

