객체지향 키워드

객체지향 : 캡슐화, 상속성, 다형성을 이용하여 코드 재사용 증가와 유지보수 감소등을 위한 장점을 얻기 위해서 객체를 연결시켜 프로그래밍 하는 것.

클래스 : 표현 대상에 대한 설계

오브젝트 : 클래스의 인스턴스

캡슐화 : 구현으로부터 인터페이스를 분리하는 것

상속성 : 하나의 클래스가 가진 특징을 물려받아 추상적인 것을 구체화

다형성 : 객체의 타입에 따라 다른 일을 할 수 있게

JVM

<https://medium.com/@lazysoul/jvm-%EC%9D%B4%EB%9E%80-c142b01571f2>

Java collection

<http://hackersstudy.tistory.com/26>

<http://withwani.tistory.com/150>

Serializable

<http://hyeonstorage.tistory.com/254>

Garbage collection

<http://www.jpstory.net/2013/12/garbage-collection-in-java/>

<http://d2.naver.com/helloworld/1329>

String, StringBuffer, StringBuilder

<http://ooz.co.kr/298>

static main

: static변수는 메모리에 올라가서 프로그램 종료까지 사라지지 않음

Main이 가장 먼저 실행되야하니까

String pool

<http://hyunc87.tistory.com/9>

Singleton

<https://blog.seotory.com/post/2016/03/java-singleton-pattern>

volatile, synchronization

<http://thswave.github.io/java/2015/03/08/java-volatile.html>

interface / abstract

즉,인터페이스는 타입을 재정의 하기위해서입니다.

클래스는 하나의 type 입니다. 인터페이스는 그 클래스가 가질수 있는 type을 오버로딩하기 위해서 만들어 진것입니다. 하나의 클래스가 여러 인터페이스를 구현하여 그 인터페이스 type으로 사용될수 있다는 것을 의미합니다.

반면 추상화 클래스는 상위 클래스에서 직접 구현을 하지 않고 하위 클래스에서 구현하게 함으로써 프로그램상에서 하위 클래스를 구현한 오브젝트가 각각 상위 클래스 타입으로 쓰일때 그 각각의 하위 오브젝트에 맞는 메소드를 동적으로 호출하기 위해서 사용합니다.

언듯 보면 매우 비슷하면서 그 사용도 같습니다. 하지만 분명한 점은 인터페이스를 구현하는 것은 이 클래스가 그 인터페이스 타입으로 사용될수 있다는 것을 컴파일러에게 알려주고 실행시간에 호출 되는 것이고 추상클래스는 실행시간에 동적으로 호출 된다는 것입니다.