== , equals() , hashcode()

작성자: 천수영

이것은 무엇일까요?

열매 중 먹을 수 있는 부분이 약 80%인데,
수분이 85~88%, 열량은 약 50kcal이다.
알칼리성 식품으로서 주성분은 탄수화물이고
당분(과당 및 자당) 10~13%,
사과산 · 주석산 · 시트르산 등의
유기산, 비타민 B와 C, 섬유소·지방 등이 들어 있다.

수정 전

```
function addTab2(id, title, url){

  let items = $('#jqxTabs').jqxTabs('length');
  //console.log("items",items);
  let flag = false;
  let selectNo = -1;
  for(let i=0;i<items;i++){
    let text = $('#jqxTabs').jqxTabs('getTitleAt', i);
    if(text==title){
      selectNo = i;
      flag = true;
      break;
    }
  }

  if(flag){
    $('#jqxTabs').jqxTabs('val', selectNo);
  }else{
    let content = '<iframe id="'+id+'" frameborder="0"  src="'+url+'"
    $('#jqxTabs').jqxTabs('addLast', title, content);
    $('#jqxTabs').jqxTabs('ensureVisible', -1);
  }
}
```

수정 후

```
function addTab( id , requestTabName , url ){

  let matchedTabNo = null;

  let existTabsLength = $('#jqxTabs').jqxTabs('length');
  for ( let i = 0 ; i < existTabsLength ; i++ ){
    let existTabName = $('#jqxTabs').jqxTabs('getTitleAt', i);
    if( requestTabName ==  existTabName ){
      matchedTabNo = i ;
    }
  }

  if(matchedTabNo !== null){
    $('#jqxTabs').jqxTabs('val', matchedTabNo );
  } else {
    let content = '<iframe id="'+id+'" frameborder="0"  src="'+ur
    $('#jqxTabs').jqxTabs('addLast', requestTabName , content);
  }
}
```

Java에서 비교하는 방법.

- ==
  : 원시 자료형의 데이터를 비교할 때

- equal()
  : 객체끼리 비교할 때.

- hashcode()
  : hash를 사용하는 Collection에서 객체를 비교할 때.
  (예: HashMap , HashSet)

== 비교: 스텍영역의 값끼리 비교한다.

equals비교: 힙영역의 값끼리 비교한다.

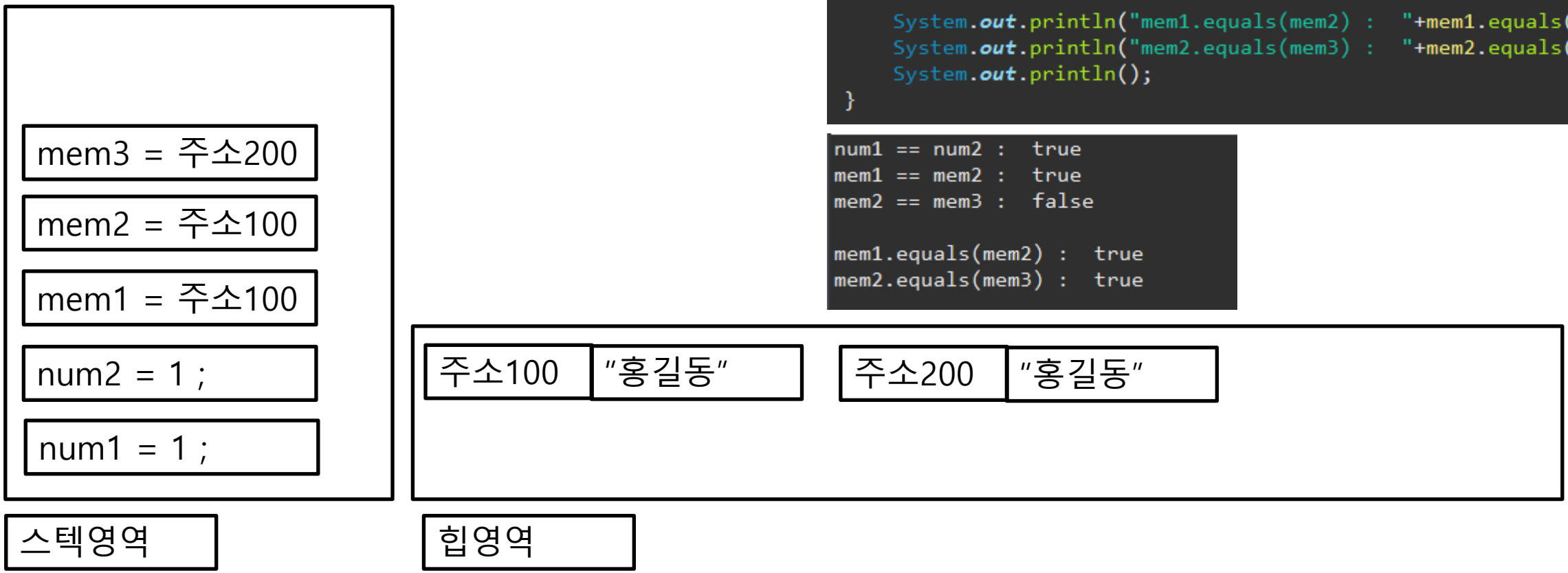
```java
public static void main(String[] args) {
    int num1 = 1;
    int num2 = 1;
    String mem1 = "홍길동";
    String mem2 = "홍길동";
    String mem3 = new String("홍길동");


    System.out.println("num1 == num2 :  "+(num1 == num2));
    System.out.println("mem1 == mem2 :  "+(mem1 == mem2));
    System.out.println("mem2 == mem3 :  "+(mem2 == mem3));
    System.out.println();


    System.out.println("mem1.equals(mem2) :  "+mem1.equals(mem2));
    System.out.println("mem2.equals(mem3) :  "+mem2.equals(mem3));
    System.out.println();
}
```

```
num1 == num2 :  true
mem1 == mem2 :  true
mem2 == mem3 :  false

mem1.equals(mem2) :  true
mem2.equals(mem3) :  true
```



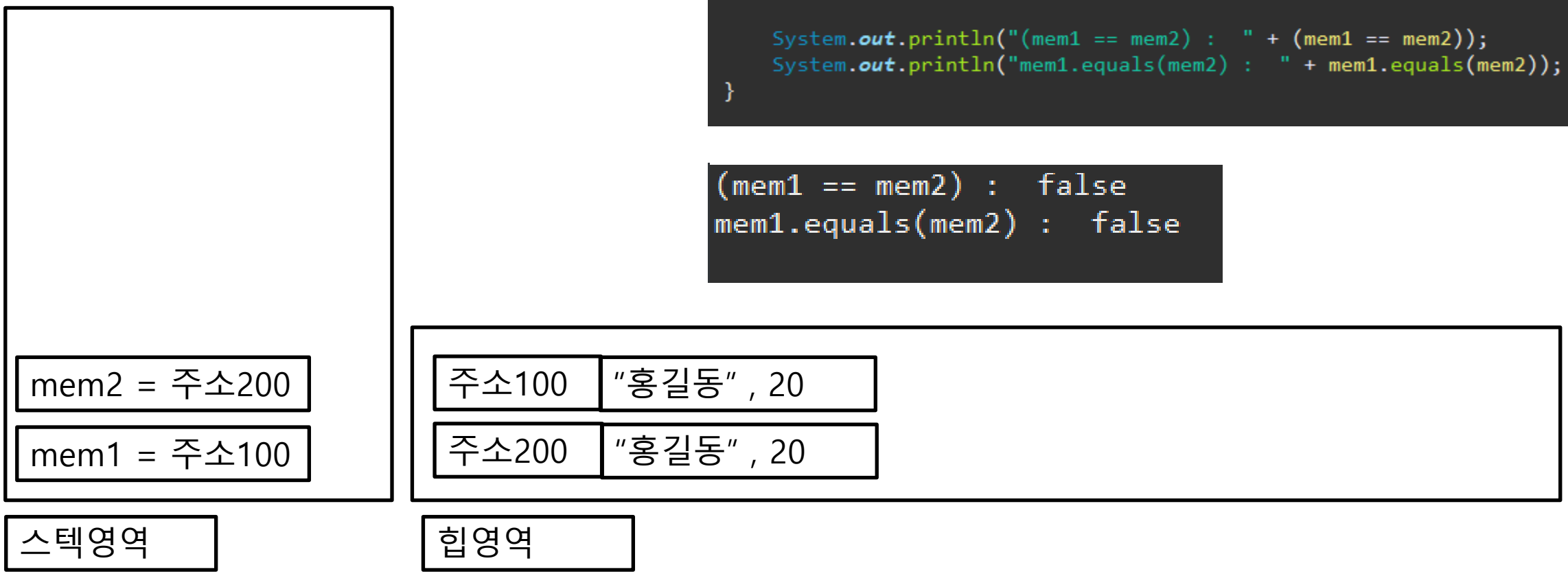
Member객체를 equal함수로 비교하기

```java
public class Member {
    private String name;
    private int age;
```

```java
public static void main(String[] args) {
    Member mem1 = new Member("홍길동",20);
    Member mem2 = new Member("홍길동",20);


    System.out.println("(mem1 == mem2) :  " + (mem1 == mem2));
    System.out.println("mem1.equals(mem2) :  " + mem1.equals(mem2));
}
```

```
(mem1 == mem2) :  false
mem1.equals(mem2) :  false
```



String클래스의 equals()

```
public boolean equals(Object anObject) {
if (this == anObject) {
    return true;
}
if (anObject instanceof String) {
    String anotherString = (String)anObject;
    int n = count;
    if (n == anotherString.count) {
    char v1[] = value;
    char v2[] = anotherString.value;
    int i = offset;
    int j = anotherString.offset;
    while (n-- != 0) {
        if (v1[i++] != v2[j++])
        return false;
    }
    return true;
    }
}
return false;
}
```

Member클래스의 equals()

```
public boolean equals(Object obj) {
return (this == obj);
}

/**
 * Creates and returns a copy of this object.  The precise meaning
 * of "copy" may depend on the class of the object. The general
 * intent is that, for any object <tt>x</tt>, the expression:
 * <blockquote>
```

Member클래스의 equals()를 override 후

```java
@Override
public boolean equals(Object obj) {
    // TODO Auto-generated method stub
    if (obj instanceof Member) {
        Member mem = (Member) obj;
        return (this.age == mem.age) && (this.name.equals(mem.name));
    }else {
        return false;
    }
}
```

```java
public static void main(String[] args) {
    Member mem1 = new Member("홍길동",20);
    Member mem2 = new Member("홍길동",20);


    System.out.println("(mem1 == mem2) :  " + (mem1 == mem2));
    System.out.println("mem1.equals(mem2) :  " + mem1.equals(mem2));
}
```

```
(mem1 == mem2) :  false
mem1.equals(mem2) :  true
```

Member클래스의 hashCode()로 비교하기

```java
public static void main(String[] args) {
    Member mem1 = new Member("홍길동",20);
    Member mem2 = new Member("홍길동",20);


    System.out.println("(mem1 == mem2) :   " + (mem1 == mem2));
    System.out.println("mem1.equals(mem2) :   " + mem1.equals(mem2));
    System.out.println();

    System.out.println("mem1.hashCode() :   " + mem1.hashCode());
    System.out.println("mem2.hashCode() :   " + mem2.hashCode());

}
```

```
(mem1 == mem2) :   false
mem1.equals(mem2) :   true

mem1.hashCode() :   1626635253
mem2.hashCode() :   1391870861
```

hashCode()값이 다르면 발생하는 문제

```
public static void main(String[] args) {
    Member mem1 = new Member("홍길동",20);
    Member mem2 = new Member("홍길동",20);
```

```
    HashMap memMap = new HashMap< Member , Integer>();
    memMap.put(mem1, 1);
    memMap.put(mem2, 1);
    System.out.println("memMap.size() :  " + memMap.size());

}
```

```
memMap.size() :  2
```

Hash 타입의 Collection은
Key값의 중복을 허용하지 않는데 .

Key값이 중복으로 적용되었다.

String클래스의 hashCode()

```
public int hashCode() {
int h = hash;
    int len = count;
if (h == 0 && len > 0) {
    int off = offset;
    char val[] = value;

        for (int i = 0; i < len; i++) {
            h = 31*h + val[off++];
        }
        hash = h;
    }
    return h;
}
```

Member클래스의 hashCode()

```
    @Override
    public int hashCode() {
        // TODO Auto-generated method stub
        return super.hashCode();
    }
```

```
public native int hashCode();

/**
 * Indicates whether some other object is "equal to" this one.
 * <p>
 * The <code>equals</code> method implements an equivalence relation
 * on non-null object references:
 * <ul>
 * <li>It is <i>reflexive</i>: for any non-null reference value
 *     <code>x</code>, <code>x.equals(x)</code> should return
 *     <code>true</code>.
 * <li>It is <i>symmetric</i>: for any non-null reference values
```

Member클래스의 hashCode()를 override 후

```java
@Override
public int hashCode() {
    // TODO Auto-generated method stub
    return age + this.name.hashCode();
}
```

```java
public static void main(String[] args) {
    Member mem1 = new Member("홍길동",20);
    Member mem2 = new Member("홍길동",20);


    System.out.println("(mem1 == mem2) :  " + (mem1 == mem2));
    System.out.println("mem1.equals(mem2) :  " + mem1.equals(mem2));
    System.out.println();

    System.out.println("mem1.hashCode() :  " + mem1.hashCode());
    System.out.println("mem2.hashCode() :  " + mem2.hashCode());
    System.out.println();


    HashMap memMap = new HashMap< Member , Integer>();
    memMap.put(mem1, 1);
    memMap.put(mem2, 1);
    System.out.println("memMap.size() :  " + memMap.size());

}
```

```
(mem1 == mem2) :  false
mem1.equals(mem2) :  true

mem1.hashCode() :  54150082
mem2.hashCode() :  54150082

memMap.size() :  1
```

HashMap에서 put()메서드 사용시 hashCode()를 사용하는 이유?

```java
public V put(K key, V value) {
    if (key == null)
        return putForNullKey(value);
    int hash = hash(key.hashCode());
    int i = indexFor(hash, table.length);
    for (Entry<K,V> e = table[i]; e != null; e = e.next) {
        Object k;
        if (e.hash == hash && ((k = e.key) == key || key.equals(k))) {
            V oldValue = e.value;
            e.value = value;
            e.recordAccess(this);
            return oldValue;
        }
    }

    modCount++;
    addEntry(hash, key, value, i);
    return null;
}
```

hashCode()를 비교하고
equals()도 비교하는 이유?

hashCode()가 속도가 더 빠르기 때문이다.

equals 비교

```
public boolean equals(Object anObject) {
if (this == anObject) {
    return true;
}
if (anObject instanceof String) {
    String anotherString = (String)anObject;
    int n = count;
    if (n == anotherString.count) {
    char v1[] = value;
    char v2[] = anotherString.value;
    int i = offset;
    int j = anotherString.offset;
    while (n-- != 0) {
        if (v1[i++] != v2[j++])
        return false;
    }
    return true;
    }
}
return false;
}
```

Hash 비교

```
if (e.hash == hash
```

If (1235412 == 57846)

교훈

대량의 객체들을 비교할 때는
hashCode()를 사용하면 효율적이다.

모든 사람을 DNA검사해볼 필요는 없다.