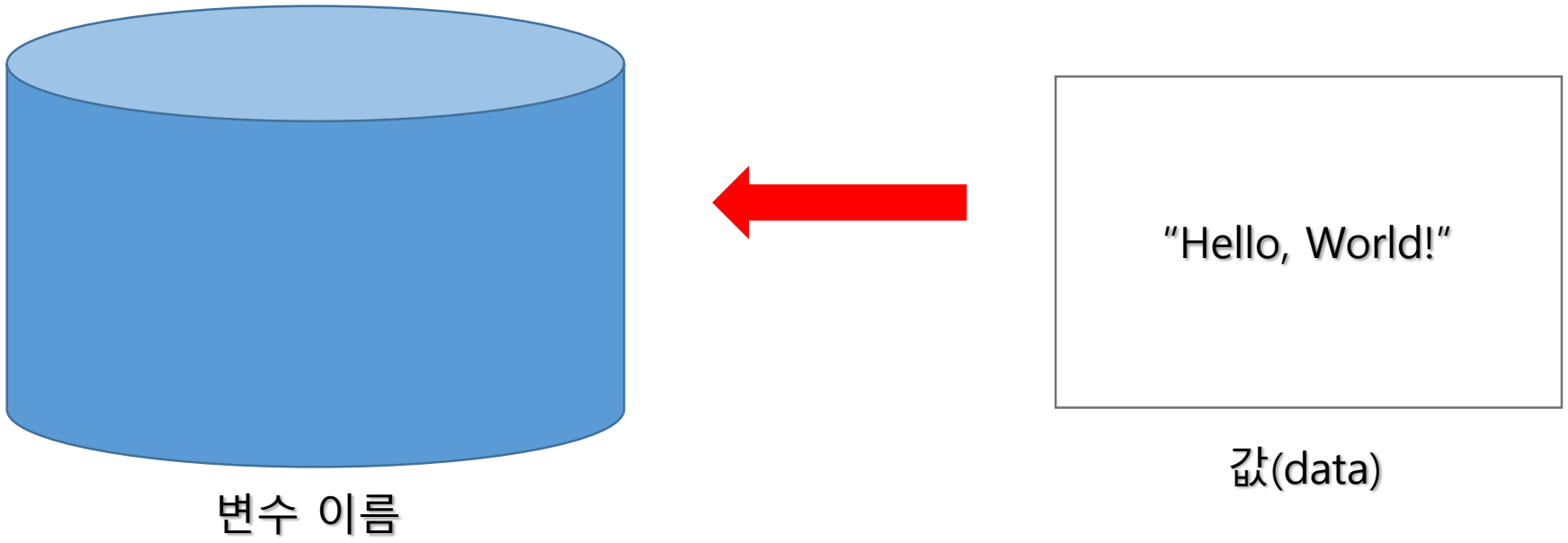
변수(Variable)

변수의 뜻

- **변수(Variable)**

어떠한 값(data)을 저장할 수 있는 저장 공간



변수를 사용하지 않을 경우

아래의 코드에서 100을 수정하기 위해서 총 7번의 수정이 필요함

FirstProject.java

```
1  package kr.co.greenart;
2
3  public class FirstProject {
4
5      public static void main(String[] args) {
6          System.out.println(100 + 30);
7          System.out.println(100 - 30);
8          System.out.println(100 * 30);
9          System.out.println(100 % 30);
10         System.out.println(100 / 30);
11         System.out.println(100 + 30 * 13);
12         System.out.println(100 / 30 + 13);
13     }
14
15 }
16
```

변수를 사용할 경우

아래의 코드에서 100을 수정하기 위해서 총 1번의 수정이 필요함

FirstProject.java

```
1  package kr.co.greenart;
2
3  public class FirstProject {
4
5      public static void main(String[] args) {
6          int a;
7          a = 100;
8          System.out.println(a + 30);
9          System.out.println(a - 30);
10         System.out.println(a * 30);
11         System.out.println(a % 30);
12         System.out.println(a / 30);
13         System.out.println(a + 30 * 13);
14         System.out.println(a / 30 + 13);
15     }
16
17 }
18
```

변수 선언

- **선언이란?**

  메모리에 값(data)을 저장할 공간을 확보하는 과정

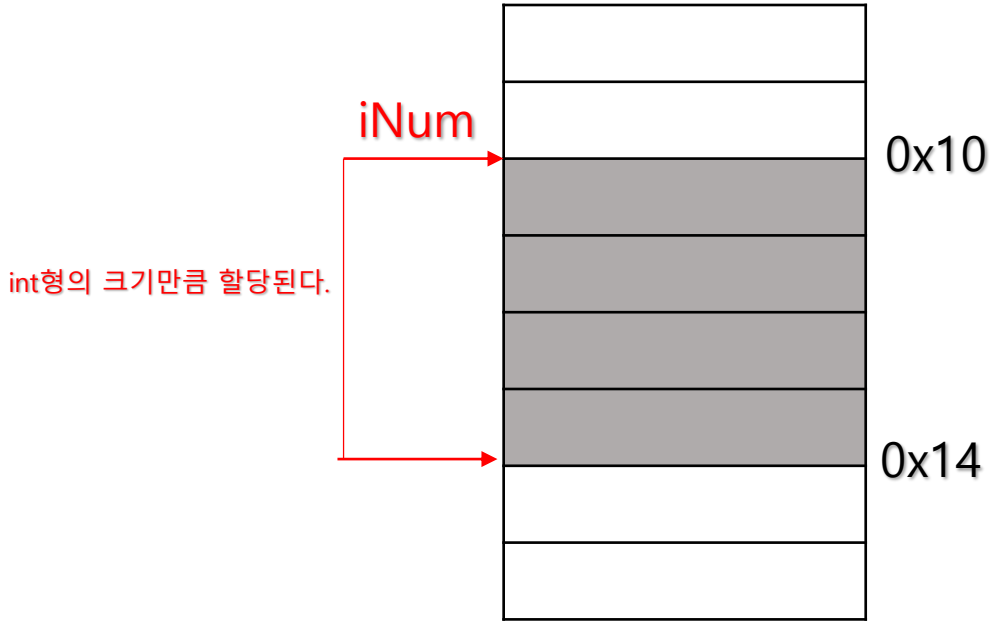
- **선언 방법**

  자료형 변수명 ;

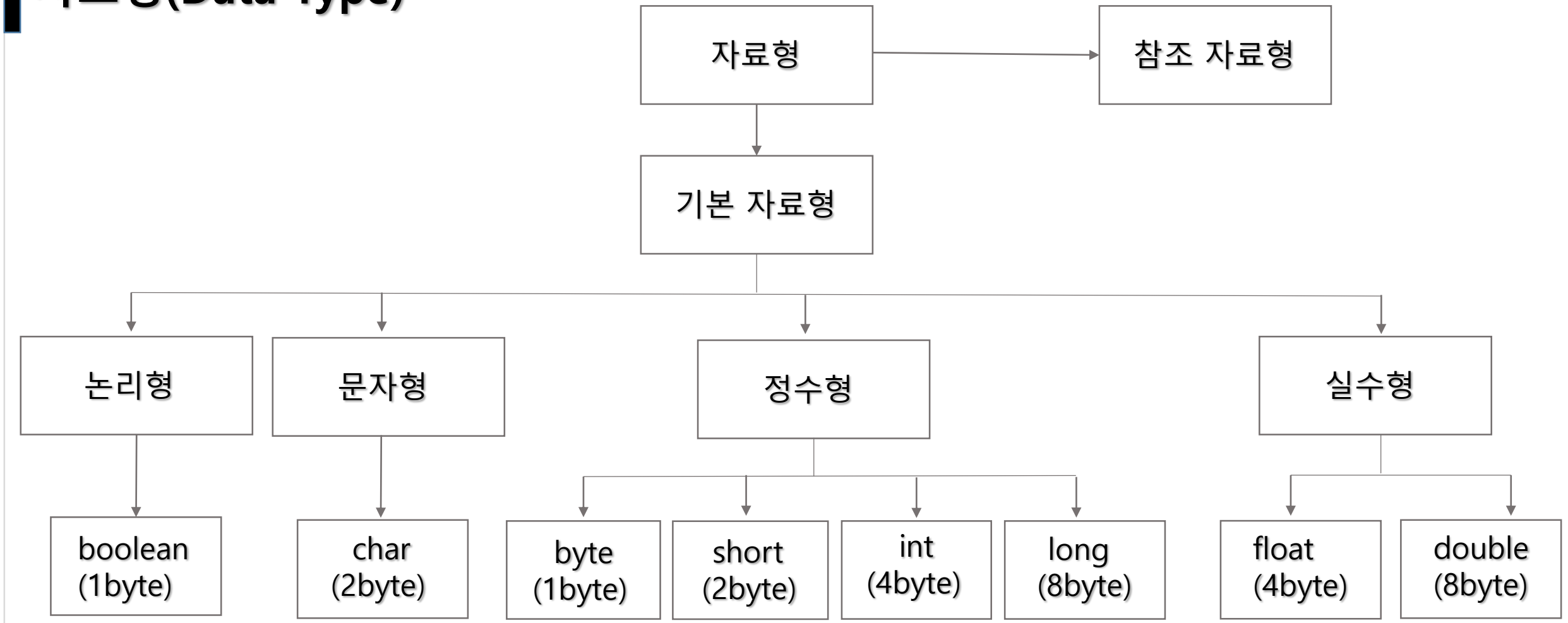
  데이터 타입 지정 변수이름 지정

```
// 정수형 변수 선언
int iNum;

// 정수형 다중 변수 선언
int iNum1, iNum2;
```



자료형(Data Type)



자료형의 크기

형태	예약어	메모리 크기	기본 값	표현 범위
논리형	boolean	1byte	false	true, false
문자형	char	2byte	'₩u0000'	0~65,535
정수형	byte	1byte	0	-128 ~ 127
	short	2byte	0	-32,768 ~ 32,767
	int	4byte	0	-2,147,483,648 ~ 2,147,483,647
	long	8byte	0L	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807
실수형	float	4byte	0.0f	±1.4E-45 ~ 3.4E38
	double	8byte	0.0d	±4.9E-324 ~ 1.8E308

- boolean의 표현 범위는 true 또는 false , 즉 두개인데 1byte의 크기를 가지고 있는 이유는 CPU가 메모리에 접근할 수 있는 최소 단위가 1byte이기 때문이다.

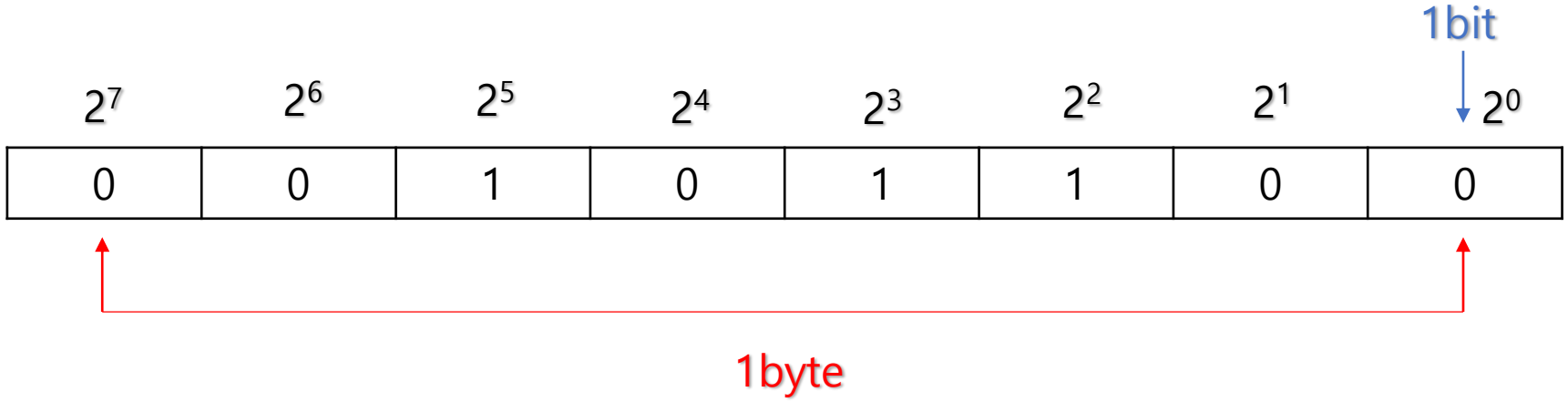
바이트(byte)와 비트(bit) 이해하기

- **비트(bit)**

  컴퓨터에서 정보를 표현하기 위한 최소 단위로써 0 또는 1로 표현됨

- **바이트(byte)**

  1byte는 8개의 bit로 이루어진 단위를 의미함 (1byte = 8bit)



변수명 네이밍 규칙 - 1

1. **대소문자를 구분하며 길이 제한이 없음**

2. **예약어 사용 불가**
   - int, true, public, final, if, …

3. **숫자로 시작할 수 없음**
   - 1num(X), num1(O)

4. **특수문자는 _ 와 $ 만 허용**
   - @num(X), $num(O)

5. **관례적으로 카멜 표기법(낙타 표기법)을 사용한다.**

변수명 네이밍 규칙 - 2

- **카멜 표기법(Camel Notation : Camel case : 낙타 표기법)**

  소문자로 시작하며 이어지는 단어들의 시작을 대문자로 작성하는 표기법
  ex) userName, boardList

- **파스칼 표기법(Pascal case)**

  대문자로 시작하며 이어지는 단어들의 시작또한 대문자로 작성하는 표기법
  자바에서는 클래스의 이름을 지을 때 사용됨
  ex) UserName, BoardList

- **스네이크 표기법(Snake case)**

  단어 사이에 언더바(_)를 활용하는 표기법
  ex) user_name, board_list

변수 초기화

- **초기화**

변수에 값을 할당하는 과정

```
// iNum 변수에 값 10을 할당
iNum = 10;
       대입 리터럴
```

* 리터럴 : 변수에 넣을 데이터(값)

- **명시적 초기화(Explicit initalization)**

변수의 선언과 동시에 초기화 하는 것

```
// 명시적 초기화
int iNum = 10;
```

상수

- **상수**

프로그래밍 언어에서의 상수는 단 한번만 저장되고 이후 변하지 않는 값을 의미함
변수명은 대문자에 언더바(_)를 넣어 대문자로 이루어진 스네이크 표기법을 사용하는것이 관례이다.

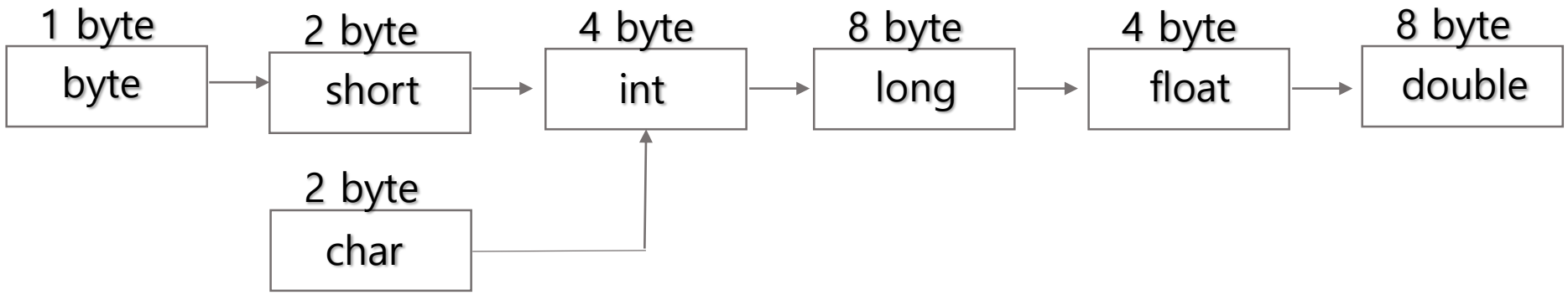
```
// 상수
final int NUM = 100;
```

```
// 상수
final int MEMBER_COUNT = 58;
```

형변환-1

- **형변환(Casting)**

변수에 이미 선언된 데이터 타입을 변경하는 것을 형변환 또는 타입변환이라 함
형변환은 크기가 작은 타입이 큰 타입에 저장되어야 데이터 손실이 발생하지 않는다.



형변환-2

- 예시

'A' (char) → 97 (int)

1234L (long) → 1234.0 (float)

1234 (int) → 1234.0 (double)

- 문자 'A'가 숫자 97로 변환되는 이유

컴퓨터는 0과 1로만 이루어진 명령을 수행할 수 있기 때문에
문자 A를 입력해도 ASCII(아스키 코드)에 의해 정의된 숫자로 변경됨

- char → int , long → float으로 형변환(캐스팅)이 가능한 이유

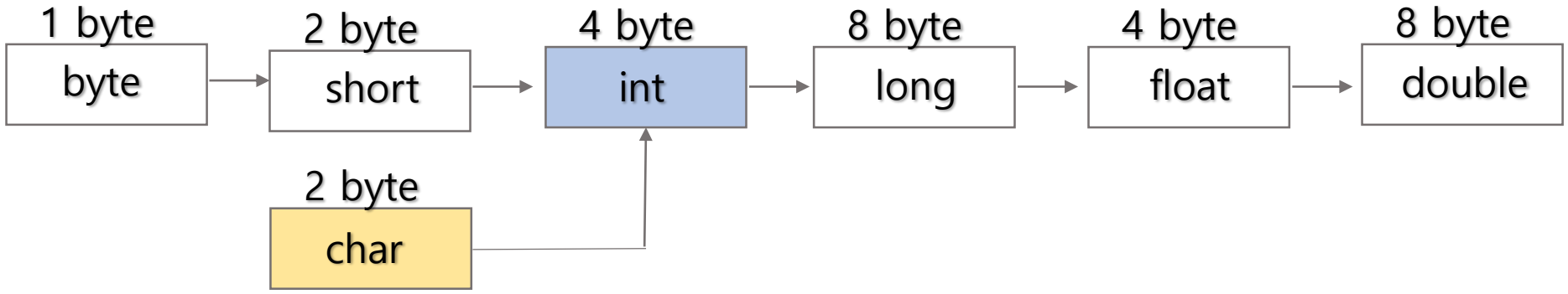
메모리 설계상 정수 타입 보다 실수 타입의 표현 범위가 더 넓기 때문이다.

자동 형변환

- **자동 형변환 (암시적 형변환)**

프로그래머가 직접 형변환을 해주지 않아도, 컴파일 과정에서 자동으로 타입 변환
범위가 작은 자료형 → 범위가 큰 자료형

```
// 자동 형변환
char ch = 'a'; 97
int num = ch;
System.out.println(num);
```



강제 형변환-1

- **강제 형변환 (명시적 형변환)**

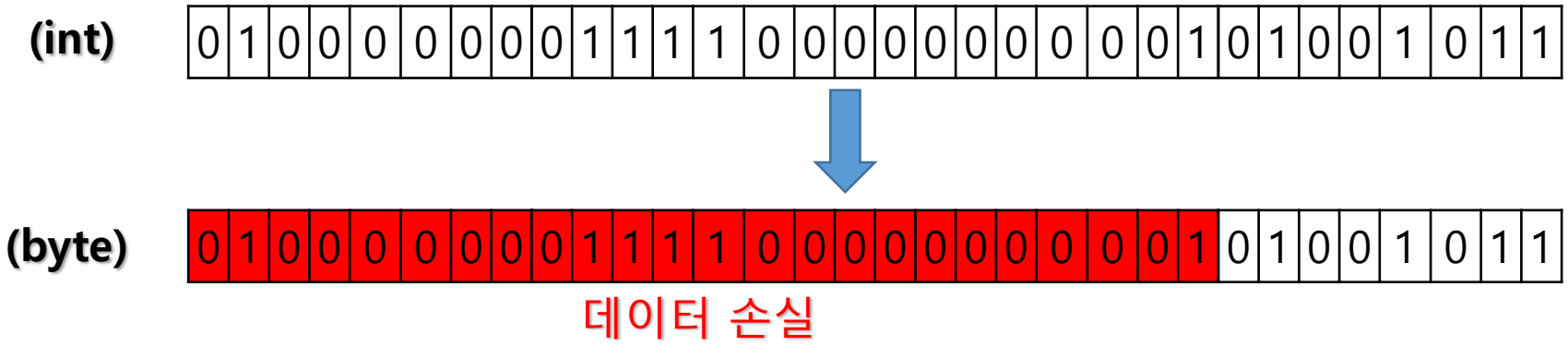
프로그래머가 직접 형변환을 해야하며, 하지 않을 경우 에러 발생
범위가 큰 자료형 → 범위가 작은 자료형

```
// 자동 형변환
double dNum = 12346.5;
int iNum = (int)dNum;
System.out.println(iNum); 결과 : 12345
```

강제 형변환-2

- **강제 형변환의 단점**

범위가 큰 자료형에서 범위가 작은 자료형으로 변환하기 때문에 데이터 손실을 발생시킬 수 있음



자바 메모리 구조 이해하기

- **스택(Stack) 영역**

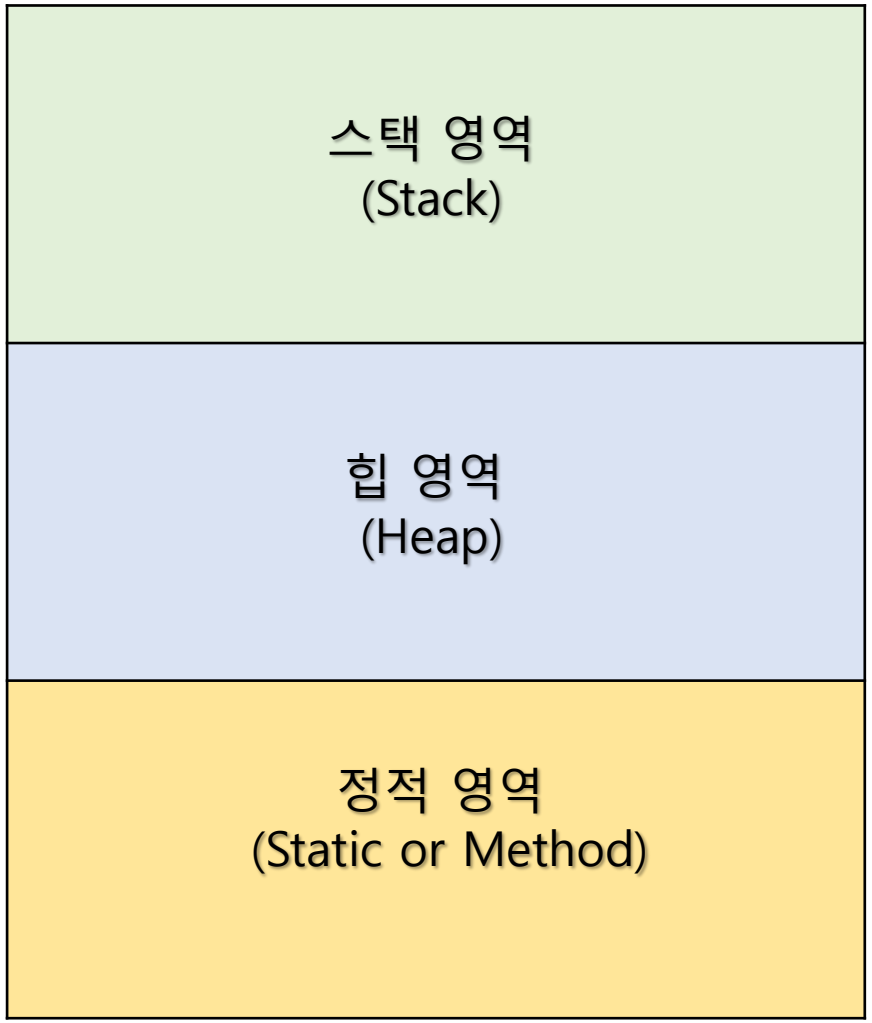
  메서드 호출 시 자동 생성 및 종료 시 자동 소멸
  매개변수, 지역변수, 리턴 값, 연산 값 등을 임시 저장하는 영역

- **힙(Heap) 영역**

  new 키워드로 생성된 객체와 배열이 동적으로 저장되는 영역

- **정적(Static or Method) 영역**

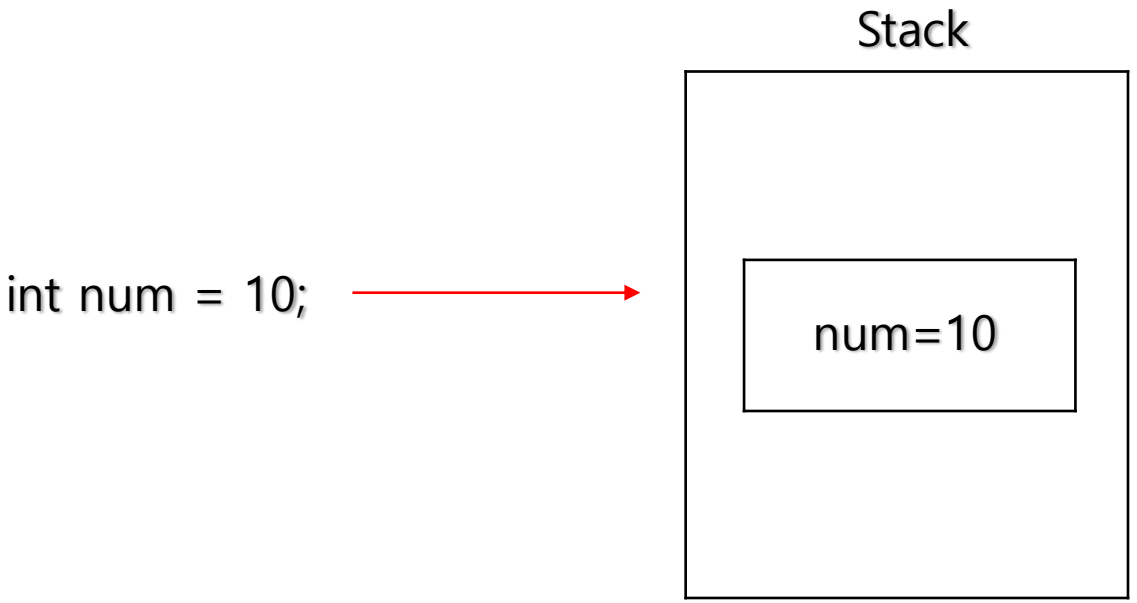
  클래스, static, 클래스 변수 등을 저장하는 영역



기본형 변수 vs 참조형 변수

- **기본형 변수(Primitive Variables)**

자바에서 미리 정의되어 있는 데이터 타입으로써 정수, 실수, 문자, 논리가 있음
이러한 기본형 변수는 스택 메모리에 직접 저장되며 값을 가지고 있다.

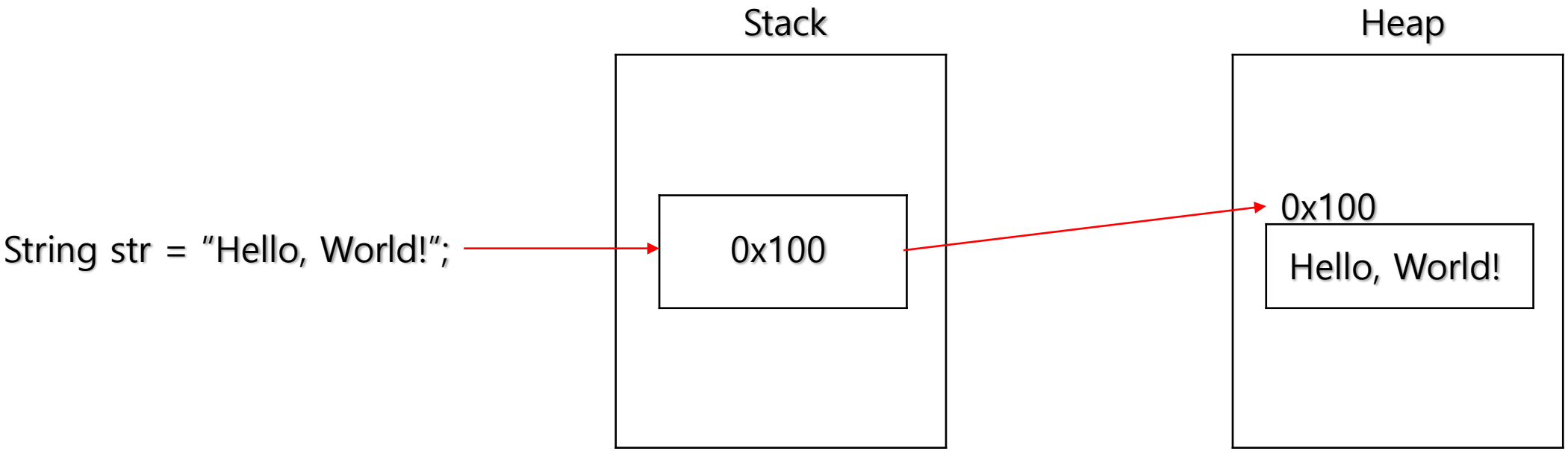


참조형 변수

- **참조 변수(Reference Variables)**

참조형 변수는 클래스, 인터페이스, 배열 등의 데이터 타입이 있으며 자바에서 미리 정의해둔 데이터 타입과 개발자가 직접 선언하여 사용할 수 있음

참조형 변수의 값은 힙 메모리에 저장된다.



스캐너

- **Scanner Class**

  사용자로부터 값(data)를 입력받는 기능을 제공하는 클래스

- **사용 방법**

1. Scanner Class 불러오기

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

2. Scanner 객체 생성하기

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

3. 스캐너로부터 문자열 입력 받고 출력하기

```
System.out.println("문자열을 입력하세요: ");
String str = sc.nextLine();

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

정수 : sc.nextInt();
실수 : sc.nextFloat(); / sc.nextDouble();
문자열 : sc.next(); / sc.nextLine

next() : 주로 단어를 처리할 때 사용되며 띄어쓰기(space) 전까지 입력 받은 데이터를 반환

nextLine() : 주로 문자열을 처리할 때 사용되며 엔터를 치기 전까지 입력받은 데이터를 반환

출력 메소드 - 1

- **System.out.print()**

  괄호 () 안의 내용을 출력. 개행 문자(줄 바꿈=₩n) 포함되지 않음

- **System.out.println()**

  괄호 () 안의 내용을 출력. 개행 문자(줄바꿈=₩n) 포함

```
int num = 456;
System.out.print("안녕하세요");
System.out.print(123);
System.out.print(num);
```

→

```
안녕하세요123456
```

```
int num = 456;
System.out.println("안녕하세요");
System.out.println(123);
System.out.println(num);
```

→

```
안녕하세요
123
456
```

출력 메소드 - 2

- **System.out.printf("출력 서식", 출력할 내용)**

괄호 () 안에 형식에 맞춰 내용을 출력. 개행 문자(줄 바꿈=₩n) 포함되지 않음

- **정렬 방법**

- %10d : 왼쪽 10칸 공백, 오른쪽 정렬
- %-10d : 오른쪽 10칸 공백, 왼쪽 정렬
- %.5f : 소수점 아래 5자리 표시

```
int num = 456;
float fNum = 1.1234567890f;
System.out.printf("num 변수의 값은 %d 입니다. \n",num);
System.out.printf("fNum 변수의 값은 %f 입니다.",fNum);
```

개행 문자

```
num 변수의 값은 456 입니다.
fNum 변수의 값은 1.123457 입니다.
```

지시자	설명
%b	boolean 형식으로 출력
%d	정수
%o	8진수
%x	16진수
%f	실수(소수점 아래 6자리)
%c	문자
%s	문자열
%n	개행 문자(줄 바꿈)
%e	지수 형태 표현

이스케이프 문자

- **이스케이스(Escape)**

  백슬래시(₩) 뒤에 문자 또는 숫자가 오며, 특수한 기능을 수행하는 문자

이스케이프 문자	설명
₩n	줄 바꿈
₩t	탭
₩b	백스페이스
₩"	특수 문자를 문자로 인식 해야할 때 사용 ex) ""Hello, World!"" 라는 문자열이 있을 때 "를 포함하여 출력하고 싶다면 "₩"Hello, World!₩"" 와 같이 작성하면 된다.
₩'	