Software Modeling & Analysis

OSP Final

[Demonstration]

- 1 to 10 CPT Tool -

Team.#	6
과목명	소프트웨어 모델링 및 분석
담당교수	유준범 교수님
팀원	201211938 황준익
	201310350 손성호
	201414135 이광제
	201212088 이용주

Contents

Activity 1. Static Analysis 대응

두 조에서 대표적인 PMD, Metrics나 FindBugs를 이용하였기 때문에 공통적인 문제가 검출되었다.

두 조의 Static Analysis 보고에 대한 대응은 다음과 같다.

PMD: 중복된 코드가 많음, 일부 변수 형식의 불일치.

우선 명명 에러에 관한 점은 우리의 개발 프로세스상 설계에서 어긋나는 변수 명명은 없었고, 대부분 직관적으로 도움이 되기 위해 "_" 나 대문자를 사용하였기 때문에 크게 문제가 된다고 판단되지 않았다.

```
JLabel lblNewLabel_1 = new JLabel("Select");
lblNewLabel_1.setHorizontalAlignment(SwingConstants.CENTER);
lblNewLabel_1.setFont(new Font("", Font.BOLD, 20));
lblNewLabel_1.setForeground(Color.WHITE);
lblNewLabel_1.setOpaque(true);
lblNewLabel_1.setBackground(Icon.makeColorRGBA(64,64,64,255));
lblNewLabel_1.setBounds(319, 49, 105, 25);
this.add(lblNewLabel_1);
```

코드 중복은 GUI 컴포넌트 초기화 과정에서 많은 부분이 중복되었다고 생각한다. 하지만 컴포넌트에 따라 필요한 옵션과 필요 없는 옵션이 다르며 이 초기화 부분을 함수로 묶어도 큰 효율 상승은 없을 것이라 판단하였다.

Metrics: 일부 Package에 대한 Cyclomatic Complexity가 높다.

개발자가 재귀보다 루프나 스위치를 좋아하여 블록이 꽤 많았다. 하지만 삼항 연산자 등을 적극 활용하여 코드의 복잡도를 줄이고자 노력하였고 복잡도 수치가 치명적 이라고는 볼 수 없었기 때문에 무난히 현황을 받아들이기로 하였다.

FindBugs: 1) 파일 입출력 관련 null Pointer 처리 확인 요망
2) Calculation과정에서 쓰이지 않는 지역변수
3) clone()사용에서의 경고

FindBugs에서 검출된 위험요소인 3가지 위험요소를 다음과 같이 수정하였다.

```
 finally
 {
     if(put != null) put.close();
 }
finally
{
    if(input != null) input.close();
}
```

1. 파일 입출력에서 예외처리 후 Reader를 close하는 과정에서 null인 상태 확인을 하지 않아서 나타나는 null Pointer Error => 확인 후 close

```
public ArrayList<CPT.Category> sort(){

    resultList.clear();
    resultList_filtered.clear();

    ArrayList<CPT.Property> temparoryPropertyList = new ArrayList<CPT.Property>();
    ArrayList<CPT.Category> tempCategoryList = new ArrayList<CPT.Category>();
    //ArrayList<CPT.RepValue> tempRepvList = new ArrayList<CPT.RepValue>();
```

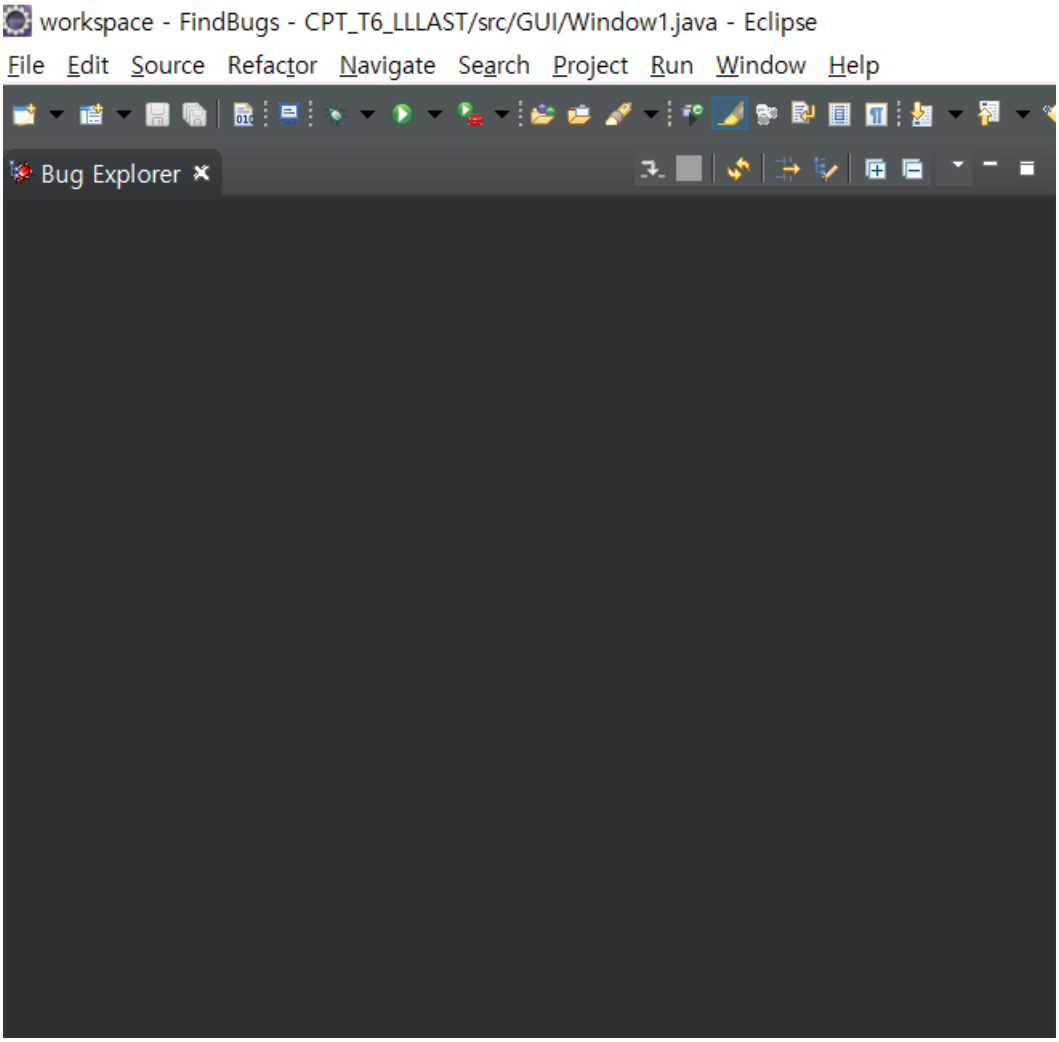
2. Calculation 과정 중에 사용하지 않는 tempRepvList 지역변수를 제거

```
public class Result implements Cloneable{



@Override
public Result clone()
```

3. Result의 clone의 깊은 복사를 위한 인터페이스와 오버라이딩 메소드 구현



위험요인을 모두 수정하여 FindBugs 에서 위험요소가 검출되지 않는다.

Activity 2. OOPT Review

- Advantage

  단계별 프로젝트 설계를 함으로 인해서, Waterfall 모델과 달리 한 번에 모든 사항에 대한 결정을 필요로 하지 않기 때문에 부담이 덜하다.

  현재 단계에서의 이슈에만 집중해서 프로젝트를 진행하기 때문에, 다음 단계에서의 이슈에 대한 고려를 배제하고 프로젝트를 진행할 수 있어 각 단계별 과정을 단순화할 수 있었다.

  각 단계별로 세부적인 사항을 고정하지 않고 개발 과정을 진행해서 고정된 내용으로 프로젝트를 진행하지 않고 자유로운 프로젝트 진행이 가능했다.

  문서의 지속적인 관리를 통해서 구현 시 필요한 정보를 즉각적으로 확인할 수 있었다.

  프로젝트를 진행하다 보면, 초기 목적과 다르게 프로젝트가 진행되는 경우가 많은데 단계별로 프로젝트를 진행하다 보니, 초기에 계획한 대로 프로그램을 완성할 수 있었다.

- Disadvantage

  각 단계별로 문서를 개별적으로 작성해야 하고, 기존의 문서를 지속적으로 수정해야 해서 문서 관리하는 데 시간이 많이 소요되었다.

  프로세스 단계에 대한 가이드라인이 부족해서 작년 혹은 재작년 내용을 지속적으로 확인해가며 개발 단계를 진행해야 했다.

  프로젝트가 큰 경우에는 개발 프로세스를 따라서 프로젝트를 진행하는 것이 전체 프로세스를 여러 단계로 나누어 단계별 확장이 가능한 장점이 될 수 있지만, 이번 프로젝트의 경우 규모가 그렇게 크지 않아서, 오히려 시간이 더 많이 소요되는 단점이 있었다.

Activity 3. Summary

- 소감

  이광제 : OOPT 를 하면서 느꼈던 가장 큰 점은, 문서가 귀찮아도 내가 생각했던 걸 저장한다는 점에서 굉장히 이롭다는 것을 느꼈다.

  손성호 : 문서를 단계별로 작성하니까, 나중에 모르는 부분이나 기존의 내용과 다른 부분이 있는 경우 문서를 확인할 수 있기 때문에 프로젝트를 좀 더 효율적으로 진행할 수 있었다. 또한, 각 단계별로 다음 단계에 대한 생각을 배제하고 프로세스를 진행하니까 좀 더 생각이 간결해지는 것 같았다.

  황준익 : 문서를 작성하는 데 어려움이 많았고, 프로세스 단계를 진행함에 따라 문서의 수정이 너무 많아서 힘든 부분이 있었지만, 프로그램 개발 단계에 접어들면서부터 문서에 기록했던 내용들이 프로그램에 적용되는 것을 보면서 단계별로 문서를 작성하는 것이 큰 도움이 됨을 느꼈다.

  이용주 : 프로그램을 만들 때, 어떤 개발 프로세스에 따라 프로젝트를 진행하느냐에 따라서 개발 과정이 많이 다를 수 있다는 것을 알 수 있었다. 다음에 프로젝트를 진행한다면 개발 프로세스에 조금 더 신경을 쓸 수 있을 것 같다.