애너테이션만들기고 사용해보기

```
@Retention( RetentionPolicy.RUNTIME)
public @interface Count {
        int value() default 1;
}

public class MyEx {
        @Count(value=3)
        private int apples;
        @Count(5)
        private int bananas;
        public MyEx(int apples, int bananas) {
                super();  // 부모의 기본생성자를 호출합니다. 이 부분이 없어도 부모의 기본생성자를 호출하는 코드가 생성됩니다
                this.apples = apples;
                this.bananas = bananas;
        }
}
```

```
public class ValidationUtils {
    public static void main(String[] args) throws IllegalAccessException {
        MyEx  myex = new MyEx(3,2);
        validateFields( myex);
    }

    public static void validateFields(Object obj) throws IllegalAccessException {
        Class  clazz = obj.getClass();
        Field[] fields = clazz.getDeclaredFields();

        for (Field field : fields) {
          //field에 대한 애너테이션정보 얻어오기
            Count annotation = field.getAnnotation(Count.class);

            if (annotation != null) {
                field.setAccessible(true);  // private 이어도 접근가능하도록 한다
                Object value = field.get(obj);  //객체정보 제공 => 필드의 값얻어오기

                if (value != null && value instanceof Integer) {
                    int expectedValue = annotation.value();   //애너테이션에 설정한 값
                    int fieldValue = (int) value;             //실제 변수값

                    if (fieldValue != expectedValue) {
                        throw new IllegalArgumentException(
                                field.getName() + "항목은  "+ expectedValue+ " 값이어야 해 !! " );
                    }
                }
            }
        }
    }
}
```