1. 인스턴스 멤버

* 인스턴스 멤버란 객체를 생성한 후 사용할 수 있는 속성과 메소드

1. 정적 멤버

* 정적 멤버는 클래스 로드 후 객체생성 없이 바로 사용할 수 있는 속성과 메소드
* 정적 멤버의 선언은 static 키워드를 이용

1. 싱글톤

* 전체 프로그램에서 단 하나의 객체의 생성만 보장되는 경우
* 싱글톤 객체의 선언

static 클래스명 singleton = new 생성자();

* 싱글톤 객체의 사용

싱글톤으로 선언된 클래스명.getInstance();로 객체를 얻어서 사용

* 싱글톤 객체의 장점

객체가 한 번만 생성되기 때문에 메모리 낭비를 줄일 수 있음

* 싱글톤 객체의 단점

싱글톤 객체를 수정할 경우 싱글톤 객체를 사용하는 다른 클래스들에 영향을 줄 수 있어서 프로그램 수정이 복잡해짐

이렇게 하나의 객체가 여러 개의 클래스의 영향을 주는 경우를 의존도가 높다고 한다.

프로그래밍에서 의존도와 결합도가 낮은 프로그램을 좋은 프로그램이라고 한다.

1. final 필드

* final 키워드는 프로그램 실행 중 변경할 수 없는 필드(속성)을 선언할 때 사용

ex) final double PI = 3.14;

final int WEEKDAYS = 7;

* final 키워드로 변수 선언 시 초기화 필수

1. 상수

* 지구에 존재하는 불변 값들
* 원주율, 지구 둘레, 지구 반지름…..
* 상수들은 대부분 대문자로 선언하게 됨

ex) static final double PI = 3.14;

* 상수들은 불변 값이므로 final 키워드를 사용하여 선언

1. 패키지

* 폴더의 역할과 클래스 식별자의 역할
* 패키지는 상위 패키지와 하위 패키지로 구분됨
* .을 이용하여 포함관계를 나타냄

상위패키지.하위패키지.하위패키지2…..클래스

1. 접근 제한자

* 다른 객체에서 클래스나 속성, 메소드에 접근하는 것을 제한하기 위해 사용
* 보안의 목적으로 접근 제한자를 걸어서 데이터 유출방지ㅐ
* 접근 제한자의 종류

public : 어떤 클래스에서든 접근가능한 접근 제한자

protected : 같은 패키지내의 클래스나 자식 클래스에서만 접근 가능한 접근

제한자

private : 모든 접근이 막혀있는 접근 제한자

default : 같은 패키지내의 클래스에서만 접근 가능한 접근 제한자

class 클래스명 {  
 default 접근 제한자 적용된 상태

}

* 클래스는 private, protected 선언할 수 없음
* private, protected 클래스내의 필드(속성), 메소드 선언에 사용

1. getter/setter 메소드

* 객체지향언어에서는 객체의 필드 값을 외부에서 수정하는 것을 막음
* 대부분 필드의 선언 private 접근 제한자를 이용하여 선언
* private 접근 제한자로 선언된 필드의 값을 가져오거나 변경할 때 getter/setter 메소드 사용

1. 상속

* 상속은 클래스들간의 부모, 자식관계 정의
* 자식클래스에서 부모클래스를 상속받아 부모클래스에 있는 필드, 메소드를 직접 사용 가능
* 상속을 사용하는 이유

클래스간의 중복코드 제거하여 코드 복잡도 낮춤