📖 Java Study Skill Up Team Review

객체 지향 프로그래밍

데이터 타입 분류

- 자바의 데이터 타입은 크게 기본 타입(primitive type[프리미티브 타입])과 참조 타입(reference type[레퍼런스 타입])으로 분류된다.
- 참조 타입 : 객체(object)의 번지를 참조하는 타입으로 배열, 열거, 클래스, 인터페이스 타입이 있다.

```
객체(object)란?
데이터와 메서드로 구성된 덩어리라고 생각하면 된다.
```

[기본 타입]

1. 정수 : byte, char, short, int, long
2. 실수 : float, double
3. 논리 : boolean

[참조 타입]

1. 배열
2. 열거
3. 클래스
4. 인터페이스

```
기본 타입으로 선언된 변수와 참조 타입으로 선언된 변수의 차이점은 '저장되는 값'이다.

기본 타입으로 선언된 변수는 값 자체를 저장하고 있지만,
참조 타입으로 선언된 변수는 객체가 생성된 메모리 번지를 저장한다.
```

[스택(stack) 영역과 힙(heap) 영역]

```java
// 기본 타입 변수
int age = 11;
double price = 100.5;

// 참조 타입 변수
String name = "비프";
String hobby = "공부";
```

- 변수를 생성하면 이 변수들은 모두 스택이라는 메모리 영역에 생성된다.
- 기본 타입 변수인 age와 price는 직접 값을 저장하고 있다.

- 참조 타입 변수인 name과 hobby는 힙 메모리 영역의 String 객체 번지를 저장하고 이 번지를 통해 String 객체를 참조한다.

메모리 사용 영역

- java 명령어로 JVM이 구동되면 JVM은 운영체제에서 할당받은 메모리 영역을 다음과 같이 구분해서 사용한다.

[메서드 영역]

- 바이트코드 파일을 읽은 내용이 저장되는 영역으로 클래스별 상수, 정적 필드, 메서드 코드, 생성자 코드 등이 저장된다.

[힙 영역]

- 객체가 생성되는 영역
- 객체의 번지는 메서드 영역과 스택 영역의 상수와 변수에서 참조할 수 있다.

[스택 영역]

- 메서드를 호출할 때마다 생성되는 프레임이 저장되는 영역이다.
- 메서드 호출이 끝나면 프레임은 자동 제거된다.
- 프레임 내부에는 로컬 변수 스택이 있다. 여기에서 기본 타입 타입 변수와 참조 타입 변수가 생성되고 제거된다.

개인 시나리오

```
바둑판과 바둑돌이 있다.
바둑판 그리고 흰 돌과 검은 돌은 다음과 같이 정의한다.

바둑판  : 힙(heap) 영역으로 검은 돌 변수의 실제 데이터만 저장된다.
흰 돌   : 기본 타입 변수
검은 돌 : 참조 타입 변수
바둑판 밖 : 스택(stack)

그리고 바둑판 각각의 자리가 존재하는데, 이 자리가 번지 주소를 가지는 공간이라고 정의한
다.
동시에 이 번지 주소를 가지는 공간에는 검은 돌 변수의 데이터만 저장할 수 있다.

1. 변수 생성
int age = 11;
double price = 100.5;

String name = "비프";
String hobby = "공부";

2. 스택 영역과 힙 영역에 생성
age, price 기본 타입 변수를 생성하면 흰 돌 2개가 바둑판 밖 스택(stack) 영역에 생성된다.
name, hobby 참조 타입 변수를 생성하면 검은 돌 2개가 바둑판 옆 스택(stack) 영역에 생성되
고, 데이터는 바둑판 위 힙(heap) 영역 자리에 저장된다.

즉,
```

```
기본 타입 변수는 11, 100.5 와 같은 값을 직접 보유하고 스택 영역에 배치된다.
참조 타입 변수도 스택 영역에 배치되지만 실제 데이터("비프", "공부")가 저장되어 있는 메모리 주소를 보유한다.

이 메모리 주소는 힙(heap) 주소를 가리키며 "비프", "공부"가 힙에 저장된 실제 데이터를 말한다.

3. 결론
원시 유형(흰 돌)은 해당 값과 함께 스택에만 존재한다.
참조 유형(검은 돌)은 스택(메모리 주소 참조)과 힙(실제 데이터) 모두에 존재한다.
```

참조 타입 변수의 ==, != 연산

```java
package com.chapter.ch05.example.ex01;

public class referenceType {
    public static void main(String[] args) {
        int[] arr1;
        int[] arr2;
        int[] arr3;

        arr1 = new int[] {1 ,2, 3};
        arr2 = new int[] {1, 2, 3};
        arr3 = arr2;

        System.out.println(arr1 == arr2);   // false: 같은 주소를 참조하지 않음
        System.out.println(arr2 == arr3);   // true: 같은 주소를 참조함
    }
}
```

null과 NullPointerException

[null]

- 참조 타입 변수는 아직 번지를 저장하고 있지 않다는 뜻으로 null 값을 가질 수 있다.
- 즉 null도 초기값으로 사용할 수 있기 때문에 null로 초기화된 참조 변수는 스택 영역에 생성된다.
- 스택 영역에 생성된다는 말만 있다면 번지를 가지지는 않는 것이라고 추측할 수 있다.

```java
String nullTest = null;

System.out.println(nullTest == null);   // true
```

[NullPointerException]

- 참조 변수 사용 시 가장 많이 발생하는 예외이다.
- 변수가 null인 상태에서 객체의 데이터나 메서드를 사용하려 할 때 이 예외가 발생한다.

```java
package com.chapter.ch05.example.ex01;

public class nullPointerException {
    public static void main(String[] args) {
        int[] intArray = null;
        intArray[0] = 10;   // NullPointerException

        String str = null;
        System.out.println(str.length());   // NullPointerException
    }
}
```

- NullPointerException이 발생하면 예외가 발생된 곳에서 null인 상태의 참조 변수가 사용되고 있음을 알아야 한다.
- 즉 위에서 추측했던 것처럼 null인 상태의 참조 변수의 번지를 대입해야 한다는 말이다.
- 프로그램에서 객체를 사용하려면 해당 객체를 참조하는 변수를 이용해야 하는데, 변수에 null을 대입하면 번지를 잃게 되므로 더이상 객체를 사용할 수 없게 된다.
- 힙 메모리에는 있지만, 위치 정보를 모르기 때문에 사용할 수 없게되면 쓰레기 수집기(Garbage Collector)를 실행시켜 자동으로 제거한다.
- 이처럼 자바는 코드를 이용해서 객체를 직접 제거하는 방법을 제공하지 않는다.
- 객체를 제거하는 유일한 방법은 객체의 모든 참조를 없애는 것이다.

```java
String hobby = "공부";
hobby = null; // 힙 영역에 "공부"에 해당하는 String 객체는 쓰레기 객체가 된다.

String hobby = "공부";
String hobby2 = hobby;
hobby = null;  // 여기서 "공부"에 해당하는 String 객체는 hobby2가 여전히 참조하기 때문에 쓰레기 객체가 되지 않는다.
```

문자 추출

```java
package com.chapter.ch05.example.ex01;

public class charAtExample {
    public static void main(String[] args) {
        String subject = "자바 프로그래밍";
        char charValue = subject.charAt(0);
        char charValue2 = subject.charAt(2);

        System.out.println(charValue);  // 자
        System.out.println(charValue2); // 공백
```

```
    }
}
```

- 문자열은 인덱스를 매길 수 있다.
- 따라서 charAt(0)은 0번 인덱스 위치가 있는 '자'가 해당된다.
- 2번 인덱스는 공백으로 띄어쓰기를 포함하고 있으니 마지막 인덱스는 7번이 된다.

문자열 길이

```
package com.chapter.ch05.example.ex01;

public class length {
    public static void main(String[] args) {
        String subject = "자바 프로그래밍";
        System.out.println(subject.length());    // 8
    }
}
```

- 문자 추출에서 언급했듯 공백을 포함하기 때문에 length는 8이 된다.

문자열 대체

```
package com.chapter.ch05.example.ex01;

public class replace {
    public static void main(String[] args) {
        String str = "자바 프로그래밍";
        System.out.println(str.replace("자바", "Java"));    // Java 프로그래밍
    }
}
```

- 특정 문자열을 다른 문자열로 대체하고 싶다면 replace() 메서드를 사용할 수 있다.