T4 테스팅 보고서

1. 1st Testing 요청서 결과

A. 프로그램

i. 회전, 반전, 리사이즈시에 숫자 이외의 문자입력시 예외처리추가됨.

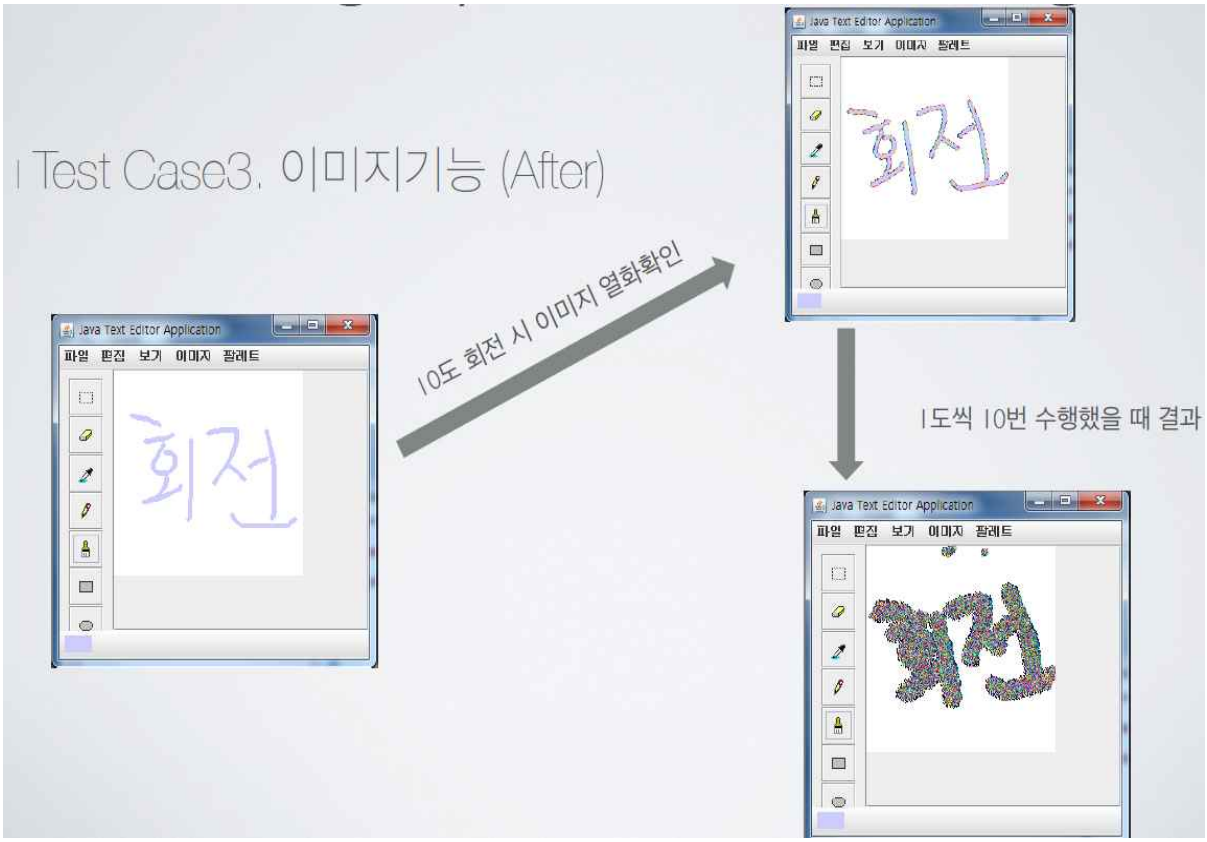
B. Usecase별 요구사항 결과

Usecase 1.1 파일을 새로 만든다.	
i. 저장 후 "예"클릭, 저장 이후 새 파일로 변하지 않음. ii. 아무것도 수정되지 않은 그림에서 새 파일을 클릭 시, 저장 창이 뜸	수정완료
Usecase 1.2 불러오기	
i. 이미 저장을 한 파일에서도 다시 저장하기를 물어봄 ii. 불러오기 수행 후 이미지가 돌아가는 현상 발생	수정완료
Usecase 1.4 종료	
i. 종료 확인 창 이후, 저장에 대해 물어보지 않고 저장 창이 뜸 ii. 아무것도 수정되지 않은 그림에서도 종료 시 저장 창이 뜸	수정완료
Usecase 2.2 축소	
i. 이미지가 줄어들면 이미지 크기에 맞게 스크롤이 이동해야 하는데, 이것이 변경되지 않음.	수정완료
Usecase 3.1 이미지 객체 복사	
i. 이후 붙여넣기(Usecase 3.3) 시 이미지가 반투명하게 붙음	수정완료
Usecase 3.2 이미지 객체 잘라내기	
i. 가끔 전체 영역이 사라짐	수정완료
Usecase 3.3 이미지 객체 붙여넣기	
i. 새 파일에서 그냥 붙여넣기 하면 알 수 없는 영역이 붙여넣어짐	수정완료
Usecase 4.1 이미지 회전	
i. 아무런 작동 안 함.	오류발생
Usecase 4.2 이미지 반전	
i. 반전이 제대로 되지 않는 오류.	수정완료
Usecase 5.1 이미지 사이즈 조절	
i. 정사각형이 아니면 width, height가 변경되는 경우가 발생 ii. 정사각형은 배경이 흰색이지만 이외에는 검정색으로 됨 iii. 정사각형일때만 제대로 작동	수정완료
Usecase 8.1 이미지 이동, Usecase 8.2 이미지 선택, Usecase8.3 이미지 삭제	
i. 작동불능 ii. 범위를 사용자가 알 수 없고, 마찬가지로 일정확률로 작동하는 경우가 발생.	수정완료

Usecase 10.1 연필	
i. 이전 클릭 포인트로부터 직선으로 그려짐 ii. 새 파일 이후에도, 이전 포인트가 초기화가 되지 않음	수정완료

2. 새롭게 발견된 오류

A. 이미지회전 : 이미지의 회전은 정상적으로 작동하나 회전시마다 픽셀이 깨지는 현상이 일어남.



3. Sonar 보고서

A. 코드중복문제 : 비슷한 기능들중에 중복이 많이 발생. 하나의 도구에 하나의 클래스를 사용하기 보다는 비슷한 기능을 하는것끼리 상속관계를 만들어서 코드의 중복을 줄이는 것이 더 효과적일것이다.

B. 비어있는 IF statement 구문 : IF statement에서 공백인 부분이 총15개 발견. 사용되지 않는 IF문은 없애고 아직 구현하지 않은 부분은 추가.

Package	Class	Line
ASAP.Tools	CircleTool	18, 23
	MoveTool	20, 23
	PipetteTool	22, 26
	RectangleTool	20, 30
	SelectTool	20, 23
	BrushTool	42
	EraseTool	64
	PenTool	49
ASAP.Canvas	MYCanvas	132, 136

C. 비어있는 finally : 예외 발생 여부와 관계없이 무조건 실행되므로 비어있는 것은 좋지 않다.

Package	Class	Line
ASAP.Canvas	MYCanvas	114

D. 사용되지 않는 변수 : 사용되지 않는 변수 4개발견

Package	Class	Line
ASAP.Canvas	MyCanvas	72,73,74,76

E. 코드라인에 주석처리된 부분

Package	Class	Line
ASAP.Canvas	Image	121
	MYCanvas	118
	SelectedImage	71

F. 불필요한 return 변수 : return 변수가 바로 직전에 선언된 경우 불필요하다.

Package	Class	Line
ASAP.Canvas	MYCanvas	159, 165

G. If(a==true)같이 사용한 경우 : if(a) 라는 것으로도 충분.

Package	Class	Line
ASAP.Tools	MoveTool	19, 36
	BrushTool	17
	CircleTool	21
	EraseTool	19
	PenTool	34
	PipetteTool	21
	RectangleTool	25
	SelectTool	19
ASAP.Canvas	MYCanvas	268

H. 가급적 protected 보단 private를 사용할 것을 추천 : 보안의 문제도 있고 예상치 못한 상속으로 인한 상황이 일어날 수 있다.

Package	Class	Line
ASAP.Tools	VirtualTool	7,9,10,12,13,15,16
ASAP.Canvas	VirtualImage	6

I. 절대적으로 더 이상 상속되지 않는 클래스의 경우 Final선언 : 설계상 마지막 구현 class라는 것을 표시하기 위해.

Package	Class	Line
ASAP.Tools	CircleTool	19, 36
	EraseTool	17
	MoveTool	21
	PenTool	19
	PipetteTool	34
	RectangleTool	21

J. NullPointExeption 발생 : NPE의 경우는 발생해서는 안돼는 오류중 하나이다. 그러므로 try, catch 를 이용해서 잡아내서 고쳤다고 한들 근본적인 해결은 아님. NPE 발생 자체가 일어나지 않도록 해야한다.

Package	Class	Line
ASAP.Canvas	MYCanvas	109

K. 각 패키지에 복잡도결과

i. ASAP.Canvas : 각각의 class가 굉장히 많은 method로 구현되어있다. 좀더 다양한 class로 세분화하는 것이 좋다.

ii. ASAP.Tools : class에 비해 너무 거대한 method를 다루고있다. 메소드를 좀더 세분화하거나, 여러가지 분할된 class를 계층화하는 것이 좋다.

iii. ASAP : FileManager가 혼자서 너무 큰 복잡도를 가지고 있다.

4. Jarchitect 보고서

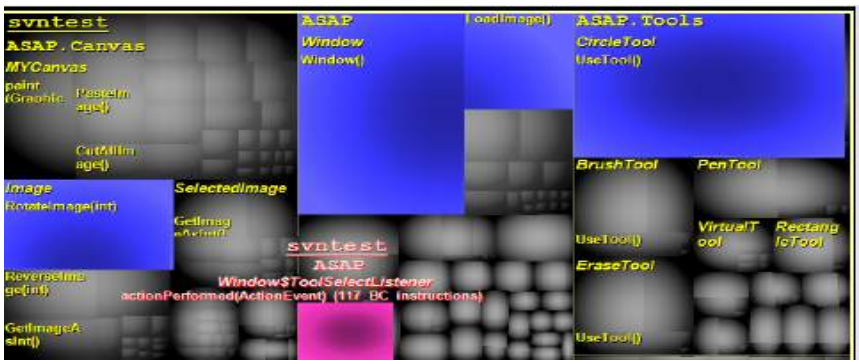
A. 의존성문제 : 패키지끼리의 의존이 사이클구조를 이루고 있다. 패키지끼리의 의존성은 직선 흐름을 가지는 것이 좋다. A패키지가 변하면 B패키지도 변하고 그로인해 다시 A패키지가 영향을 받을 수가 있기 때문에 코드수정이 예측하지 못한 오류를 불러올 가능성이 있기 때문이다.

⇨ ASAP.Canvas 와 ASAP.Tool 은 서로가 서로를 직접적으로 사용하고 있는 dependency cycle구조를 이루고 있다.

⇨ ASAP 와 ASAP.Tool 역시 간접적으로 너무 많은 dependency cycle 을 이루고 있다.

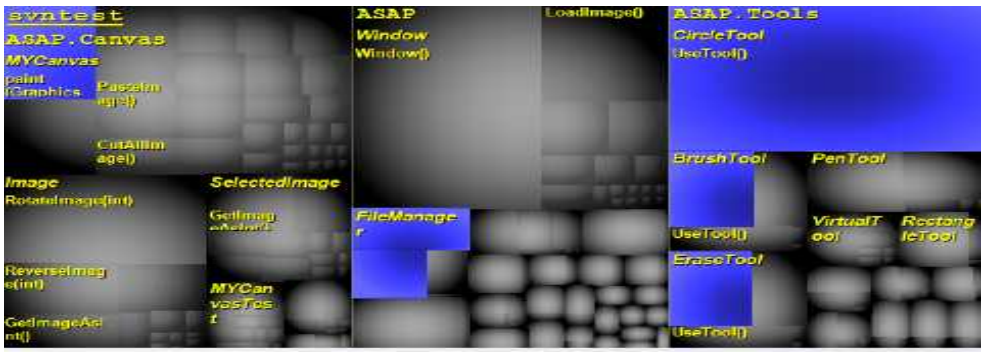
B. Code의 품질

i. 너무 큰 method : 5건



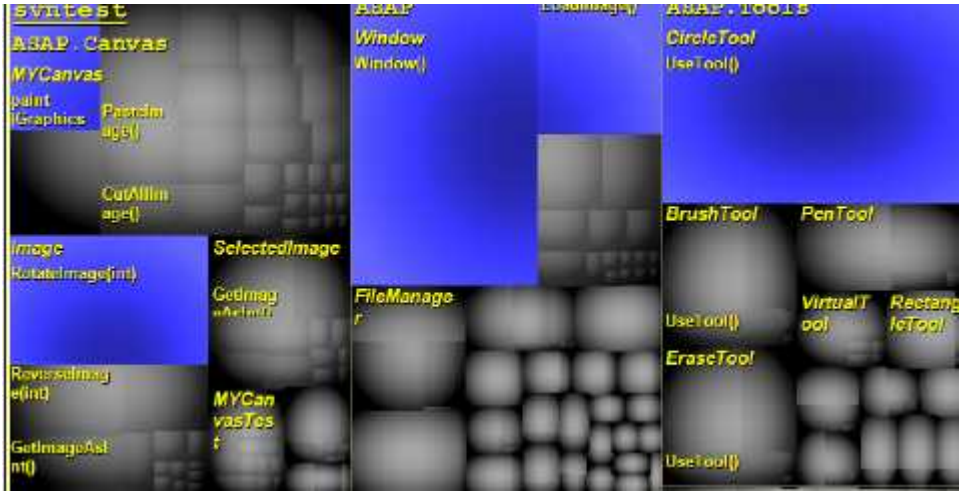
Package	Class	Method
ASAP	Window	Window
ASAP	Window	LoadImage
ASAP.Canvas	Image	RotateImage
ASAP.Tools	CircleTool	UseTool

ii. 주석이 없는 method : 6건



⇨ 전체적으로 양호함.

iii. Method에 너무 많은 지역변수 : 5건



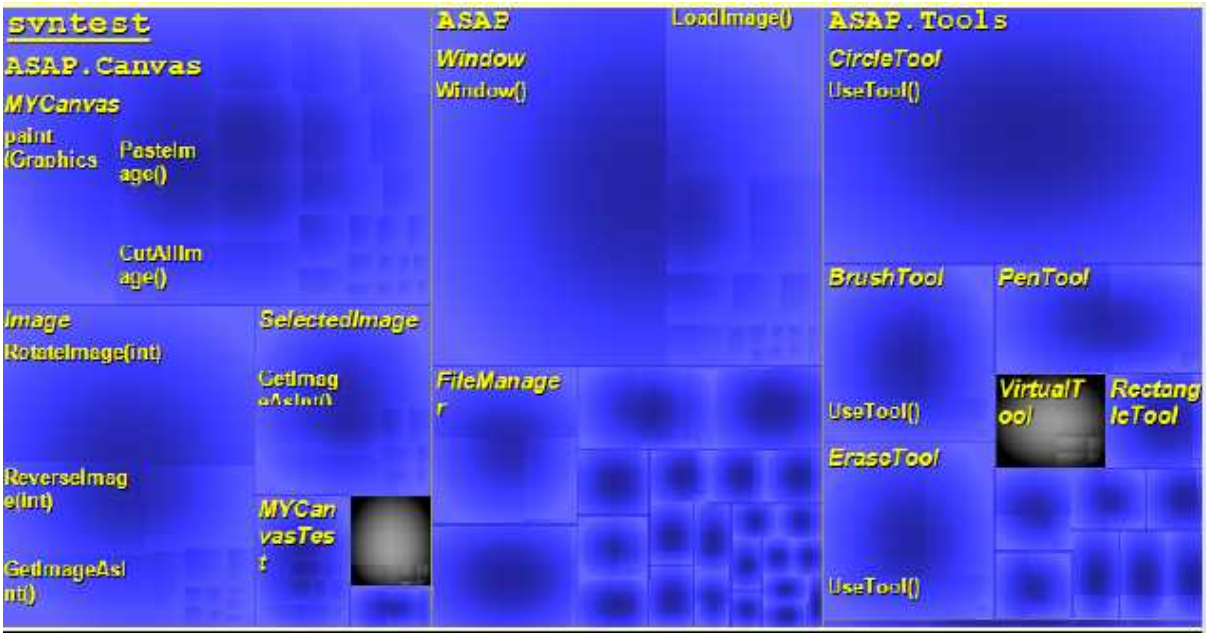
Package	Class	Method
ASAP	Window	Window
ASAP	Window	LoadImage
ASAP.Canvas	MYCanvas	paint
ASAP.Canvas	Image	RotateImage
ASAP.Tools	CircleTool	UseTool

iv. Method가 너무 복잡 : 1건

C. 무분별한 싱글턴패턴의 문제 : 싱글턴패턴은 c의전역변수와 비슷한 문제점을 발생시킬 가능성이 있다. 설계단계에서 최대한 싱글턴 패턴이 나오지 않도록 설계해야 하며 무분별한 싱글턴패턴을 이용은 메모리 관리측면에서 좋지 않다.

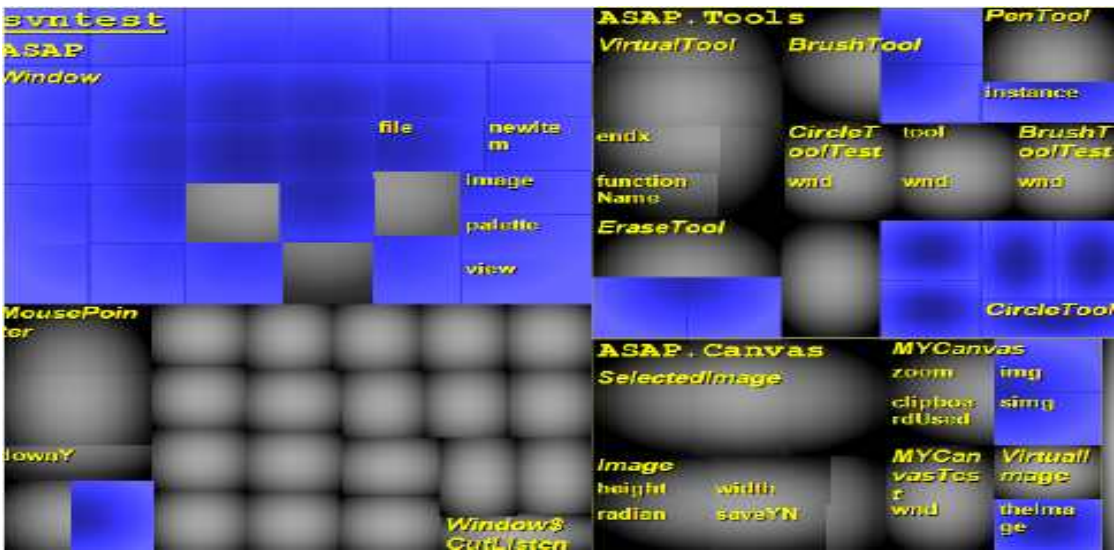
⇨ .Tool의 대부분 메서드

D. 객체지향 디자인측면 : 최종적으로 구체화된 class의 경우, 더 이상 상속되지 않는 다는 표시를 위해서 final keyword를 붙여주는 것이 좋다. 이것은 더 이상 이 class밑으로 구체화할 것이 없다는 암묵적 표시이다.



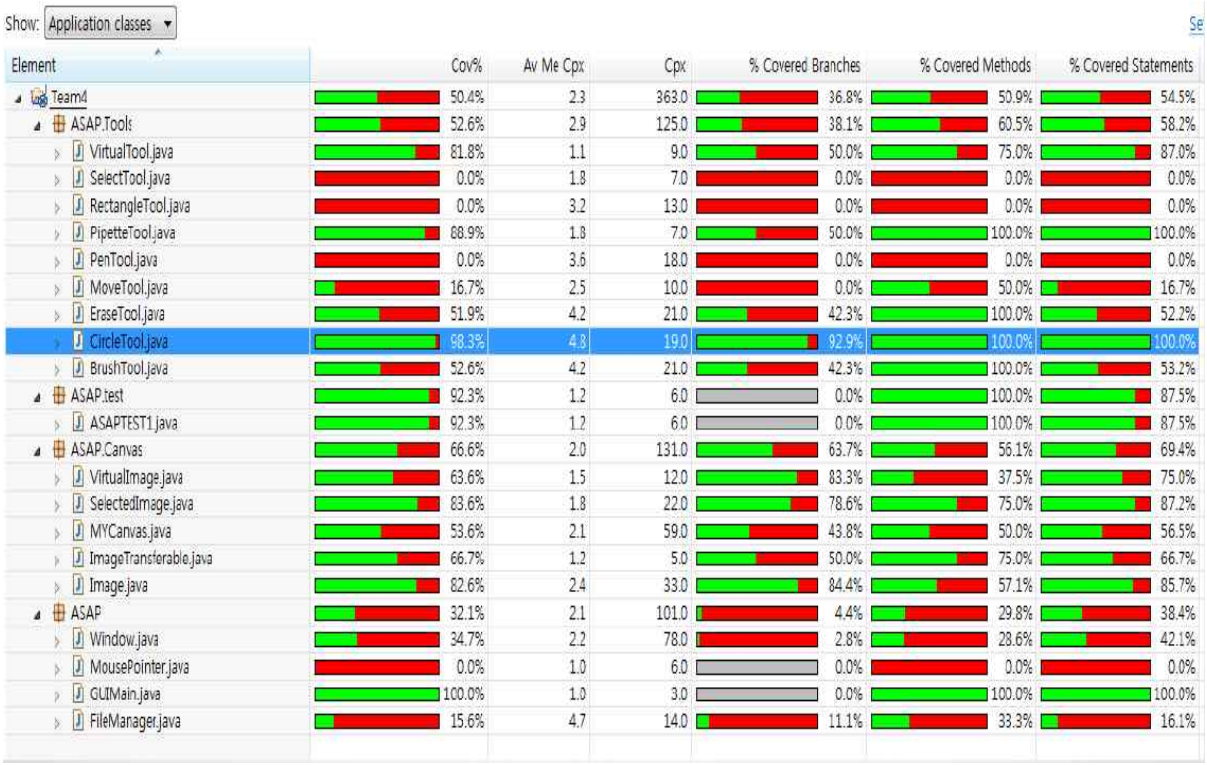
⇨ 거의 대부분의 class에서 문제 발생

E. Field는 가능한한 ReadOnly로 되어야 한다 : Data를 저장하고 있는 Field는 가능한한 쉽게 수정이 불가능하도록 private로 선언되어 함수를 통해 접근되어야 한다. Field를 쉽게 수정하다보면 원본Data를 잃어버리기 쉽고 예기치 못한 오류를 발생시킬 수 있다.



⇨ 대부분의 window class 에서 발견됨.

5. Clover 보고서



Show: Application classes

Element	Cov%	Av Me Cpx	Cpx	% Covered Branches	% Covered Methods	% Covered Statements
Team4	50.4%	2.3	363.0	36.8%	50.9%	54.5%
ASAP.Tools	52.6%	2.9	125.0	38.1%	60.5%	58.2%
VirtualTool.java	81.8%	1.1	9.0	50.0%	75.0%	87.0%
SelectTool.java	0.0%	1.8	7.0	0.0%	0.0%	0.0%
RectangleTool.java	0.0%	3.2	13.0	0.0%	0.0%	0.0%
PipetteTool.java	88.9%	1.8	7.0	50.0%	100.0%	100.0%
PenTool.java	0.0%	3.6	18.0	0.0%	0.0%	0.0%
MoveTool.java	16.7%	2.5	10.0	0.0%	50.0%	16.7%
EraseTool.java	51.9%	4.2	21.0	42.3%	100.0%	52.2%
CircleTool.java	98.3%	4.8	19.0	92.9%	100.0%	100.0%
BrushTool.java	52.6%	4.2	21.0	42.3%	100.0%	53.2%
ASAP.test	92.3%	1.2	6.0	0.0%	100.0%	87.5%
ASAPTEST1.java	92.3%	1.2	6.0	0.0%	100.0%	87.5%
ASAP.Canvas	66.6%	2.0	131.0	63.7%	56.1%	69.4%
VirtualImage.java	63.6%	1.5	12.0	83.3%	37.5%	75.0%
SelectedImage.java	83.6%	1.8	22.0	78.6%	75.0%	87.2%
MYCanvas.java	53.6%	2.1	59.0	43.8%	50.0%	56.5%
ImageTransferable.java	66.7%	1.2	5.0	50.0%	75.0%	66.7%
Image.java	82.6%	2.4	33.0	84.4%	57.1%	85.7%
ASAP	32.1%	2.1	101.0	4.4%	29.8%	38.4%
Window.java	34.7%	2.2	78.0	2.8%	28.6%	42.1%
MousePointer.java	0.0%	1.0	6.0	0.0%	0.0%	0.0%
GUIMain.java	100.0%	1.0	3.0	0.0%	100.0%	100.0%
FileManager.java	15.6%	4.7	14.0	11.1%	33.3%	16.1%

A. 전체 Code Coverage : 50.4%

i. Covered Branches : 36.8%

ii. Covered Methods : 50.9%

iii. Covered Statements : 54.5%

⇨ Code Coverage 비율이 너무 낮다. Test 구문의 수가 더 많아져야 함. 특히 Branches에 대한 test가 많이 낮다.

B. Complex 가 높은 package및 class :

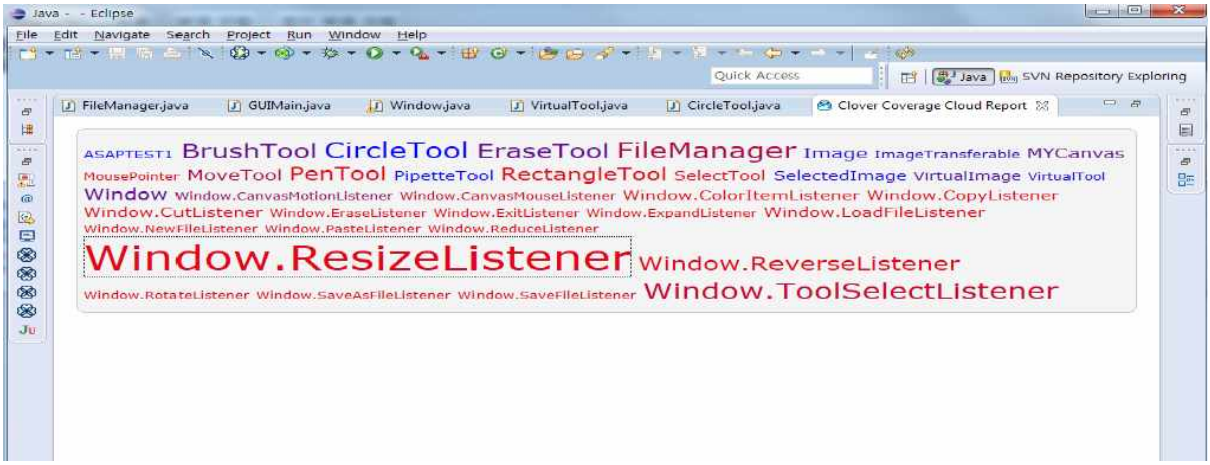
i. Most Complex Classes : ASAP.Canvas

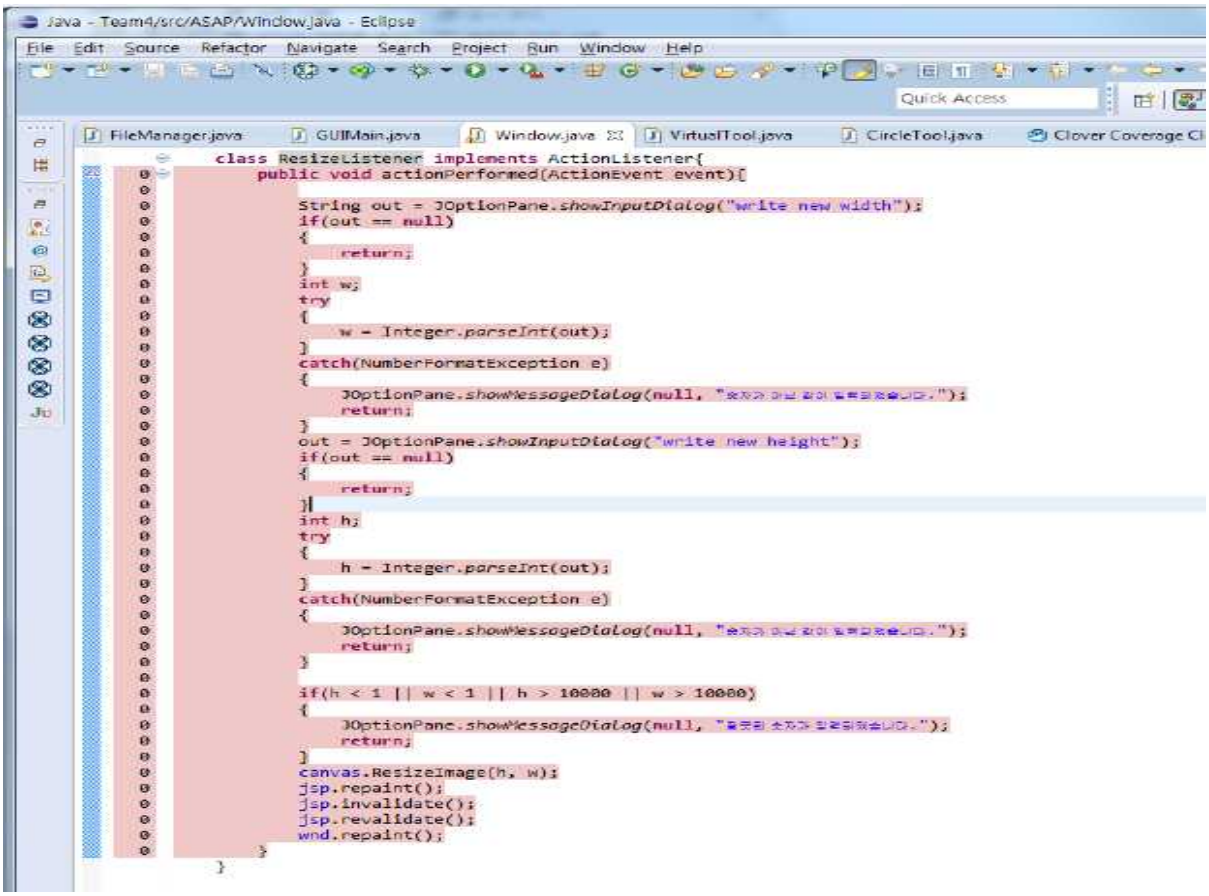
⇨ Test의 비율을 더 높이고 기능을 세분화 해서 Class를 나누는 것이 나을것이라고 분석됨.

ii. Most Complex classes : MYCanvas

C. 위험한 class : Test 비율도 낮고 complex도 높은 class. 가장 잠재적인 위험요소를 많이 가지고 있다고 판단되는 class이다.

⇨ Window.ResizeListener class : 코드내에 중복되는 if문, Test가 0%로 진행됨.





6. JDepend

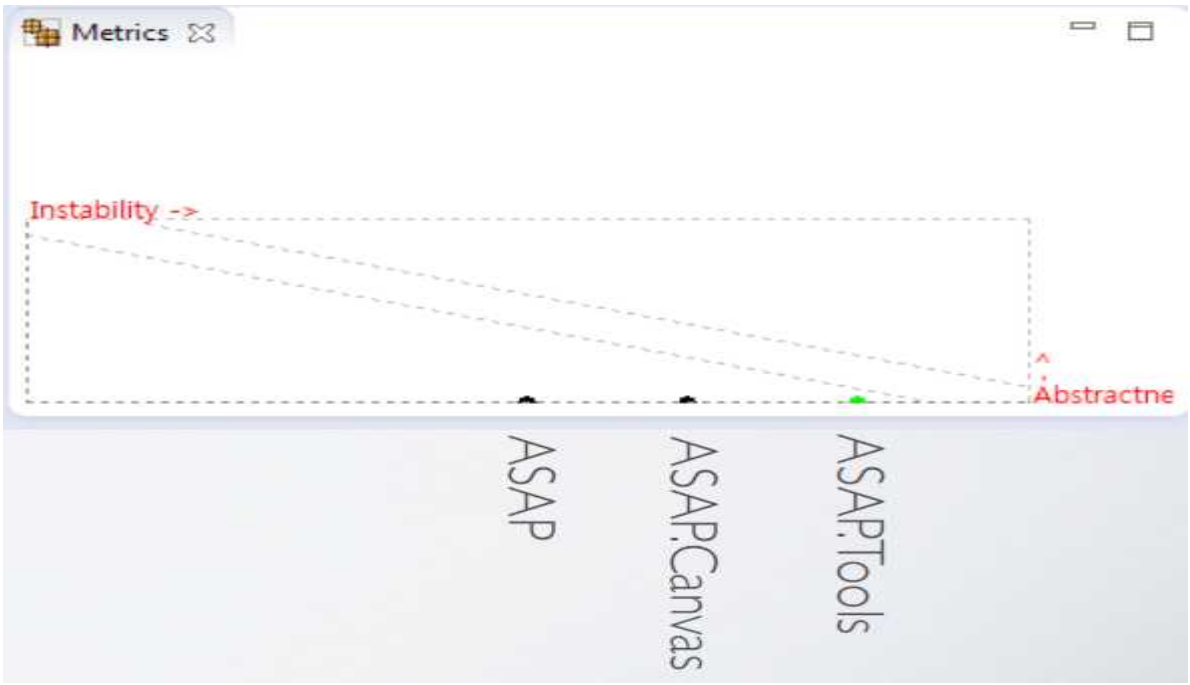
Packages with cycle

Package	CC(concr.cl.)	AC(abstr.cl.)	Ca(aff.)	Ce(eff.)	A	I	D	Cycle!
ASAP	28	0	3	3	0.00	0.50	0.50	
ASAP.Canvas	12	0	2	4	0.00	0.66	0.33	
ASAP.Tools	26	0	1	5	0.00	0.83	0.16	

A. Cycle Dependency가 3개의 package에서 모두 발생. 패키지 구조가 좋지 않다.

B. AC 수치가 0 : AC수치는 각 패키지의 추상화 정도를 나타낸다. 이 수치가 0이라는 것은 추상화가 전혀 안되어있다는 것을 의미한다.

C. D 수치가 0에 가깝지 않은 2개의 패키지 : ASAP, ASAP.Canvas 두개의 패키지에 대해서 Main Sequence 에서 많이 떨어져 있다. Main Sequence란 완전 추상적이면서 안정적인 패키지거나 완전 구체적이면서 불안정한 패키지를 의미한다. 가장 이상적인 패키지 형태를 나타내며 밑에 그래프에서 가운데 대각선에 들어가는 것을 추천하고 있다. (원은 앞에서



⇨ JDepend Tool 사용결과

- ■ 객체지향에 맞지 않는 설계. 추상화가 전혀 안되어 있다.
- ■ Cycle Dependency는 좋은 않은 현상, 패키지 내 구조변경이 필요함.

**** 위에 써진 내용은 절대적으로 틀리다는 것이 아니라 잠재적인 문제를 가지고 있을 가능성이 높다는 의미로 반드시 수정해야할 대상은 아니다. 하지만 몇가지 불필요한 것으로 보이는 code는 좀더 수정하는게 좋을 것.