ASAP Paint(As Soon As Possible)

Static Test Analysis Report

Paint tool developed by OSP(Object Space Process)

On 14 Jun 2013

Team Organization– T4

Kim, Sang Yoon	200811411	gdzergling@core-a.org
Oh, Na Yun	200814189	brilliantjay@naver.com
Lim, Min Woo	200910793	dn3108@gmail.com

1. Sonar 보고서 대응

A. 코드 중복 문제

- 비슷한 기능에서 중복이 발생.

  BrushTool, PenTool, EraseTool 이 서로 중복되는 함수를 보유

  RectangleTool, CircleTool, SelectTool 이 서로 중복되는 함수를 보유

  해당하는 함수들을 묶어 공통된 내역을 새로 상속시키는 관계를 구성함.

B. 비어있는 IF 구문 문제

- Tool 패키지 내에서 많이 발생. 일부는 MYCanvas 내에도 존재함.

  소스코드 해석의 편리함을 위해 일부러 써 놓은 곳이 다수 있음.

  소스코드의 해석을 해치지 않는 한도 내에서 불필요한 IF 구문을 삭제함.

C. 비어있는 finally

- Finally : try catch 이후 어떤 상황이든 반드시 실행하는 부분.

  문제된 finally 영역의 경우, 디버깅을 위해 브레이크 포인트를 걸기 위해 작성했던 부분으로, 현재 해당 부분에 오류가 발생할 위험이 없는 것을 확인, 해당 구문을 삭제함.

D. 사용되지 않은 변수

- 디버깅과정에서의 편리함을 위해 또는 알고리즘의 변환과정 중 정리되지 않았던 변수가 삭제되지 않아 남은 변수들.

  보고서에 작성된 목록을 통해 해당 내역들을 확인하고 쓰이지 않거나, 선언되었지만 아무런 의미를 지니지 않은 변수들을 모두 삭제함.

E. 코드라인 주석

- 코드라인을 주석으로 남겨놓는 경우는 크게 4가지
  - 디버깅을 위해 작성한 다음 릴리즈를 위해 주석 처리한 경우

- 과거 알고리즘과의 비교를 위해 일부러 남겨놓은 경우
- 아직 제 기능을 못하는 알고리즘을 건너뛰게 하기 위해서
- 불필요한 코드이나 이슈상 해결된 내용이 남아있는 경우.

이중 문제가 되는 경우는 3번과 4번이나, 완성된 프로그램이기에 3번은 존재하지 않으며 4번에 해당하는 소스코드는 없는 것으로 확인.

F. 불필요한 return 용 변수

- 수정되지 않음.

G. If(a == true) 와 같은 수식

- If(a) 로 사용하더라도 충분하다고 하나, a == false 와 a == true 의 경우를 일부러 구분짓기 위해 사용.
- 굳이 구분할 필요가 없다 싶은 몇몇 부분을 정리함.

H. Protected 대신 Private를 사용할 것.

- 상속 및 override 와 같은 경우 문제가 발생하는 이슈를 확인 하였으나, 이에 맞춰 함수를 작성할 경우, 현재 완성된 클래스 구조 전체를 들어내고 새로 작성해야 하는 것으로 판단되어 해당 이슈는 인지하는 것으로 끝냄.

I. 최하위 클래스의 final 선언

- 자바 문법의 미숙함으로 인해 발생한 문제. 더 이상 상속되지 않는 모든 클래스에 final 키워드 추가함.

J. Nullpointexception 문제

- 해당 예외는 소스코드 작성시 배열에 발생한 오류를 잡아내기 위해 작성한 코드.
- 현재로선 가치가 소실되어 해당 구문을 삭제함.

K. 패키지 복잡도 문제

- ASAP.Canvas 의 복잡도가 지나치게 높다.

  직접적으로 Image 클래스와 SelectedImage 클래스에 접근할 수 있는 클래스로 실제적인 역할을 이 클래스가 대부분 담당하기에 높은 복잡도는 어쩔

수 없는 것으로 판단함.

- ASAP.Tools 패키지 내의 클래스들의 method가 너무 크다.

  문제가 되는 것으로 예상되는 Circle 클래스의 경우 타원을 그리기 위해 2차 함수를 계산하는 과정이 상당히 긴 것으로 이는 알고리즘의 최적화가 어려움.

2. Jarchitect 보고서 대응

A. 의존성 문제

- ASAP 패키지가 ASAP.Canvas 패키지를, 그리고 그 반대를 서로 참조하는 순환참조가 몇군데 발견됨.
- ASAP와 ASAP.Tool 패키지의 의존을 제거하기 위해 ASAP 내의 Window 클래스가 지닌 color 라는 변수를 하나의 클래스와 패키지로 새로 분리해냄.(SelectedColor 클래스)
- 이외의 부분에서는 순환참조 문제 발생하지 않음.

B. 너무 큰 함수 문제

- Window 클래스의 생성자, Image 클래스의 RotateImage, CircleTool 의 UseTool 이 지정됨
- 이중 생성자는 전체를 3부분으로 분리, 그중 2부분을 각각 TopsideMake() 함수와 LeftSideMake 함수로 분리해냄.
- 나머지, RotateImage와 Usetool의 경우 하나의 알고리즘으로 묶여있기에 분리가 어려우므로 그대로 놔둠.

C. 싱글턴 패턴 문제

- 싱글턴 패턴 적용이 메모리 관리 측면에서 좋지 않은 점이 존재하나, 설계상의 이점을 가질수가 있으며 현재단계에선 구조를 대폭 수정하기가 어려우므로 이대로 놔둠.

D. 객체지향 디자인 측면

- Java 사용 미숙으로 발생한 문제, 상위 클래스는 abstract로 최하위 클래스는

final로 각각을 구분해야 했으나, 이것이 제대로 진행되지 않아 추상화가 전혀 진행되지 않은 것으로 표시되었음.

3. Clover 보고서 대응

   A. 해당 보고서는 기존에 작성했던 JUnit 테스트를 기반으로 하여 만들어진 보고서임.

   B. 해당 보고서에서 나타난 Code Coverage 수치가 낮다는 말은 테스트 하지 않은 또는 테스트 할 수 없는 부분이 많다는 것을 나타냄.

   C. 그러나 중복된 내용 또는 단순화시킨 내용이 존재하는 클래스 (pentool 은 brush tool의 하위호환) 의 경우 굳이 테스트를 진행할 필요성이 없다고 판단되어 제외함.

   D. 따라서 추가된 테스트 보고서에는 Mycanvas 와 같은 메소드가 많지만 테스트가 진행되지 않았던 부분을 추가적으로 진행하는 것으로 결정함. (추가 내용은 단위 시험 보고서 참고)

4. JDepend 보고서 대응

   A. Cycle Dependency Issue

      - Jarchitect 보고서에서 설명함.

   B. 추상화도 0 Issue

      - Abstract 선언이 없었기에 발생한 문제.

5. 결론.

   A. 전체적으로 Java 문법 사용의 미숙함과 Debugging 단계에서 남은 찌꺼기가 제대로 처리되지 않음으로 발생한 문제가 대다수를 차지함.

   B. 설계상의 결함을 소스코드에 죄를 짓는 방법으로 해결한 부분은 여지없이 검출되었음.

   C. 코딩을 한 본인이 쉽게 생각하지 못할 코드의 중복도와 같은 부분에 대한 검사는 설계의 중요성을 다시 한 번 느끼게 해줌.