



제15장

java.lang패키지-Object클래스

1. java.lang패키지

▣ java 프로그램의 기본적인 클래스들을 포함하고 있는 패키지를 말한다.

- 포함된 클래스나 인터페이스는 따로 import없이 사용 가능하다.

- 주요 클래스들은 아래와 같다.

클래스		용도
Object	중요	- 자바 클래스의 최상위 클래스로 사용
System		- 표준 입력장치(키보드)로부터 데이터를 입력 받을 때 사용 - 표준 출력장치(모니터)로 출력하기 위해 사용 - 자바 가상 머신을 종료시킬 때 사용 - 쓰레기 수집기를 실행 요청할 때 사용
Class		- 클래스를 메모리로 로딩할 때 사용
String	중요	- 문자열을 저장하고 여러가지 정보를 얻을 때 사용
StringBuffer, StringBuilder		- 문자열을 저장하고 내부 문자열을 조작할 때 사용
Math		- 수학 함수를 이용할 때 사용
Wrapper	Byte, Short, Character	- 기본 타입의 데이터를 갖는 객체를 만들 때 사용
	Integer, Float, Double	- 문자열을 기본 타입으로 변환할 때 사용
	Boolean	- 입력값 검사에 사용

2. Object클래스

▣ java의 **최고 조상 클래스**이다.

- 일반적으로 다른 클래스를 상속을 명시적으로 하지 않는다면, Object클래스를 상속한다.

(생략을 하더라도 컴파일 시에 extends Object가 붙는다.)

- Object클래스에는 11개의 메서드가 존재하며, 어떠한 클래스에서도 사용 가능하다.



compact1, compact2, compact3

java.lang

Class Object

java.lang.Object

```
public class Object
```

Class Object is the root of the class hierarchy. Every class has Object as a superclass.

Since:

JDK1.0

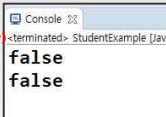
Modifier and Type	Method and Description
protected Object	clone() Creates and returns a copy of this object.
boolean	equals(Object obj) Indicates whether some other object is "equal" to this object.
protected void	finalize() Called by the garbage collector on an object when garbage collection determines that no more references to the object are in use.
Class<?>	getClass() Returns the runtime class of this Object.
int	hashCode() Returns a hash code value for the object.
void	notify() Wakes up a single thread that is waiting on this object.
void	notifyAll() Wakes up all threads that are waiting on this object.
String	toString() Returns a string representation of the object.
void	wait() Causes the current thread to wait until the specified object is notified by another thread.
void	wait(long timeout) Causes the current thread to wait until the specified object is notified by another thread, or the specified amount of time has elapsed.
void	wait(long timeout, int nanos) Causes the current thread to wait until the specified object is notified by another thread, or the specified amount of time has elapsed.

3. Object클래스의 메서드 - 1

▣ 객체의 주소 비교 메서드

- `public boolean equals(Object obj) { .. }`
- `equals`메서드는 원론적으로 `==`연산자와 동일한 결과를 리턴을 한다. 즉, 메모리 번지비교를 한다.

```
public static void main(String[] args) {  
    new연산자가 하는 일은? (수도 없이 언급했다)  
    Object object1 = new Object();  
    Object object2 = new Object();  
    //주소 비교  
    boolean result1 = object1.equals(object2);  
    boolean result2 = object1 == object2;  
    System.out.println(result1);  
    System.out.println(result2);  
}
```



```
Console  
<terminated> StudentExample [Java...]  
false  
false
```

3. Object클래스의 메서드 -2

■ 객체의 논리적 동등이란?

- 논리적 동등의 개념은 객체의 주소가 달라도 해당 클래스의 저장 값이 동일하다면, 일단 같은 객체로 보자는 개념이다.

- 하여, 사용자 정의 클래스에서는 논리적 동등을 만들기 위해서는 Object클래스의 equals()를 오버라이딩을 하여 값을 비교하게끔 만들어야 한다.

ex) String클래스의 equals메서드는 문자열 비교로 참, 거짓이 리턴하게끔 오버라이딩이 되어 있다.

```
public static void main(String[] args) {  
  
    String str1 = new String("홍길동");  
    String str2 = new String("홍길동");  
  
    boolean result = str1.equals(str2);  
    System.out.println(result);  
}
```

"홍길동"

"홍길동"

Console
<terminated> CarEx
true

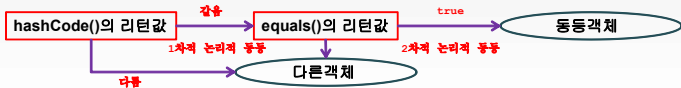
3. Object클래스의 메서드 -3

■ 객체의 해시코드(hashCode())

- 메모리 번지를 기준으로 하여, 객체를 식별할 수 있는 유일한 값을 의미한다. 물론 10진수이다.
- 따라서, 개별 객체(인스턴스)들은 다 해시코드가 다르다.

■ 논리적 동등 비교 시 hashCode() 오버라이딩이 필요한 이유

- 통상, hasing기법을 이용해서 저장하고 읽어오는 컬렉션 프레임워크인 HashSet, HashMap, HashTable, LinkedHashMap같은 클래스는 두 객체가 동등한지 판단을 내릴 때, **먼저 hashCode()의 리턴값을 보고 1차 결정 후, equals()의 리턴값으로 최종 결정을 지어 저장할지 말지를 결정한다.**
- 하여, 위와 같은 HashXXXX들은 논리적 동등 객체로 만들기 위해서는 hashCode()가 리턴하는 값이 같도록 오버라이딩을 해야 한다.



3. Object클래스의 메서드 -4

■ 객체의 문자정보를 알려주는 toString()

- 객체의 패키지명@16진수코드를 반환하는 메서드이다. 통상, 메모리 번지를 리턴한다고 생각하자.
메모리 번지를 리턴한다.

```
public static void main(String[] args) {  
  
    Object object = new Object();  
    System.out.println(object.toString());  
  
}
```

```
Console  
<terminated> CarExample [Java Application] C:\Program Files  
java.lang.Object@15db9742
```

- 메모리 번지와 같은 의미없는 데이터를 반환하는 것 보다는 의미가 있는 문자정보를 반환하도록 오버라이딩을 하면 된다.(ex. Date, String클래스)

```
Date date = new Date();  
System.out.println(date.toString());  
  
String str = new String("자바");  
System.out.println(str.toString());
```

```
Console  
<terminated> CarExample [Java Application] C:\Program Files\Java\W  
Tue Aug 27 14:16:45 KST 2019  
자바
```

3. Object클래스의 메서드 -5

■ 객체를 복제(clone())하는 메서드

— 원본 객체의 필드 값과 똑같은 값을 가지는 또 다른 객체를 생성하는 것이다.

■ 복제의 종류

— 얇은 복제(thin clone) : 멤버변수 값만 복사함, 참조변수들은 번지를 서로 공유한다.

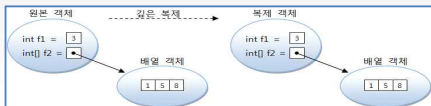
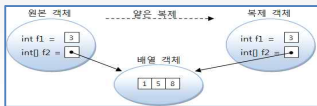
— 깊은 복제(deep clone) : 똑같은 객체로 만든다.(참조변수들의 번지 역시 서로 다르다.)

■ Object클래스의 clone()메서드는 얇은 복제를 한 객체를 리턴한다.

— 또한, java.lang.Cloneable인터페이스를 구현한 객체만 복제가 가능하다.

— 구현하지 않은 클래스를 복제 시도를 하면, CloneNotSupportedException이 발생한다.

— 하여, 참조변수들의 값은 프로그래머들이 직접 복제하는 코드를 필히 작성해줘야 한다.



3. Object클래스의 메서드 -6

- 객체를 소멸하는 `finalize()`는 될 수 있으면 사용하지 말도록 한다.
 - 기본적으로 쓰레기 객체를 GC가 소멸하기 직전에 `finalize()`를 실행시킨다.
 - 아울러, `finalize()`에는 아무런 내용이 기재되어 있지 않다.
 - 그럴 일은 없겠지만, 객체가 소멸할 때 실행할 코드가 있다면 `finalize()`를 오버라이딩하여 사용하면 될 것이다.

- 아울러 GC는 메모리에 있는 모든 쓰레기 객체들을 다 소멸하지는 않는다.
 - JDK1.8부터는 거의 다 소멸시키는 것으로 보인다.
 - GC의 구동시점으로 애매한 말밖에 없다.
 - ex) 메모리가 부족하다, CPU가 한가하다 등

감사합니다.

