Java.lang - Object I

Object Class

- **모든 Class의 최상위 Class**
- **모든 Class는 Object Class를 상속받는다.**

---

finalize( ) Method - Garbage Collector의 동작원리

- **protected void finalize() throws Throwable**
- Object에 선언된 멤버이므로 모든 인스턴스에 이 메소드가 존재한다.
- **Garbage Collector (JVM)에 의해 인스턴스가 완전히 소멸되기 직전에 호출되는 메소드**

```
public class MyName {
	String objName;

	public MyName(String name){ objName = name; }

	@Override
	protected void finalize() throws Throwable{
		super.finalize();
		System.out.println(objName+"이 완전히 소멸되었습니다.");
	}
}

public class InstFinalize {
	public static void main(String[] args) {
		MyName obj1 = new MyName("인스턴스1");
		MyName obj2 = new MyName("인스턴스2");

		// null 값이 들어가는 순간 Garbage Collector의 제거대상이 된다.
		obj1 = null;
		obj2 = null;

		System.out.println("프로그램을 종료합니다.");
		// System.gc();
		// System.runFinalization();
	}
}
```

- **인스턴스 종료시점에서 할 일들을 구현하고 싶다. 하지만 위의 예제에서는 finalize() Method가 호출되지 않을 수 있다.**
  - **프로그램 종료 시, JVM은 사용한 메모리를 전부 소멸시키고 반납한다. (GC를 실행시키지 않을 수도 있다.)**
  - **finalize() Method는 Gabage Collection (JVM)에 의해 인스턴스의 완전소멸이 진행될 때에만 호출되는 메소드**

- **Garbage Collector는 제거대상인 인스턴스를 체크했더라도 바로 제거하지 않을 수 있다.**
  (JVM은 엄청 바쁘기 때문에, JVM이 여유가 있을 때 실행)
  - **System.gc() : Garbage Collection 실행 메소드**
  - **System.runFinalization() : GC에게 소멸이 결정된 인스턴스의 완전소멸을 명령하는 메소드**
  - **이 두 메소드가 실행되어야 finalize() 메소드의 실행을 볼 수 있다!**

Garbage Collection 추가정리

i. GC는 한 번도 발생하지 않을 수 있다.

ii. GC가 발생하면 소멸의 대상이 되는 인스턴스는 결정되지만, 이것이 소멸로 바로 이어지지 않는다.

iii. 인스턴스의 소멸이 이루어지지 않은 상태에서도 프로그램은 종료가능! 어짜피 종료되면서 메모리는 소멸되니까!

iv. 따라서, finalize() Method를 호출하려면
System.gc(), System.runFinalization() Method가 호출되어야 한다.
