-static : 객체 생성 없이 사용 가능.  
 => 자바에서 미리 해당 내용만 가져다가 생성.  
 장/단점 : 어디서 불러다 사용하여도 동일한 내용을 가져옴.  
 (공유변수, shared, 공공재)  
 바꾸면 모두 바뀜.

<변수>  
 접근권한 static 타입 변수명 = 값;  
 ex) public static int abc = 123;  
 <메소드>  
 접근권한 static 반환타입 메소드명(인자타입 인자명, …) {  
 내용  
 }  
 ex) public static void main(String[] args) {}

-Static 선언된 변수/메소드 사용방법  
 변수 - 클래스명.변수명  
 메소드 - 클래스명.메소드명(~~~)  
 <김철수 구분>  
 서울.김철수 -> 서울에 사는 김철수  
 부산.김철수 -> 부산에 사는 김철수

-Java에서의 main 메소드 : 해당 프로그램을 실행할 때 해당 main 메소드를 자동 실행한다. 따라서 무조건적으로 main이 존재하여야 한다.

- final : 상태를 고정시킨다. 단독으로 쓰이는 경우 드뭄. 보통 static과 final을 같이 씀.  
 변수 - 값을 고정. 암호화 키(값이 바뀌면 해석 X)  
 메소드 - 오버라이딩 불가. 위변조 방지(배포용 소스)  
 주로 변수에서 static final을 같이 써서 ‘상수화’할 때 사용.(파일 업로드 경로, 필요한 설정 값…)

변수 - 지역변수 - **객체생성 필요**. 변경 가능  
 ㄴ 전역변수 - (일반)전역변수 - **객체생성 필요**. 변경 가능  
 ㄴ 정적변수(static) - **객체생성 필요 없음**. 변경 가능  
 ㄴ 상수(static final) - **객체 생성 필요 없음.** 변경 불가

-디자인 패턴 : 개발의 효율적인 형태를 취하기 위한 구현방식.  
 \*필수가 아닌 선택(권장 사항)

-Singleton 패턴 : 객체의 재사용성을 높여 자원효율 및 관리를 극대화시킨 디자인 패턴.  
 객체 생성을 한 번만 해서 쓰자.(객체 호출 시점부터 프로그램 끝까지)  
 빈번하게 사용되는 객체를 따로 만들어서 호출.  
 객체의 주소값을 변수에 담아서 사용하기 때문에 상태값이 유지됨  
 (static은 아니지만 static처럼 사용 가능)  
  
