C++,
The Convenient ATM

T2
201411262 김도현
201411271 박상우
201411312 장하나
201411316 정진호

CONTENTS

1. System Test 대응

2. Static Analysis 대응

3. OOPT Review

1. System Test 대응

1. System Test 대응

- 거래 내역을 조회할 때 시작 날짜와 끝 날짜를 동일하게 입력하였을 때, 시작 일자 00:00 의 거래 내역은 조회되지 않는 현상 발생

- if 문의 조건을 수정하여 해결

```
157  157          for (Transaction t : transactions) {
158       -           if (t.getTime().compareTo(start) > 0 && t.getTime().compareTo(end) < 0) {
     158  +           if (t.getTime().compareTo(start) >= 0 && t.getTime().compareTo(end) < 0) {
159  159                 list.add(t);
```

1. System Test 대응

- 복권 주차에 0이나 실제 존재하지 않는 주차를 입력해도 오류로 인식하지 않음
- 복권 번호에 0을 입력하여도 오류로 인식하지 않음

1. System Test 대응

- 0을 예외에 포함하여 해결
- 로또 시작 주차와의 차이를 계산하여 주차 존재 여부 확인

```
if (week <= 0 || week > weekMax) {
```

```
if (values[i] <= 0 || values[i] > 45) {
```

```
long weekMax = (new Date().getTime() - new Date(102, 10, 30).getTime()) / (7 * 24 * 60 * 60 * 1000);
```

2. Static Analysis 대응

2. Static Analysis 대응

- 카드 분실 신고 switch 문

수정 전

```
switch (answer) {
    case JOptionPane.YES_OPTION:
        try {
            system.askRenewCard(true);
        } catch (DataStoreError ex) {
        }

        JOptionPane.showMessageDialog(parentFrame,
            setLocalizedString(system, card[i].toString() + " 카드를 재발급 요청하였습니다", card[i].toString() + " is requested to renew"),
            "Info",
            JOptionPane.INFORMATION_MESSAGE);
    case JOptionPane.NO_OPTION:
        parentFrame.setContentPane(new SelectFunction(parentFrame, system).getPanel());
        parentFrame.invalidate();
        parentFrame.validate();
        break;
```

수정 후

```
if (answer == 0) {
    try {
        system.askRenewCard(true);
    } catch (DataStoreError ex) {
    }
}
JOptionPane.showMessageDialog(parentFrame, setLocalizedString(system, card[i].toString() + " 카드를 재발급 요청하였습니다", card[i].toString() + " is requested to renew"),
    "Info",
    JOptionPane.INFORMATION_MESSAGE);
parentFrame.setContentPane(new SelectFunction(parentFrame, system).getPanel());
parentFrame.invalidate();
parentFrame.validate();
}
```

2. Static Analysis 대응

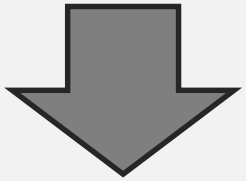
- 지폐 수 계산 default 문

```java
public int[] calcBillAmount(int cashAmount, String cashType) {

    int[] bills = new int[11];
    int[] billType;
    int length;

    switch (cashType) {
        case "WON":
            billType = new int[]{50000, 10000};
            length = billType.length;
            for (int i = 0; i < length; i++) {
                int bill = cashAmount / billType[i];
                bills[3 - i] = bill;
                cashAmount -= billType[i] * bill;
            }
            break;
        case "Dollar":
            billType = new int[]{100, 50, 20, 10, 5, 2, 1};
            length = billType.length;
            for (int i = 0; i < length; i++) {
                int bill = cashAmount / billType[i];
                bills[10 - i] = bill;
                cashAmount -= billType[i] * bill;
            }
            break;
        default:
            return new int[]{0,0,0,0,0,0,0,0,0,0,0};
    }
    return bills;
```

```java
default:
    return null;
    return new int[]{0,0,0,0,0,0,0,0,0,0,0};
```

지폐 종류 인식 실패 시 null 리턴하나

null 과 int를 더하는건 불가능



대신 0 반환

2. Static Analysis 대응

- String 출력 중복

```
String[] card = system.getUser().getCardList();
```

```
        setLocalizedString(system, card[i].toString() + " 카드를 중지처리하였습니다", card[i].toString() + " is now closed"),
        setLocalizedString(system, card[i] + " 카드를 중지처리하였습니다", card[i] + " is now closed"),
        "Info",
        JOptionPane.INFORMATION_MESSAGE);

lic class ReportLostCard extends JFrame {
        } catch (DataStoreError ex) {
        }
    }
    JOptionPane.showMessageDialog(parentFrame, setLocalizedString(system, card[i].toString() + " 카드를 재발급 요청하였습니다", card[i].toString() + " is requested to renew"),
    JOptionPane.showMessageDialog(parentFrame, setLocalizedString(system, card[i] + " 카드를 재발급 요청하였습니다", card[i] + " is requested to renew"),
```

2. Static Analysis 대응

- User Frame, Admin Frame 접근제어지시자

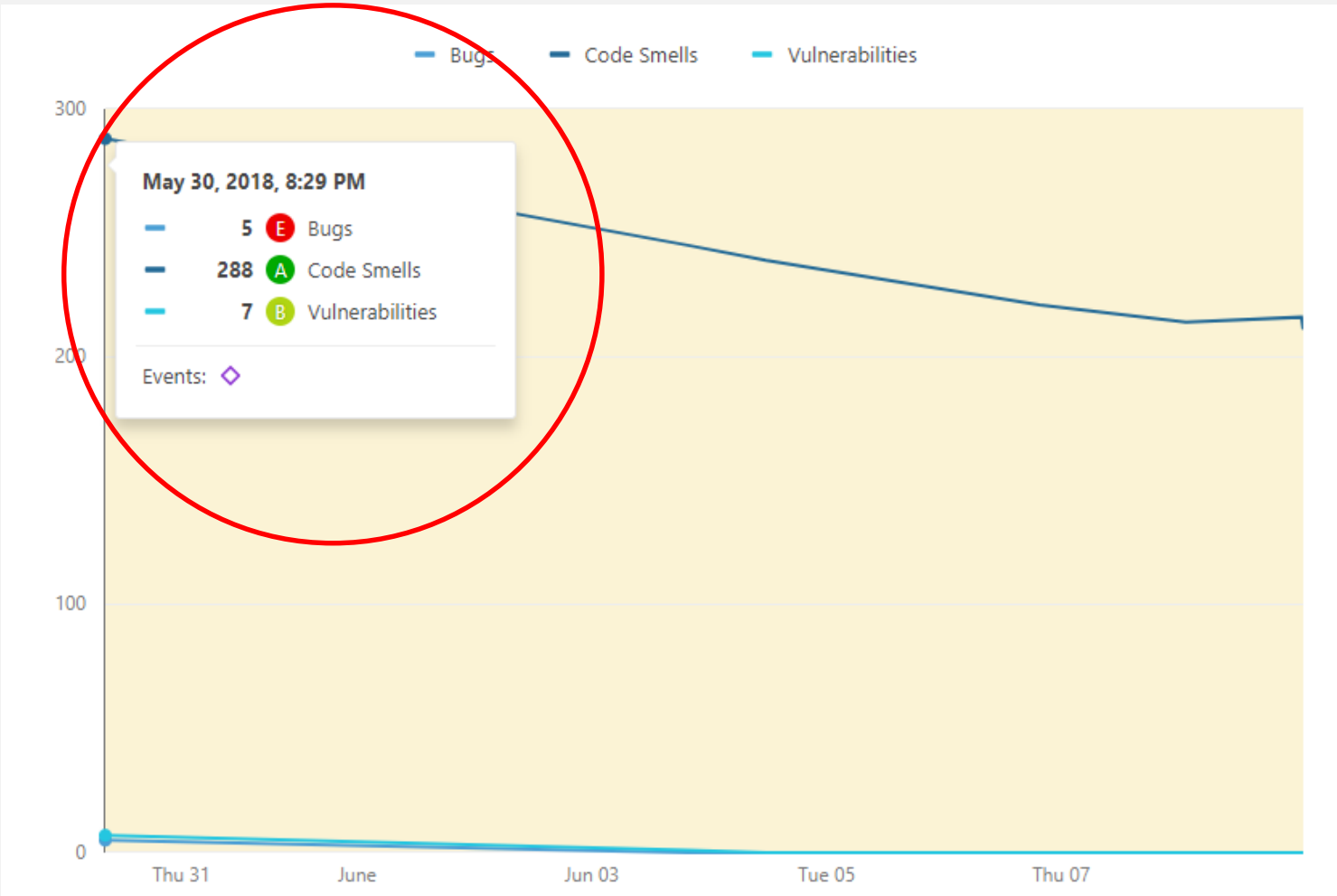
```
private Main() {
    super();
    userFrame = new JFrame( title: "C++, The Convenient ATM");
    adminFrame = new JFrame( title: "Admin Interface for C++ ATM");
}

public static Main getInstance() {
    if ( application == null ) {
        application = new Main();
    }

    return application;
}
```

```
 public class Main {
-    public JFrame userFrame;
-    public JFrame adminFrame;
+    private JFrame userFrame;
+    private JFrame adminFrame;
```

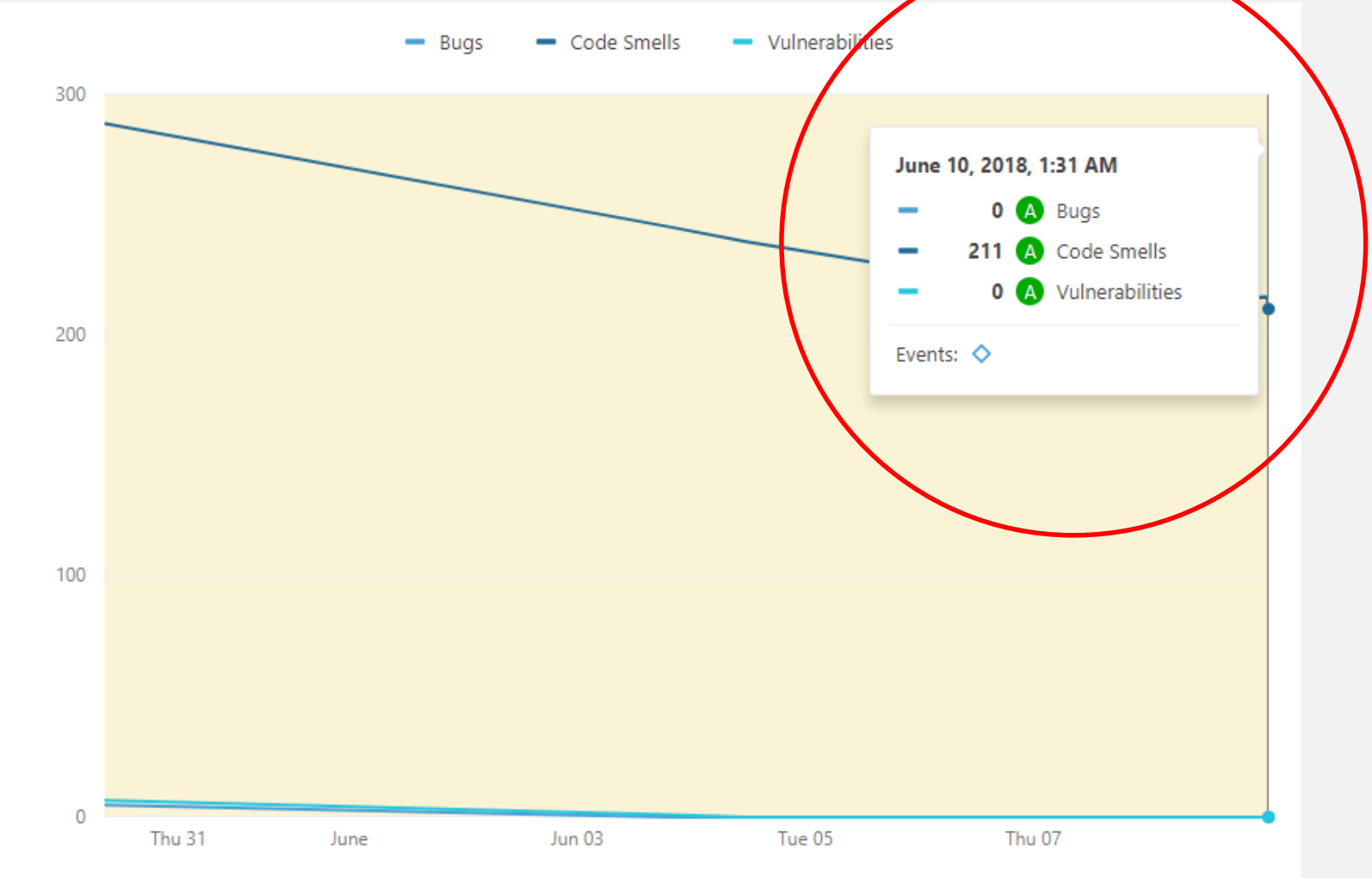
public -> private

잠재적 보안 문제 해결

2. Static Analysis 대응



2. Static Analysis 대응



3. OOPT Review

3. OOPT Review

- 소프트웨어 모델링 작업의 중요성 확인
- 구현단계 전까지의 설계 및 수정 작업의 반복으로 실제 소프트웨어의 구현이 좀 더 용이
- 다양한 소프트웨어 검증 단계를 거치며 구현을 하니 수정이 훨씬 쉬움
- 정적 분석을 통하여 평소 코딩 스타일에서 미흡했던 부분들을 고침.

3. OOPT Review

장점

- 요구사항의 정리가 매우 용이
- Revise를 수행하므로 소프트웨어 아키텍처 및 문서에 대한 수정이 없어 큰일이 일어나지 않음
- 요구사항의 수정이 있더라도 반영이 매우 쉬움
- 클라이언트가 결과물을 최대한 빠르게 볼 수 있어서 개발자의 성취감과 클라이언트의 만족을 동시에 얻을 수 있음
- 우리팀이 반복적으로 수정하고 설계한 것을 바탕으로 코드를 만드니까 클래스를 만들고 내부 코드를 채워가는 과정이 굉장히 순조로움

3. OOPT Review

단점

- 작은 수정이 많이 필요할 수 있어서 사실상 큰 수정 한번 하는 것과 큰 차이가 없는 상황이 발생 가능
- 잦은 수정으로 인한 피로감이 듬

3. OOPT Review

검증팀과의 협업

- 검증팀의 환경이 생소하여 익숙해지는 데 시간이 필요
- 서로간의 소통이 잘 되어 CTIP 환경을 잘 사용함
- 팀 간 협업 또한 원활