Java

Java데이터 타입

데이터 타입

진법

- 지정된 범위의 수로 표현하는 방법

진법	범위	표현식	사용 예
2진수	0, 1		0100 0001
8진수	0 ~ 7	0(숫자)	0101
10진수	0 ~ 9		65
16진수	0 ~ 9, A ~ F	0x	0x41

데이터 타입

데이터 타입

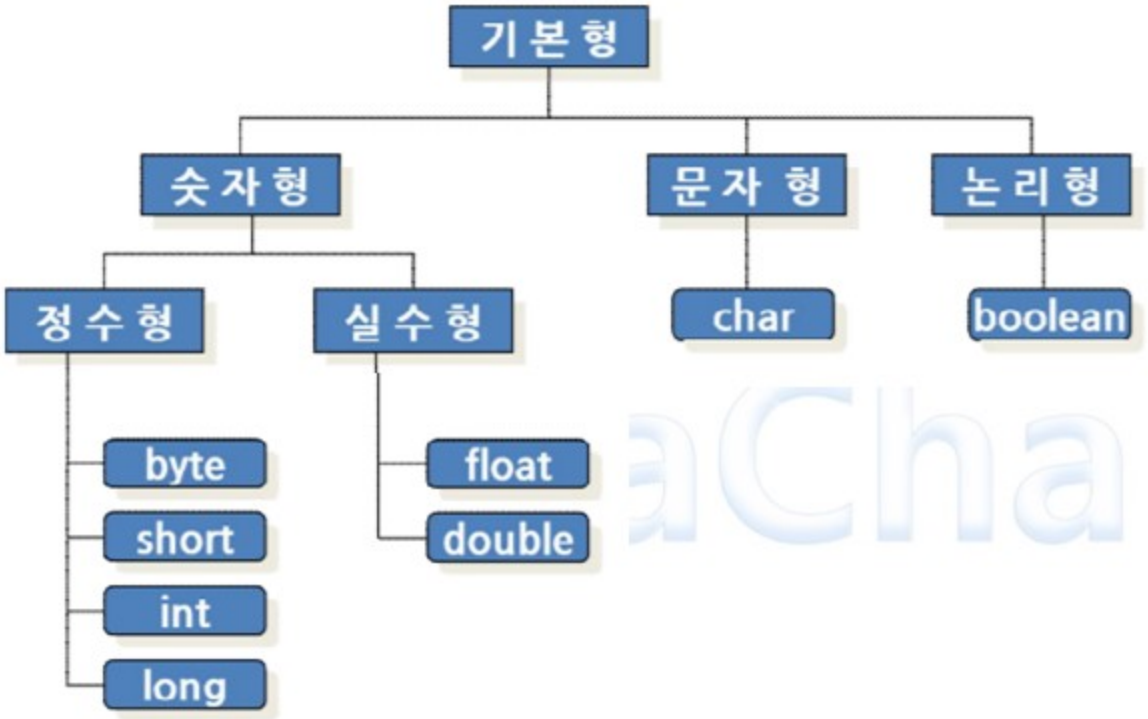
- 데이터 타입
  - 정수, 실수, 문자를 구분하며, 또한 데이터의 범위나 부호의 사용여부 등을 결정짓는 것을 말한다.

- 기본형
  - boolean, char, byte, short, int, long, float, double

- 참조형
  - 기본형을 제외한 나머지 타입
  - 객체의 주소 또는 null값을 저장한다

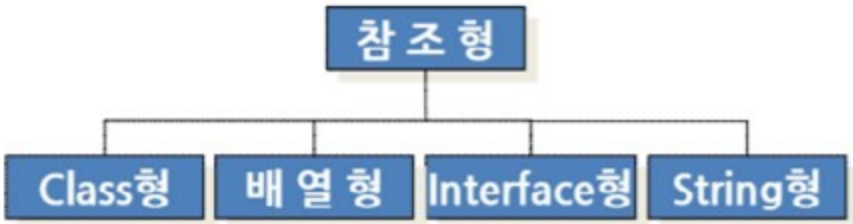
데이터 타입

- 기본자료형(primitive data type)



데이터 타입

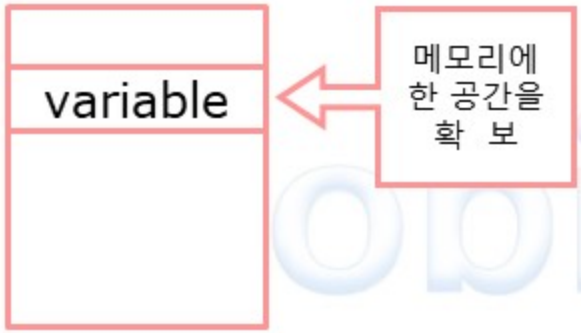
- 참조 자료형(reference data type)



데이터 타입

변수

- 변수는 데이터를 저장하기 위한 공간으로, 저장된 값의 변경이 가능



데이터 타입

변수

- 변수 선언
  - 변수 선언을 하면 메모리 공간을 할당 받아 사용할 수 있게 된다.

- 변수 선언
  - 데이터형 변수명;
  - 예) int i;

- 같은 형의 변수를 콤마(,) 연산자를 이용해서 선언할 수 있다.
  - 데이터형 변수명1, 변수명2, 변수명3, ... ,변수명n;
  - 예) int i, j, k;

데이터 타입

변수

- 변수명 작성규칙
  - 첫 변수 명에 숫자는 올 수 없다.(반드시 영문자나 밑줄문자로 시작한다.)
  - 대 . 소문자를 구분한다.
  - 특수문자는 '_'와 '$'만을 허용한다.
  - 예약어는 사용할 수 없다.

- 변수명 권장 규칙
  - 클래스 이름이 첫 글자는 항상 대문자
  - 변수와 메서드 이름의 첫 글자는 항상 소문자
  - 여러 단어로 이루어진 이름은 단어의 첫 글자를 대문자로 한다

- 변수의 올바른 형태
  - sun10, SUN10, m1, a_7, abc, _sum

- 변수의 잘못된 형태
  - 2m(첫글자 숫자), for(예약어), KBS@TV(특수문자), USN 10(공백)

데이터 타입

변수

- 예약어 (reserved word)
  - Java언어에서 사용하는 명령어
- 데이터 형 관련 예약어
  - Boolean, byte, char, int, float, short, long, double, enum, void
- 기억 관련 예약어
  - static
- 제어 관련 예약어
  - if, else, for, while, do, switch, case, break, continue, return
- 기타 예약어
  - abstract, class, catch, finally, implements, import, interface, package, super .....

데이터 타입

변수

- 변수 초기화
  - 선언된 변수에 값을 대입하는 과정
  - 변수명 = 값;
  - 예) i = 3;

- 문자를 쓸 때는 반드시 작은따옴표(' ')안에 써야 한다.

- 변수를 선언하면서 초기화 할 수 있다.
  - 데이터형 변수명 = 값;
  - 예) int i = 3;

데이터 타입

기본 데이터 타입

- 논리형
  - true와 false 중 하나의 값을 가지며, 조건식과 논리적 계산에 사용된다
- 문자형
  - 문자를 저장
- 정수형
  - 정수 값을 저장
  - byte형은 이진 데이터를 위해 사용
  - short형은 c언어와의 호환성을 위해 사용
  - int형이 기본 자료형
- 실수형
  - 실수 값을 저장
  - double형이 기본 자료형

데이터 타입

정수 데이터 형

- 소수점 이하를 표현하지 못하는 데이터 형
- Java에서는 unsigned를 사용 하지 않는다.
- JVM은 피 연산자를 4byte 변환하여 연산한다

정수형	범위	크기
short	-32,768 ~ 32,767	2 byte
int	-2,147,483,648~2,147,483,647	4 byte
long	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807	8 byte

데이터 타입

정수 데이터 형

- 변수 선언 예

```
byte b = 1;
short s = 2;
int i = 10;
long big = 100L; //long형 리터럴 상수에는 접미사 L을 붙여야 한다.

int octN = 010; //8진수 10
int hexN = 0x10; //16진수 10
```

데이터 타입

실수 데이터 형

- 소수점 이하를 표현할 수 있는 데이터 형
- 정수와 실수를 메모리에 저장하는 방식이 다르기 때문에 같은 4byte라도 정수 데이터 형보다 훨씬 큰 수와 작은 수를 저장
- 부동소수점형의 크기는 규정되어 있지 않다.
  - 실제크기는 시스템의 의존적.
  - float 형은 double 형의 크기보다 항상 작거나 같다.

데이터 타입

실수 데이터 형

- 고정 소수점 방식
  - 실수를 정수와 같은 방식으로 표현
  - 부호, 정수부와 소수부로 나눈다

- 부동소수점 방식
  - 부호, 지수부와 가수부로 나누는 방식
  - 가수의 정수부는 0이 아닌 한자리로 제한

정수형	범위	크기
float	$\pm 3.4*10^{-37} \sim \pm 3.4*10^{38}$	4 byte
double	$\pm 1.7*10^{-307} \sim \pm 1.7*10^{308}$	8 byte

데이터 타입

실수 데이터 형

- 변수 선언 예

```
float pi = 3.14f;       //float형 리터럴에는 접미사 f를 사용한다
float pi = 3.14;        //error : float형 변수에 double형 리터럴을 저장 할 수 없다

double df = 3.0e5d; //doubl형 리터럴에는 d를 사용한다
double df = 1.234;   //d 생략 가능
```

데이터 타입

문자형 데이터 형

- 문자 인코딩 방식으로 Unicode(2byte)를 사용해서 문자를 표현
- 문자열을 사용하려면 String class를 사용
  - String name = "홍길동";

데이터 타입

문자형 데이터 형

- BCD 코드
  - 2진수 코드를 10진수 코드로 표현
  - 상위 2비트 존 비트, 하위 4비트 데이터 비트
- 아스키 코드
  - 7비트 부호 체계
  - 상위 3비트 존 비트, 하위 4비트 데이터 비트
- 유니 코드
  - 4바이트로 표현할 수 있는 코드
- EUC-KR/CP949
  - 2바이트로 표현할 수 있는 코드

데이터 타입

참조 데이터 타입

- 참조형
  - 사용자가 정의 할 수 있다.
  - 클래스 이름이 변수의 타입이 된다
  - 참조형 변수 간의 연산을 할 수 없다.

- 참조형 변수 선언 예
  - 클래스이름 변수명; //타입이 클래스이름인 것은 모두 참조 변수
  - Date today = null;
  - Date today = new Date();

데이터 타입

상수

- 한번 값을 초기화하면 변경할 수 없는 공간
- 선언하면서 바로 초기 값을 설정해야 한다.
- 상수 선언
  - final 데이터형 변수명 = 초기값;
  - final double PI = 3.141592;

데이터 타입

상수

- 심볼릭(Symbolic) 상수
  - 이름을 갖는 상수로 final 키워드 붙여 선언한 변수
  - 변수 선언 앞에 final 키워드를 붙이면 상수가 된다.

- 리터럴(Literal) 상수
  - 이름을 갖지 않는 상수로 값 자체를 의미
  - int a = 3 + 5 라는 연산 식에서 3과 5는 상수, a는 변수

- 정수상수
  - 소수점 이하가 없는 데이터
  - 0으로 시작하면 8진수를 의미하며, 0x로 시작하면 16진수
  - 4byte로 메모리에 저장되는데. 뒤에 L(l)을 붙이면 long형으로 표현된다.

- 실수형 상수
  - 소수점 이하가 있는 데이터로 지수 형태도 포함
  - 8byte로 메모리에 저장
  - 4byte에 저장하고자 할 때는 실수 상수 뒤에 F(f)를 붙이면 된다.

데이터 타입

상수

- 문자 상수
  - 반드시 작은따옴표(' ')안에 한 글자로 표기
  - 제어문자(Escape Sequence)도 문자 상수에 포함

- 문자열 상수
  - 두 글자 이상
  - 큰 따옴표(" ")를 사용하여 표기

데이터 타입

자료형 변환

- boolean형을 제외한 기본형에서 서로 형 변환이 가능하다.
  - 캐스트 연산자를 이용하여 형변환을 구현 할 수 있다.

- cast 연산자에 의한 자료형 변환
  - 작은 자료형에서 큰 자료형의 변환은 캐스트 연산자 생략 가능
  - (자료형) 변수 & 상수

데이터 타입

날짜 데이터

- Java 8 부터 java.time(joda.time) api
  - java.util.Date > java.util.Calendar > java.time(org.joda.time)

- LocalDate
  - 로컬 날짜 클래스로 날짜 정보 관리
  - LocalDate currentDate = LocalDate.now();

- LocalTime
  - 로컬 컴퓨터의 현재 시간 정보
  - LocalTime currentTime = LocalTime.now();

- LocalDateTime
  - 로컬 컴퓨터의 현재 날짜와 시간 정보
  - LocalDateTime currentDateTime = LocalDateTime.now();

데이터 타입

날짜 연산

- java.time.LocalDateTime
  - plusYears()
  - plusMonths()
  - plusWeeks()
  - plusDays()
  - plusHours()
  - plusMinutes()
  - plusSeconds()
  - plusNanos()

데이터 타입

날짜 연산

- 날짜 비교
  - LocalDate startDate = LocalDate.of(2000, 01, 01);
  - LocalDate endDate = LocalDate.now();

  - startDate이 endDate 보다 이전 날짜 인지 비교
    a. startDate.isBefore(endDateTime);

  - 동일 날짜인지 비교
  - startDate.isEqual(endDate);

  - startDate이 endDate 보다 이후 날짜인지 비교
    a. startDate.isAfter(endDate);

데이터 타입

날짜 연산

- 시간 비교
  - 나노초가 존재할 경우 나노초의 시간까지 비교.

  - LocalTime startTime = LocalTime.of(23, 59, 59);
  - LocalTime endTime = LocalTime.now();

  - startTime이 endTime 보다 이전 시간 인지 비교
    a. startTime.isBefore(endTime);

  - startTime이 endTime 보다 이후 시간 인지 비교
    a. startTime.isAfter(endTime);

데이터 타입

날짜 연산

- 날짜 차이 계산
  - 31일은1개월 1일로 처리한다.

  - period.getYears();
  - period.getMonths();
  - period.getDays();

  - 전체 년 차이

    a. ChronoUnit.YEARS.between(startDate, endDate)

  - 총 월 차이

    a. ChronoUnit.MONTHS.between(startDate, endDate)

  - 총 일 차이

    a. ChronoUnit.DAYS.between(startDate, endDate)

데이터 타입

날짜 연산

- 시간 차이 계산
  - LocalTime startTime = LocalTime.of(0,0,0);
  - LocalTime endTime = LocalTime.now();

  - Duration duration = Duration.between(startTime, endTime);

  - duration.getSeconds()
  - duration.getNano()

데이터 타입

날짜 포맷

- 2024년 9월 11일 오전 11시 7분
  - LocalDateTime now = LocalDateTime.now();
  - DateTimeFormatter dateTimeFormatter = DateTimeFormatter.ofPattern("yyyy년 M월 d일 a h시 m분");
  - now.format(dateTimeFormatter);

- 2024-09-11 11:09:59
  - 12시간 표기 : 01시는 13시

  - LocalDateTime now = LocalDateTime.now();
  - DateTimeFormatter dateTimeFormatter = DateTimeFormatter.ofPattern("yyyy-MM-dd hh:mm:ss");
  - now.format(dateTimeFormatter);

데이터 타입

날짜 변환

- LocalDate -> String
  - LocalDate.of(2020, 12, 12).format(DateTimeFormatter.BASIC_ISO_DATE);
- LocalDateTime -> String
  - LocalDateTime.now().format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss"));
- LocalDate -> java.sql.Date
  - Date.valueOf(LocalDate.of(2019, 12, 27));
- LocalDateTime -> java.util.Date
  - Date.from(LocalDateTime.now().atZone(ZoneId.systemDefault()).toInstant());
- LocalDateTime -> java.sql.Timestamp
  - Timestamp.valueOf(LocalDateTime.now());

데이터 타입

날짜 변환

- String -> LocalDate
  - LocalDate.parse("1995-05-09");
  - LocalDate.parse("20191224", DateTimeFormatter.BASIC_ISO_DATE);
- String -> LocalDateTime
  - LocalDateTime.parse("2019-12-25T10:15:30");
  - LocalDateTime.parse("2019-12-25 12:30:00", DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss"));
- java.util.Date -> LocalDateTime
  - LocalDateTime.ofInstant(new Date().toInstant(), ZoneId.systemDefault());
- LocalDateTime -> LocalDate
  - LocalDate.from(LocalDateTime.now());
- LocalDate -> LocalDateTime
  - LocalDate.now().atTime(2, 30);