[Java] 오토 박싱 & 오토 언박싱

자바에는 기본 타입과 Wrapper 클래스가 존재한다.

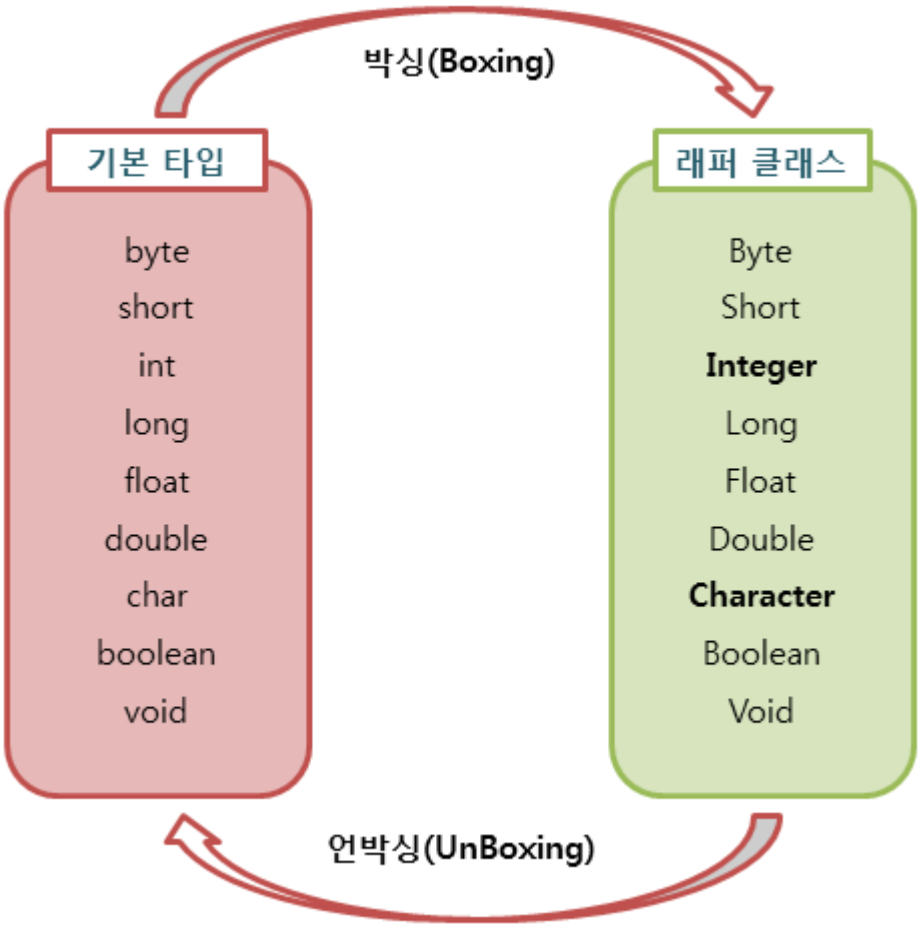
- 기본 타입 : `int, long, float, double, boolean` 등
- Wrapper 클래스 : `Integer, Long, Float, Double, Boolean` 등

박싱과 언박싱에 대한 개념을 먼저 살펴보자

> 박싱 : 기본 타입 데이터에 대응하는 Wrapper 클래스로 만드는 동작
>
> 언박싱 : Wrapper 클래스에서 기본 타입으로 변환

```
// 박싱
int i = 10;
Integer num = new Integer(i);

// 언박싱
Integer num = new Integer(10);
int i = num.intValue();
```



오토 박싱 & 오토 언박싱

JDK 1.5부터는 자바 컴파일러가 박싱과 언박싱이 필요한 상황에 자동으로 처리를 해준다.

```
// 오토 박싱
int i = 10;
Integer num = i;

// 오토 언박싱
Integer num = new Integer(10);
int i = num;
```

성능

편의성을 위해 오토 박싱과 언박싱이 제공되고 있지만, 내부적으로 추가 연산 작업이 거치게 된다.

따라서, 오토 박싱&언박싱이 일어나지 않도록 동일한 타입 연산이 이루어지도록 구현하자.

오토 박싱 연산

```
public static void main(String[] args) {
    long t = System.currentTimeMillis();
    Long sum = 0L;
    for (long i = 0; i < 1000000; i++) {
        sum += i;
    }
    System.out.println("실행 시간: " + (System.currentTimeMillis() - t) + " ms");
}

// 실행 시간 : 19 ms
```

동일 타입 연산

```
public static void main(String[] args) {
    long t = System.currentTimeMillis();
    long sum = 0L;
    for (long i = 0; i < 1000000; i++) {
        sum += i;
    }
    System.out.println("실행 시간: " + (System.currentTimeMillis() - t) + " ms") ;
}

// 실행 시간 : 4 ms
```

100만건 기준으로 약 5배의 성능 차이가 난다. 따라서 서비스를 개발하면서 불필요한 오토 캐스팅이 일어나는지 확인하는 습관을 가지자.

[참고 사항]

- 링크
- 링크