Software Engineering term Project

* Issue targeting System

2팀(총원1명)

20203758 정찬혁

2024년도 봄학기 소프트웨어 공학 텀 프로젝트에 대한 리포트 입니다.

2024.6.1.

Abstraction.

사용 언어 Java

UI 구현: Java - Swing

가산점 요인: 없음.

사용한 저장소: 파일 시스템

구현됨 :

이슈 브라우즈 및 검색

이슈 등록 (자동으로 채워지는 기능 포함.)

이슈 배정을 포함한 이슈 상태 변경

이슈 상세 정보 확인 (추가설명)

이슈 코멘트 추가

회원가입

사용자 조회

로그인 / 로그아웃

미구현됨 :

이슈 통계 분석

프로젝트를 진행하기에 앞서 이 프로젝트는 봄학기 텀 프로젝트로써 주제가 정해져 있었습니다.

중간에 주제가 바뀔 위험도 없었고 요구사항의 추가가 일어나지도 않을것이 확실하였기에

waterfall model을 사용하여 프로그램 개발을 진행하였습니다.

리스크가 확실하게 통제되어 있었기 때문에 waterfall을 적용시키는데 문제가 없었습니다.

사실 1인 팀이라 어떤 모델을 사용하여도 유연성이 보장이 되었기에… 모델이 크게 중요하지 않았지만 그래도 배운 지식을 적용시켜 보고 싶었습니다.

먼저, 프로젝트를 시작함과 동시에 인터페이스의 설계를 시작했습니다. ‘

영속하는 저장소로 파일 시스템을 사용할 예정이였고

MVC 패턴을 사용하여 시스템을 구현할 예정이였습니다.

현실적으로 여력이 부족하였기 때문에 Fancy한 프로그램을 만든다는 생각보다 배운 개념을 적용시킨다는 계획 하에

조금 수수하더라도 잘 기능하는 프로그램을 만들자 라는 생각으로 프로젝트를 진행하였습니다.

사용한 디자인 패턴입니다.

먼저 가장 주의를 기울였던 부분은 MVC 패턴의 적용입니다.

다중 인터페이스를 사용하기 위해서는 MVC 패턴이 최선이라고 생각 하였고(구현하지는 못했습니다.)

제시된 주제가 MVC패턴을 적용시키기에 적합한 주제이기도 하였습니다.

Model로써 2차원 스트링 벡터, file system등을 이용하였고

View로써 Swing 라이브러리를 사용하였습니다.

Controller로써 Account와 Textbox클래스등 기능을 가진 클래스와 메서드등을 이용하여 패턴의 적용을 노력하였습니다.

또한 SOLID원칙의 단일책임의 원칙을 지키려 노력하였습니다.

이에 하나의 메서드는 최대한 하나의 기능만 수행할수 있도록, 코드가 지나치게 비대해지지 않는것에 주의를 기울였습니다.

아래는 SOLID원칙을 지키기 위해 노력한 예시입니다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

(textbox Class의 일부분)

하나의 메서드에서 txt를 읽고 바로 벡터로 옮길수도 있지만, readtxt는 단순히 텍스트만을 읽는 기능을, txtToVector 메서드는 그 읽어진 값을 벡터로 옮기는 기능만을 합니다.

rewriteIssue와 IssueEditing 메서드도 이와 유사한 관계를 가지고 있습니다.

전체적인 프로그램의 로직은 다음과 같습니다. (이후에 다이어그램이 존재합니다)

먼저 사용자가 로그인을 시도하면 해당 아이디와 비밀번호가 존재하는지 검사를 하고 존재한다면 로그인을 허용합니다.

로그인에 성공하면 mainUI를 띄워 사용자가 기능을 사용할수 있게 합니다.

MainUI의 생김새입니다.

텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

제시해주신 기능으로 검색, 이슈생성, 사용자 조회를 큰 버튼으로 넣어두었고

테이블 리프레시를 위한 ‘R’ 이라는 버튼을 만들어 두었습니다.

(리프레시라서 직관적으로 R이라는 이름을 붙였으나 새로고침이라는 단어를 쓸걸 그랬습니다.)

생성된 티켓은 바로 아래에 생성되며, 상단에 있는 메뉴바로도 기능 선택이 가능합니다.

지금부터 예제 시나리오를 진행해보겠습니다. (영상에도 포함 되어있습니다. )

텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

Tester1으로 로그인을 합니다. 편의상 모든 계정의 비밀번호는 1234로 하였습니다.

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

메인 화면에 접근하였고, 기존에 추가해둔 데이터들이 보입니다.

새로운 이슈를 생성하고 티켓 번호는 제 학번인 20203758로 해보겠습니다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

20203758번 이슈가 추가된 모습을 볼 수 있습니다.

이슈를 클릭하여 코멘트를 추가하겠습니다.

텍스트, 스크린샷, 디스플레이, 컴퓨터이(가) 표시된 사진

자동 생성된 설명

기본으로 None 으로 되어 있는 코멘트 란에 문구를 추가하겠습니다.

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

코멘트를 추가하였고 추가된 코맨트는 이슈를 클릭하여 detail을 확인하면 볼수 있습니다.

시스템에 의해 자동으로 Fixer는 tester1이 되었습니다.

이제 로그아웃하고 PL1으로 로그인을 한 뒤 dev1을 배정하도록 하겠습니다.

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

이제 Fixer는 PL1dl 되었고 dev1이 배정되었습니다.

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

Dev1은 자신이 배정되어있는 이슈를 검색하고 이 이슈를 확인합니다.

텍스트, 스크린샷, 폰트, 화이트이(가) 표시된 사진

자동 생성된 설명

Tester1과 PL1이 추가한 코멘트를 dev1은 확인할수 있습니다.

코멘트를 추가하고 status를 fixed로 바꿉니다.

다음은 다이어그램에 대한 설명입니다.

전체 다이어그램은 제출물 다이어그램 폴더 안에 들어 있습니다.

문서의 분량이 지나치게 비대해지는 것을 방지하기 위해 다이어그램 이미지는 따로 삽입하지 않고 파일명에 대한 설명만을 덧붙이겠습니다.

Class Diagrams

1. Account Class

Account Class는 로그인된 사용자의 이름을 저장하는 name 필드를 가지고 있으며

Account 인터페이스를 구현하는 객체입니다.

Account class의 핵심 기능으로는 그래픽 인터페이스로부터 전달받은 사용자 로그인 시도를 점검하고, 새로운 계정을 생성하는 기능을 수행합니다.

계정정보를 저장하는 파일로부터 사용자 목록을 출력하는 기능을 또한 포함하고 있습니다.

1. Main UI Class

mainUI Class는 MVC패턴의 view의 구현으로써 기능을 사용하기 위한 GUI로 기능합니다.

SwingUI라는 인터페이스의 구현이며 Swing UI는 register, login, mainframe 이라는 세가지 UI에 대한 명세를 가지고 있습니다.

실제 내부에서 호출하는 ui창은 여럿이나 인터페이스에 단 3가지 UI에 대한 명세만 가진 이유는 이 세가지 UI를 제외한 다른 UI는 mainframeUI에 종속적으로 기능하는 UI이기 때문입니다.

각 기능들은 mainframeUI에 존재하는 버튼과 메뉴바등을 이용하여 호출이 되며 이에 인터페이스에 해당 내용을 포함시킬 필요가 없다고 판단하였습니다.

Main UI Class는 Account Class와 Textbox Class를 include하는 관계를 가지고 있으며

핵심이 되는 테이블의 모델로써 2차원 스트링 벡터, Vector<Vector<String>>을 사용하여 모델을 표현하고 있습니다.

프로그램의 로직은 MainUI class에 있는 Login 혹은 mainFrameUI를 호출하며 개시됩니다.

1. textbox Class

textbox Class는 파일 시스템을 사용하여 데이터를 리턴하는 컨트롤러로써의 기능을 수행하는 클래스입니다. 이슈의 주소를 String으로써 자신의 필드에 가지고 있고 이 주소로부터 데이터를 읽어서 정해진 프로토콜에 따라 가공하여 view로 전달하는 기능을 수행합니다.

또한 view로부터 전달받은 값을 저장하는 기능 또한 수행하고 있습니다.

Textbox class는 역시 textBoxInterface를 구현하는 객체입니다.

전체 클래스 다이어그램을 종합하여 보자면 다음과 같습니다.

텍스트, 평행, 도표, 번호이(가) 표시된 사진

자동 생성된 설명

Domain Model Diagram

다음은 도메인 모델에 대한 분석입니다.

명세에 의하면 사용자는 네가지 분류로 존재합니다. Admin/ PL/ Tester/ Developer

이 네가지 사용자가 존재하기에 이들은 User라는 객체의 하위 클래스로 분류하였습니다.

또한 티켓, 즉 이슈는 변화하는 고정된 값, 즉 Priority와 Status를 가집니다.

이는 각각 5개의 값을 가지고 이는 다음과 같습니다.

Priority: Blocker/ Critical/ Major/ Minor/ Trivial

Status: New/ Assigned/ Resolved/ Closed/ Re-Opened

역시 각각은 각각의 이름을 가지는 서브 클래스로 판단을 하였습니다.

전체 도메인 모델의 다이어그램은 다음과 같습니다.

(다음페이지)

텍스트, 도표, 평면도, 평행이(가) 표시된 사진

자동 생성된 설명

Trouble Ticket Domain Model

System Sequence Diagram

1. Login SSD

먼저 사용자는 로그인을 시도합니다.

만약 사용자가 입력한 값이 LoginUi가 알고 있는 파일에 저장된 값이라면

Main UI를 호출하고 LoginUI는 종료됩니다.

그렇지 않다면 Loop Frame에 명시된것과 같이 루프를 돌아 사용자로부터 입력을 다시 받습니다.

1. New Issue SSD

사용자가 로그인에 성공한 상태를 가정하고 시퀀스가 진행됩니다.

사용자가 new Issue라는 버튼을 누른다면 Main UI는 new IssueUI를 호출합니다.

사용자는 이 UI의 비어있는 필드를 채우게 되고, 저장 버튼을 누르는 순간 새로운 이슈가 생성이 되고 테이블은 리프레시가 되어 다시 제어가 mainUI로 돌아옵니다.

New issue UI는 저장을 하고 나서 종료가 됩니다.

1. Edit Issue SSD

역시 사용자가 로그인에 성공한 상태를 가정하고 시퀀스를 진행합니다.

사용자가 텍스트 박스에 있는 특정 이슈를 클릭합니다.

MainUI는 해당하는 이슈의 티켓 번호를 Edit Issue UI에 전달하고

Edit issue UI는 해당하는 티켓 번호에 해당하는 파일을 열어 정해진 필드에 값을 배치합니다.

사용자는 변경을 희망하는 필드의 값을 변경하고 저장버튼을 누릅니다.

그러면 EditIssue UI는 해당 값을 저장하고 종료됩니다.

사용자는 다시 R버튼을 눌러 리프레시를 하고 이후의 동작을 진행합니다.

Use Case Diagram

유스 케이스의 경우에는 사용자의 행동을 표현 한 것이기 때문에 오히려 설명보다 다이어그램이 직관성을 가진다고 생각합니다.

이에 다이어그램에 대한 간단한 설명만을 덧붙이겠습니다. 상당수의 행위가 로그인 이후에 이루어지기에 로그인을 가정한 것도, 로그인을 가정하지 않은 것도 있습니다.

1. Check Other User UCD

다른 사용자를 조회하는 다이어그램입니다.

1. Delete UCD

이슈를 제거하는 다이어그램입니다.

1. EditIssues UCD

이슈를 수정하는 다이어그램입니다. 로그인에 성공하는 것을 가정하였습니다.

1. Edit status UCD

상태를 수정하는 다이어그램입니다. 역시 로그인에 성공하는 것을 가정하였습니다.

1. Login UCD

로그인을 시도하는 다이어그램입니다. Register 기능을 포함합니다.

1. LogOut UCD

로그아웃을 하는 다이어그램입니다. 로그아웃은 로그인을 호출합니다.

1. New Issue UCD입니다.

새 이슈를 생성하는 다이어그램입니다.

1. RefreshTable UCD

테이블에 변화가 있을시 테이블을 새로고침하는 경우의 다이어그램입니다.

1. Search UCD

사용자가 특정 데이터를 포함한(티켓번호, Assignee, Status등) 이슈를 조회하기 위해 검색을 하는 경우의 다이어그램입니다.

1. Total UCD

사용자의 자율적인 행동을 가정한 다이어그램입니다. 프로그램 로직이 전체적으로 표현되어 있습니다. Include extend 관계가 이곳에 표현되어 있습니다.

도표, 그림, 스케치, 텍스트이(가) 표시된 사진

자동 생성된 설명

<Total UCD>

Junit test

unitTest 디렉토리에 있는 textboxTest 파일은 Junit을 사용하여 유닛 테스트를 진행하는 테스트 파일입니다.

먼저 테스트를 진행하기에 앞서 임시로 디렉토리를 만들고, textbox 객체를 만듭니다.

테스트가 끝난 후, 생성된 파일들과 디렉토리를 제거합니다.

테스트 코드입니다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

String data에 있는 문장으로 newIssue를 생성해보고

실제 디렉토리에 필드에 해당하는 라벨을 가진 데이터가 생성되었는지를 검사합니다.

이경우 코멘트를 넣지 않은 테스트 코드이기에 별도의 코멘트가 없습니다.

코멘트를 추가한 경우 같은 파일 내에 있는 testNewIssueWithComments 테스트 코드를 통해 테스트 해볼수 있습니다.

모든 클래스에 대해 테스트 코드를 작성해보려 하였으나 시간상 가장 중요한 textbox 클래스에 대해서만 테스트 코드를 작성할 수밖에 없었습니다.

사실 프로그램 런타임 도중에 다양한 테스트를 시도할수 있기 때문에 테스트 코드의 중요성에 대해서 아직 크게 실감하지는 못했습니다.

실제 편의성 면에서 프로그램 내부에서 테스트하는 것 대비 제가 작성한 코드가 떨어지는 것 때문이라고도 생각 합니다.

이상으로 보고서 작성을 마무리합니다.

긴 문서 읽으시느라 고생 많으셨습니다. 감사합니다

작성자: 20203758 정찬혁