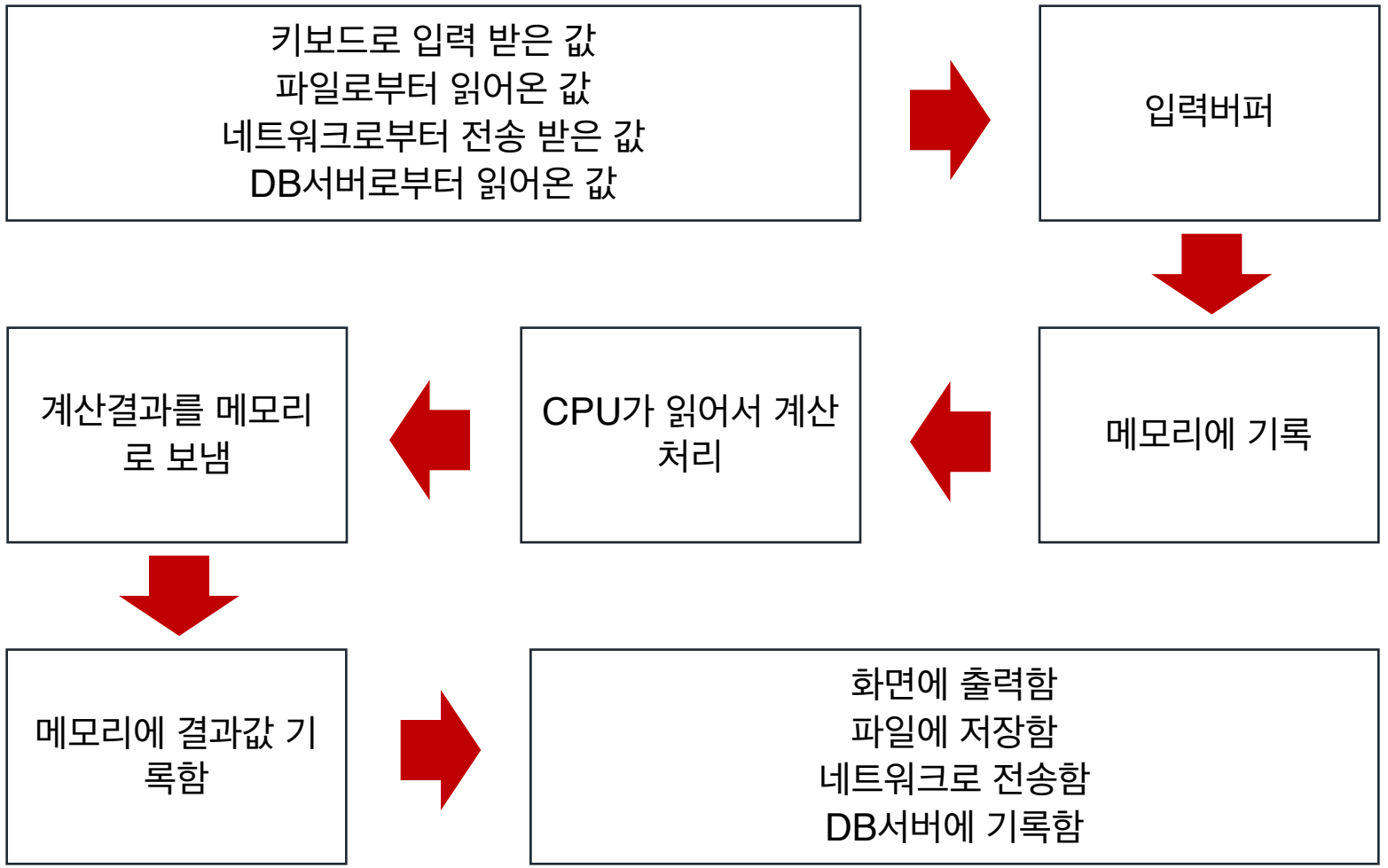
변수
(variable)

▶ 프로그램 작동 원리


키보드로 입력 받은 값
파일로부터 읽어온 값
네트워크로부터 전송 받은 값
DB서버로부터 읽어온 값
입력버퍼
메모리에 기록
CPU가 읽어서 계산 처리
계산결과를 메모리로 보냄
메모리에 결과값 기록함
화면에 출력함
파일에 저장함
네트워크로 전송함
DB서버에 기록함

▶ 데이터 저장 단위

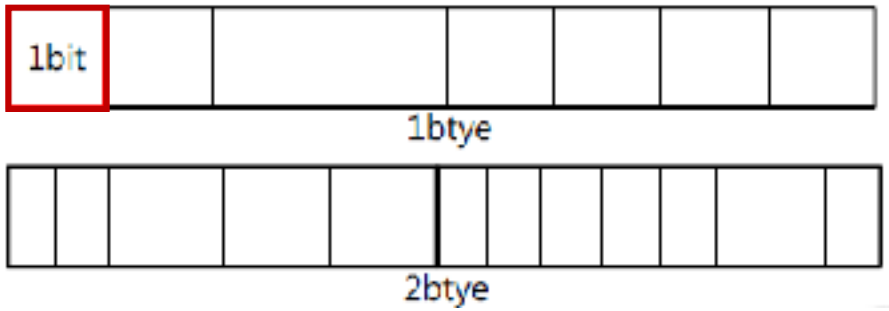
저장 공간은 제한 적이다. 따라서 저장 크기에 대한 기준과 CPU가 데이터를 처리할 때 일정한 기준이 필요하기 때문에 저장단위를 구성한다

✓ 비트(bit)

컴퓨터가 나타내는 데이터의 저장 최소 단위로서 2진수 값 하나를 저장할 수 있는 메모리공간을 의미

✓ 바이트(byte)

데이터 처리 또는 문자의 최소 단위로서 8개의 비트가 모여 하나의 바이트가 구성됨



▶ 변수란?

메모리 공간(RAM)에 한 개의 값을 기록하기 위한 장소(공간)

✓ 변수의 자료형(Type)

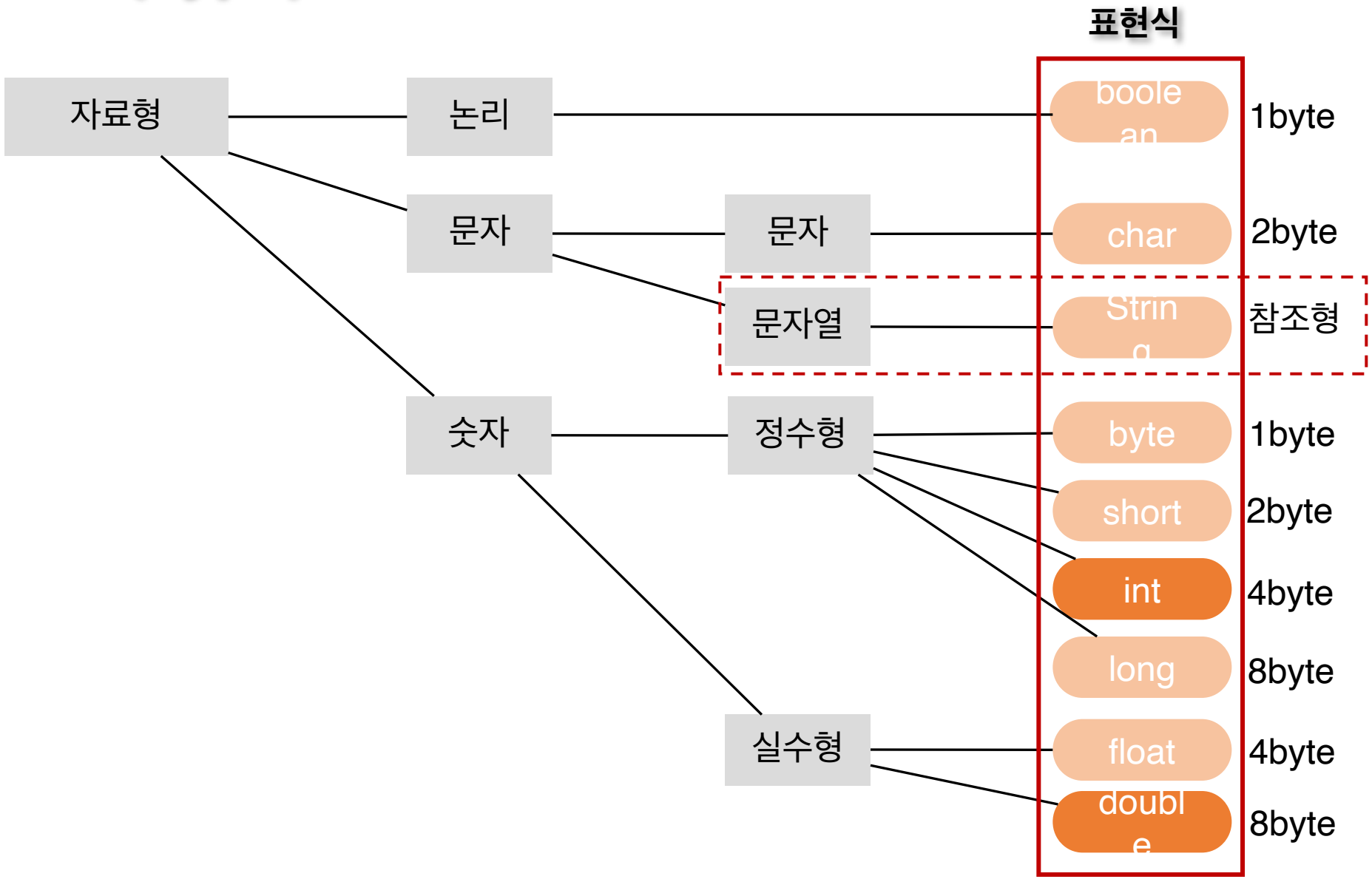
*** 기본형(Primitive Type)**

- 실제 데이터(값)을 저장
- 논리형, 문자형, 정수형, 실수형으로 나눠지고 8개의 자료형이 있음
- 각 자료형 별 데이터 저장크기가 다름

*** 참조형(Reference Type)**

- 데이터가 저장되어 있는 주소를 저장(객체의 주소)
- 기본형을 제외한 나머지(String 등), 사용자 정의 자료형
- 4byte의 공간을 저장공간으로 할당

▶ 자료형(Type)


표현식
자료형
논리
문자
숫자
문자
문자열
정수형
실수형
boole an
char
Strin g
byte
short
int
long
float
doubl e
1byte
2byte
참조형
1byte
2byte
4byte
8byte
4byte
8byte

▶ 변수 저장 가능 범위

자료형	범 위	크기(bit)
boolean	true, false	8
char	0~65,535(유니코드문자)	16
byte	-128 ~ 127	8
short	-32,768 ~ 32,767	16
int	-2,147,483,648 ~ 2,147,483,647	32
long	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807	64
float	±1.4E-45 ~ 3.4E38	32
double	±4.9E-324 ~ 1.8E308	64

컴퓨터는 2진수로 인지하기 때문에 $2^n(n = \text{비트크기})$로 범위가 할당됨

▶ 변수의 선언

메모리 공간에 데이터를 저장할 수 있는 공간을 할당하는 것

자료형 변수명 ; 마침

변수타입지정 변수명지정

✓ 선언 예시

```
// 논리형 변수 선언
boolean isTrue;

// 문자형 변수 선언
char ch;

//문자열 변수 선언
String str;

// 정수형 변수 선언
byte bnum;
short snum;
int inum;
long lnum;

// 실수형 변수 선언
float fnum;
double dnum;
```

▶ 변수의 명명 규칙

1. 대소문자가 구분되며 길이 제한이 없다.

2. 예약어를 사용하면 안 된다.

3. 숫자로 시작하면 안 된다.

   ex) age1은 가능하지만 1age는 불가능

4. 특수문자는 '_'와 '$'만을 허용한다.

   ex) sh@rp는 불가능하지만 $harp는 가능

5. 여러 단어 이름은 단어의 첫 글자를 대문자로 한다.

   ex) ageOfVampire, userName

▶ 주요 예약어

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

▶ 변수의 초기화

변수를 사용하기 전에 처음으로 값을 저장하는 것

→ 지역변수는 반드시 초기화 해야 된다.

✓ **선언 후 초기화**

```
int age;

age = 100;
```

✓ **선언과 동시에 초기화**

```
int age = 100;
```

▶ 값 대입과 리터럴

✓ 값 대입

생성한 변수(저장 공간)에 값을 대입하는 것

```
int age;
age = 10;
age = 20;
```

* 변수는 한 개의 데이터만 보관, 마지막에 대입한 값만 보관

✓ 리터럴

변수에 대입되는 값 자체를 의미한다.

```
short s = 32767;
int i = 100;
long l = 10000L;
float f = 0.123f;
double d = 3.14;
```

```
char c = 'A';
String str = "ABC";
```

▶ 상수란?

수학에서는 변하지 않는 값을 의미하고, 컴퓨터에서는 한번만 저장할 수 있는 공간으로 초기화 이후 다른 데이터(값)을 대입할 수 없다.

✓ 상수 선언 예

```
final int age;
final double PI = 3.14;
```

▶ 문자열

✓ 문자열 표현

컴퓨터에서 “기차“, “출력하세요“등과 같이 단어나 문장을 문자열이라고 표현하고, “”로 묶여 있으면 문자열로 인식한다.

Java에서는 String 객체를 이용하여 저장한다.

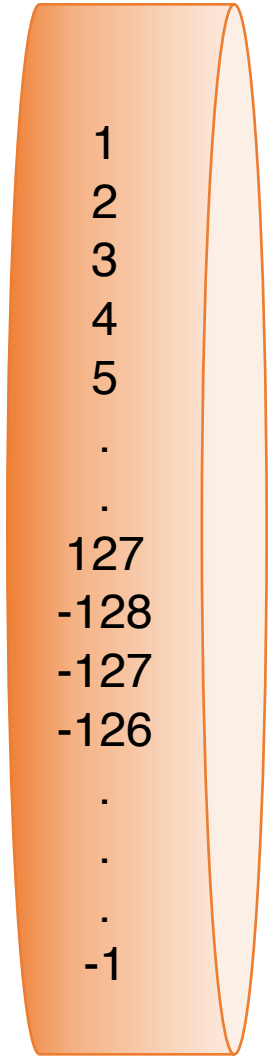
✓ 문자열 초기화

String str = “기차”;
String str = new String(“기차”);
String str = “기차” + “칙칙폭폭”;
String str = new String(“기차” + “칙칙폭폭”);
String str = “기차” + 123 + 45 + “출발”;
String str = 123 + 45 + “기차“ + “출발”;

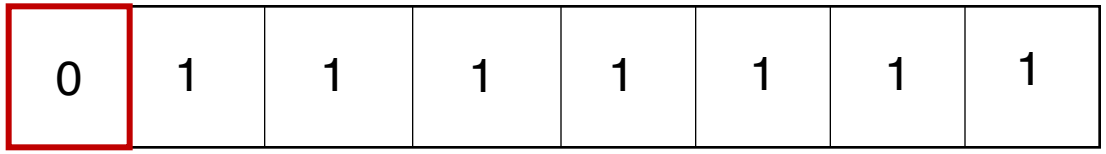
?

다른 자료형 + “문자열” → 문자열
“문자열” + 다른 자료형 → 문자열

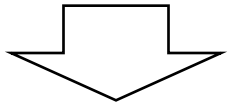
▶ 데이터 오버플로우

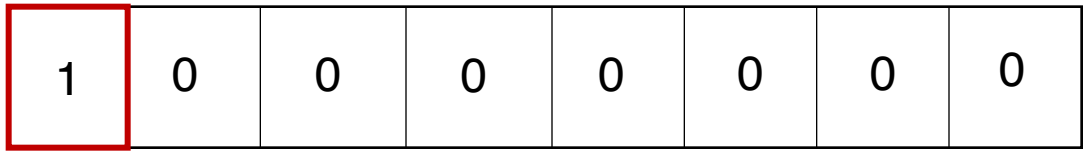


byte형



127+1을 하면 범위를 초과한 128이 되고 허용된 범위 이상의 비트를 침범하게 되는데 이를 **오버플로우**라고 한다.





byte형 허용범위 최소값인 -128이 되는 것이다.

▶ 형변환(casting)

✓ 컴퓨터의 값 처리 원칙

같은 종류 자료형만 대입 가능

같은 종류 자료형만 계산 가능

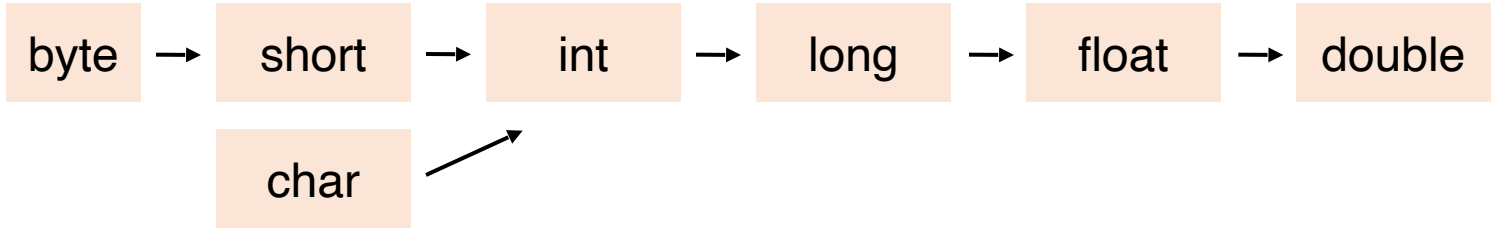
계산의 결과도 같은 종류의 값이 나와야 함

→ 이러한 원칙이 지켜지지 않은 경우에 형 변환이 필요함

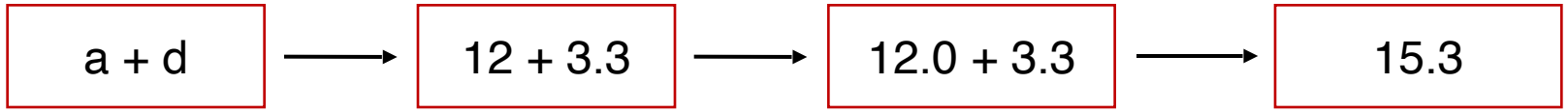
▶ 형변환(casting)

✓ 자동 형변환

연산시 컴파일러가 자동으로 형이 변환하는 것을 의미한다.



예시) int a = 12;
double d = 3.3;
double result = a + d;



*** 단, byte와 short 자료형 값의 계산 결과는 무조건 int로 처리한다.**

▶ 형변환(casting)

✓ byte와 short연산

Java에서는 byte와 short연산 시 자동으로 결과값을 int형으로 반환한다.

```
byte bnum = 10, bnum2 = 20;
byte result = bnum + bnum2;
int result = bnum + bnum2;
byte result = (byte)(bnum + bnum2);
```


에러

OK

OK

▶ 형변환(casting)

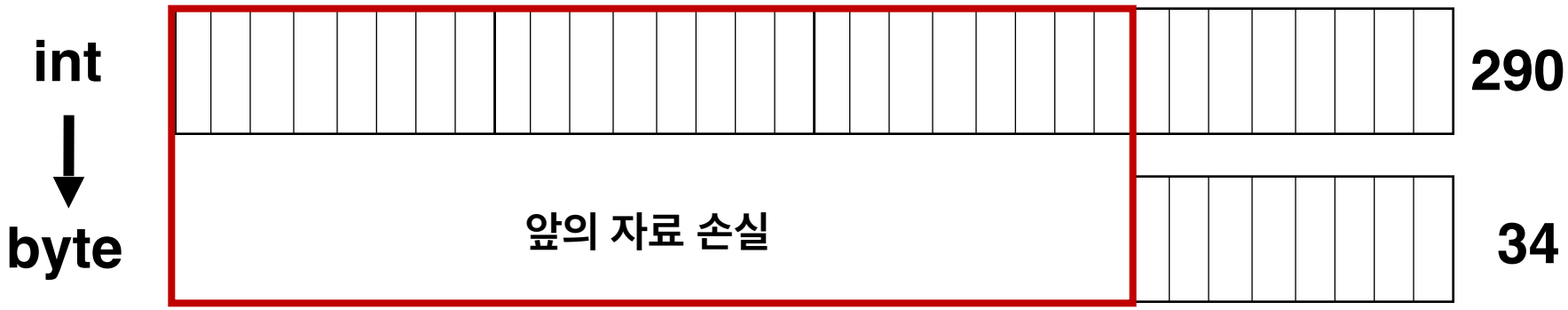
✓ 강제 형변환

자료형으로 형변환을 해줄 수 있다. 데이터가 큰 자료형에서 작은 자료형으로 변경 시

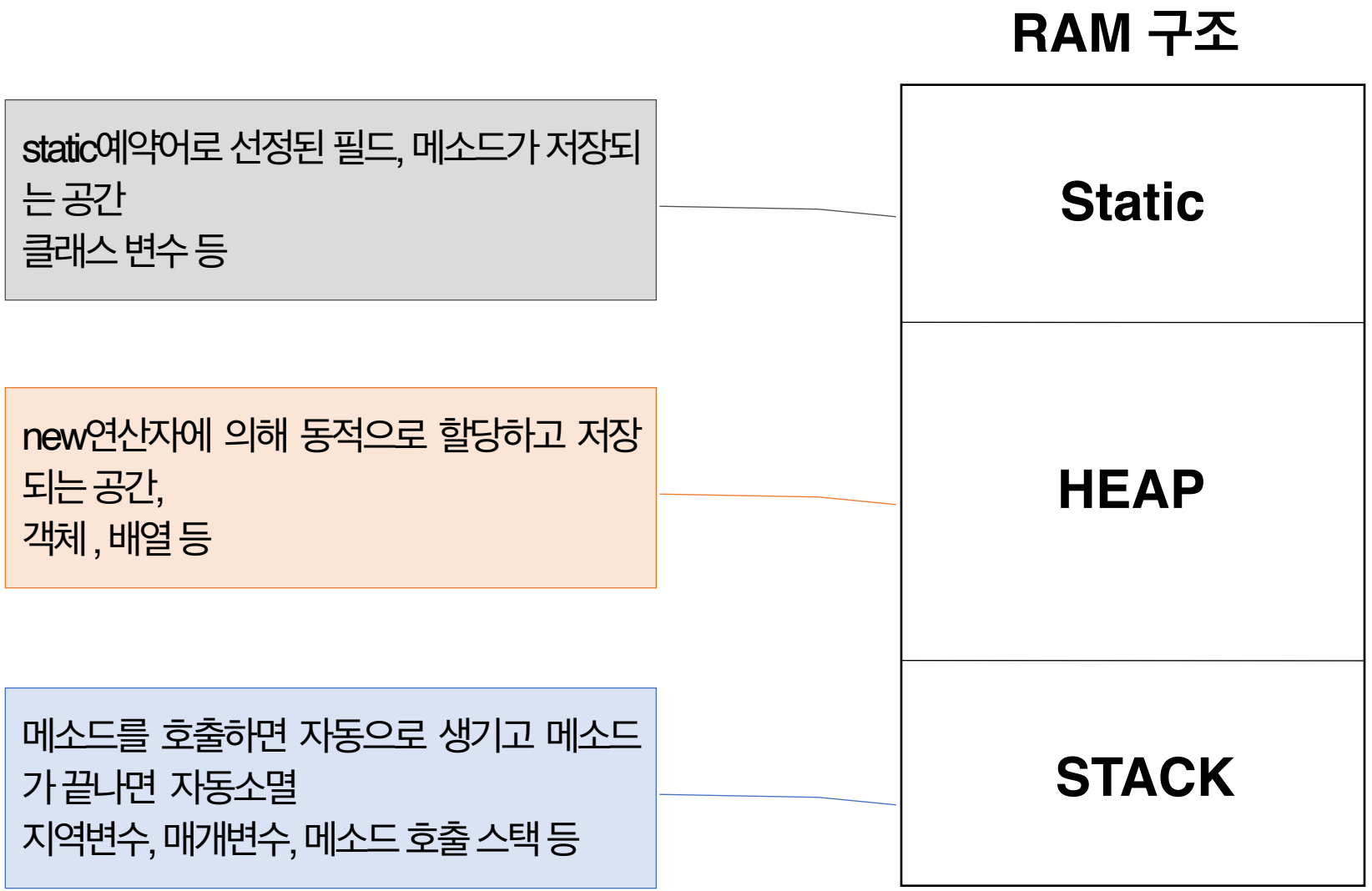
데이터 손실이 있을 수 있어 유의해야 한다.

```
double temp;
int name = (int)temp;
```

✓ 데이터 손실



▶ 변수와 메모리 구조


RAM 구조
static예약어로 선정된 필드, 메소드가 저장되는 공간
클래스 변수 등
new연산자에 의해 동적으로 할당하고 저장되는 공간,
객체 , 배열 등
메소드를 호출하면 자동으로 생기고 메소드가 끝나면 자동소멸
지역변수, 매개변수, 메소드 호출 스택 등
Static
HEAP
STACK

▶ 출력메소드

✓ System.out.print( )

() 안의 변수, 문자, 숫자, 논리 값을 모니터에 출력해주는 메소드

✓ System.out.println( )

print문과 동일하게 출력은 해주지만 출력 후 자동으로 출력창에 줄바꿈을 해주는 메소드

예) System.out.print(“안녕하세요”);
System.out.print(123);
System.out.print(변수명);

System.out.println(“안녕하세요”);
System.out.println(123);
System.out.println(변수명);

▶ 출력메소드

✓ System.out.printf(“%형식”, 변수 등)

정해져 있는 형식에 맞춰서 그 형식에 맞는 값(변수)을 줄바꿈 하지 않고 출력

%d : 정수형, %o : 8진수, %x : 16진수
%c : 문자, %s : 문자열
%f : 실수(소수점 아래 6자리), %e : 지수형태표현, %g : 대입 값 그대로
%A : 16진수 실수
%b : 논리형

정렬방법

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

▶ escape 문자

특수문자	문자 리터럴	비 고
tab	\t	정해진 공간만큼 띄어쓰기
new line	\n	출력하고 다음라인으로 옮김
역슬래쉬	\\	특수문자 사용시 백슬러시(\)를 넣고 특수문자를 넣어야 함
작은 따옴표	\’	
큰 따옴표	\”	
유니코드	\u	유니코드 표시할 때 사용

▶ Scanner

✓ Scanner Class

사용자로부터 입력되는 정수, 실수, 문자열을 처리하는 클래스

✓ import 작성

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

✓ Scanner 생성

```
Scanner sc = new Scanner(System.in);
```

✓ 키보드 입력값 받기

1. **정수 : sc.nextInt();**
2. **실수 : sc.nextFloat(); 또는 sc.nextDouble();**
3. **문자열 : sc.next(); 또는 sc.nextLine();**

next()는 띄어쓰기 입력불가
nextLine()은 문자열에 띄어쓰기 가능