hashcode()와 equals() Method?

1. 객체 비교(equals()와 ==)

① Object 클래스의 equals() : 이 메소드는 비교 연사자인 == 과 동일한 결과를 리턴한다. 오로지 참조값(객체의 주소값)이 같은지, 다시 말하면 동일 객체인지를 확인하는 기능이다.

② 자바에서는 두 객체를 동등 비교할 때 equals() 메소드를 흔히 사용한다. equals() 메소드는 두 객체를 비교해서 논리적으로 동등하면 true를 리턴하고 그렇지 않으면 false를 리턴한다.

▶ 논리적으로 동등하다는 것은 둘의 참조값이 다르더라도 객체 내부 value는 같다는 것을 의미한다. 이 equals함수를 재정의한 대표적인 예가 String class이다. String class는 equals() 메소드를 재정의해서 번지 비교가 아닌 문자열 '값'을 비교한다.

- 동일성 비교는 == 비교다. 객체 인스턴스의 주소 값을 비교한다.
- primitive data type의 경우 ==를 통해 값 비교가 가능하다.
- 동등성 비교는 equals() 메소드를 사용해서 객체 내부의 값을 비교한다.

▶ 참고로 primitive 타입이 == 비교를 통해 값 비교가 가능한 이유는 아래와 같다.

변수 선언부는 Java Runtime Data Area의 Stack 영역에 저장이 되고, 해당 변수에 저장된 상수는 Runtime Constant Pool에 저장되어있다. Stack의 변수 선언부는 해당 Runtime Constant Pool의 주소값을 가지게 되고, 만약 다른 변수도 같은 상수를 저장하고 있다면, 이 다른 변수도 같은 Runtime Constant Pool의 주소값을 가지게 때문에 엄밀히 말하면 primitive type 역시 주소값 비교가 되는 것이다.

2. Java hash code란?

- 객체 해시코드란 객체를 식별하는 하나의 정수값을 말한다. Object의 hashCode() 메소드는 객체의 메모리 번지를 이용해서 해시코드를 만들어 리턴하기 때문에 객체 마다 다른 값을 가지고 있다. 객체의 값을 동등성 비교 시 hashCode()를 오버라이딩 할 필요성이 있는데, 컬렉션 프레임워크에서 HashSet, HashMap, HashTable은 다음과 같은 방법으로 두 객체가 동등한지 비교한다.

① hashCode() : 메소드를 실행해서 리턴 된 해시코드 값이 같은지를 본다. 해시 코드값이 다르면 다른 객체로 판단하고, 해시 코드값이 같으면 다음으로,

② equals() 메소드로 다시 비교한다. 이 두 개가 모두 맞아야 동등 객체로 판단한다. 즉, 해시코드 값이 다른 엔트리끼리는 동치성 비교를 시도조차 하지 않는다.

3. equals()와 hashcode()를 같이 재정의해야 하는 이유

만약 equals()와 hashcode() 중 하나만 재정의 하면 어떻게 될까? 위 예에서도 봤듯이 hashcode()를 재

정의 하지 않으면 같은 값 객체라도 해시값이 다를 수 있다. 따라서 HashTable에서 해당 객체가 저장된 버킷을 찾을 수 없다.

반대로 equals()를 재정의하지 않으면 hashcode()가 만든 해시값을 이용해 객체가 저장된 버킷을 찾을 수는 있지만 해당 객체가 자신과 같은 객체인지 값을 비교할 수 없기 때문에 null을 리턴하게 된다. 따라서 역시 원하는 객체를 찾을 수 없다.

이러한 이유로 객체의 정확한 동등 비교를 위해서는 (특히 Hash 관련 컬렉션 프레임워크를 사용할때!) Object의 equals() 메소드만 재정의하지 말고 hashCode() 메소드도 재정의해서 논리적 동등 객체일 경우 동일한 해시코드가 리턴되도록 해야한다.

▶ 예제 코드

```
package ch22;

import java.util.Objects;

class Person{
    String name;
    int age;

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Person person)) return false;
        return Objects.equals(name, person.name);
    }

    @Override
    public int hashCode() {
        return Objects.hash(name);
    }

    public Person(String name, int age){
        this.name = name;
        this.age = age;
    }
```

```
}


public class EqualTest {
	public static void main(String[] args) {
		int a = 3;
		int b = 3;
		if(a == b){
			System.out.println("a와 b는 같다.");
		} else {
			System.out.println("a와 b는 같지 않다.");
		}

		String str1 = "홍길동";
		String str2 = "홍길동";
		if(str1 == str2){
			System.out.println("str1과 str2는 같다.");
		} else {
			System.out.println("str1과 str2는 같지 않다.");
		}

		String str3 = new String("홍길동");
		String str4 = new String("홍길동");
		if(str3 == str4){
			System.out.println("str3과 str4는 같다.");
		} else {
			System.out.println("str3과 str4는 같지 않다.");
		}

		String str5 = new String("홍길동");
		String str6 = new String("홍길동");
		if(str5.equals(str6)){
			System.out.println("str5과 str6은 같다.");
		} else {
			System.out.println("str5과 str6은 같지 않다.");
		}
```

```
        Person person1 = new Person("홍길동",10);
        Person person2 = new Person("홍길동", 20);
        if(person1 == person2){
            System.out.println("person1과 person2은 같다.");
        } else {
            System.out.println("person1과 person2은 같지 않다.");
        }

        if(person1.equals(person2)){
            System.out.println("person1과 person2은 같다.");
        } else {
            System.out.println("person1과 person2은 같지 않다.");
        }

        if(person1.name.equals(person2.name)){
            System.out.println("person1.name과 person2.name은 같다.");
        } else {
            System.out.println("person1.name과 person2.name은 같지 않다.");
        }
    }
}
```