1. 싱글톤
 - 클래스의 인스턴스가 하나만 되도록 구현하는 디자인 패턴의 한 종류.

2. 구현 방법
 - new 연산자로 외부에서 생성자를 호출할 수 없도록 해야한다.
   -> 생성자 앞에 private 접근 제한자를 사용해준다.
 - 자신의 클래스 타입인 정적 필드를 하나 선언하고 자신의 객체를 생성해 초기화한다.
 - 외부에서 호출할 수 있도록 정적메소드를 선언하고 정적필드에서 참조하고 있는 자신의 객체를 리턴해준다.

```
public class SingletonExample{
    //정적 필드
    private static singletonExample = new SingletonExample();
    // 생성자
    private SingletonExample(){}
    //정적 메소드
    static SingletonExample getSingletonExample(){
        return singletonExample;
    }
}
```

```
public class Main{
 public static void main(){
  // 두 인스턴스는 같은 객체이다.
  SingletonExample singletonExample1SingletonExample.getSingletonExample();
  SingletonExample singletonExample2SingletonExample.getSingletonExample();
 }
}
```

3. 싱글톤 패턴의 장점
 - 고정된 메모리 영역만을 사용하여 하나의 인스턴스를 사용하기 때문에 메모리 낭비를 방지한다.
 - 싱글톤으로 만들어진 클래스의 인스턴스는 전역이기 때문에 다른 곳에서 데이터를 공유하기 쉽다.
 - 두 번재 이용시부터 객체 로딩 시간이 줄어든다.

4. 싱글톤 패턴의 단점
 - 싱글톤 인스턴스가 많은 일을 해아할 경우나 많은 데이터 공유가 필요할 때 다른 클래스의 인스턴스들 간의 결합도가 높아져 "개방-폐쇠 원칙"을 위배하게 된다.
   -> 유지보수 및 비용이 높아질 수 있다, 멀티 스레드 환경에서 동기화 처리를 안하면 인스턴스가 2개가 생길 수 있다.