Term-project #2

과목명	오픈소스SW프로젝트
제출일	2023.06.08.(목)
팀 명	TEAM 20
팀 원	김지민(20204461)
	장민호(20194668)
	조언욱(20193933)
	차현호(20183968)

Ⅰ. GitHub repository

프로젝트 깃허브 주소 : https://github.com/advicewook/FileBrowser/tree/master

Ⅱ. Details about implementation

1. 주요 의존성 및 오픈소스

(1) JGit : Eclipse Foundation에서 관리하는 Java의 Git 라이브러리이다.

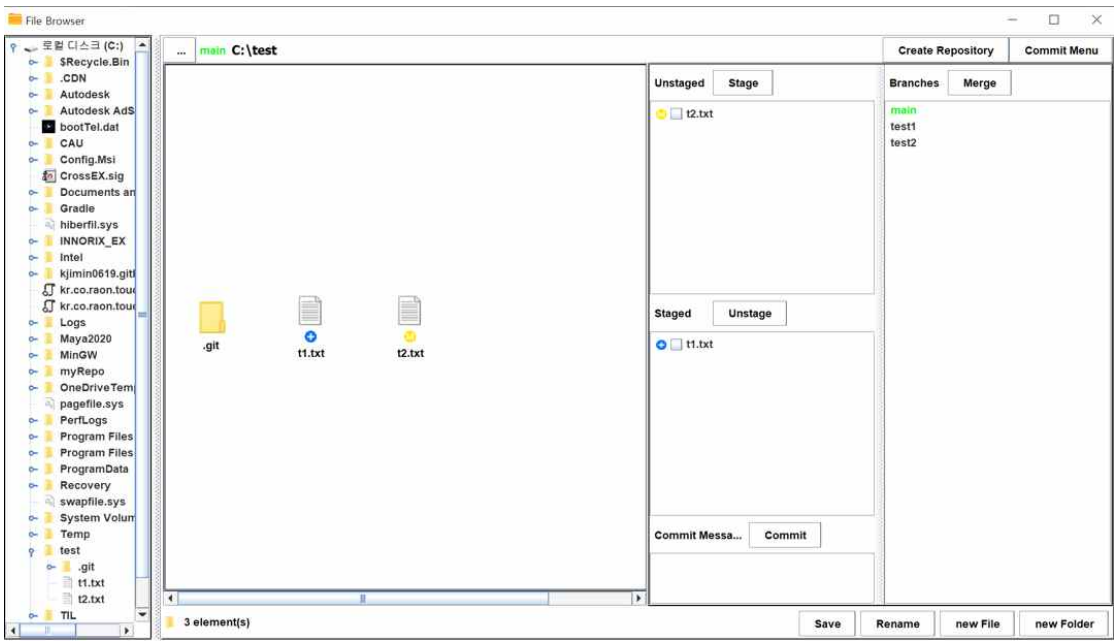
(2) Swing : Java로 작성된 GUI 라이브러리로 그래픽 인터페이스 구현을 위한 클래스를 제공한다.

(3) JGit Cookbook : JGit을 편하게 사용할 수 있도록 example 및 code snippet을 제공하는 오픈소스이다.

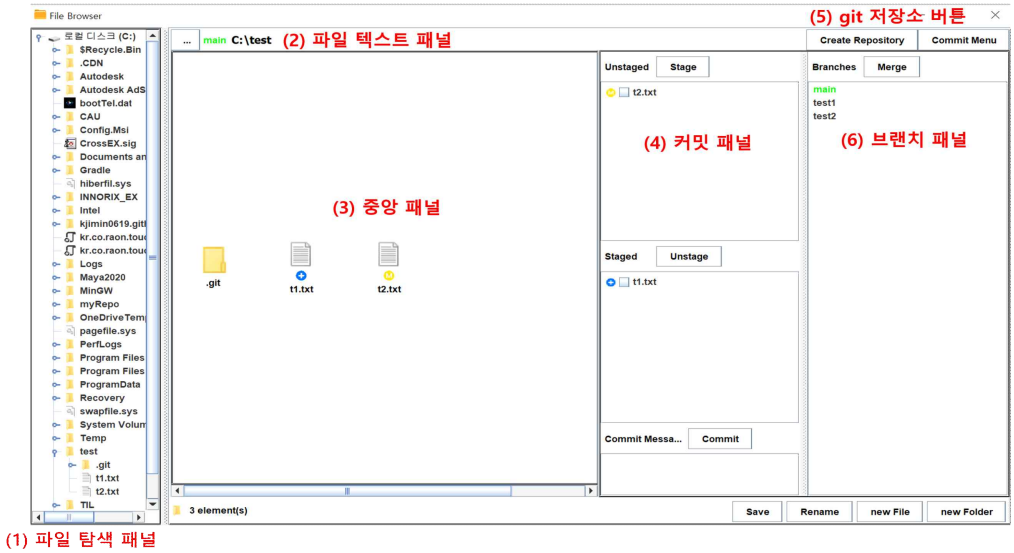
(4) FileBrowser : Java Swing을 활용한 파일 브라우징 오픈소스로, Eclipse Public License 2.0을 채택하여 소스 코드의 복제, 배포, 수정이 자유롭다. 단, 소스 코드 내 명시된 라이선스 정보를 그대로 유지한 상태에서 재배포해야 한다. 따라서 본 프로젝트에서는 Eclipse Public License 2.0을 적용한다.

2. Graphical Design

전체 GUI 구성은 다음과 같다.



<그림 1> file browser GUI



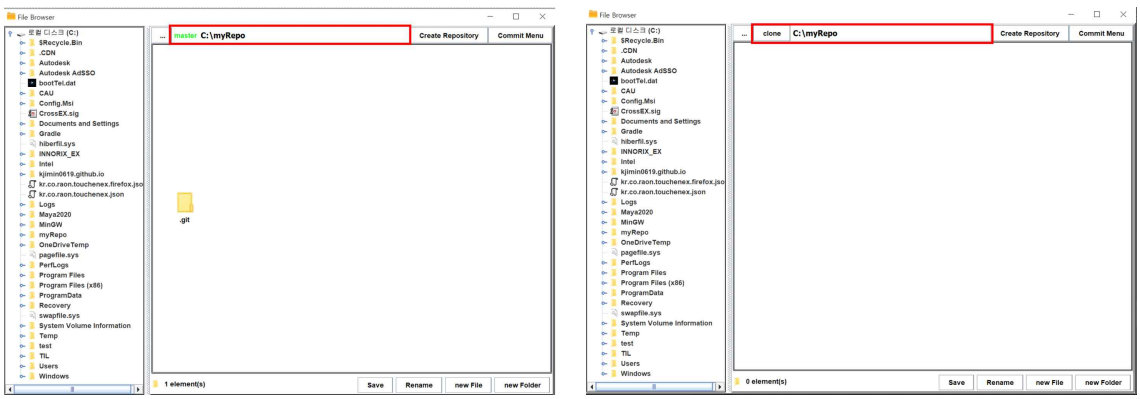
<그림 2> GUI 구조

(1) 파일 탐색 패널

- 컴퓨터에 존재하는 파일 및 폴더를 탐색하는 패널이다. 더블클릭으로 원하는 폴더에 접근할 수 있다.

(2) 파일 텍스트 패널

- git repository의 경우 폴더의 경로와 함께 현재의 브랜치 이름을 제공한다.
- git repository가 아닌 경우 git clone을 지원하는 clone 버튼이 생성된다.



<그림 3> git repository일 때(좌)와 git repository가 아닐 때(우) gui 차이

(3) 중앙 패널

- 폴더의 내부 요소들을 보여준다. 선택된 폴더가 git repository라면 파일의 상태 정보(아이콘)을 함께 제공한다. 이때 local의 현재 상황만 반영하기 때문에 파일의 상태를 중복으로 표시하지 않으며 하위 폴더에 대한 상태 정보는 제공하지 않는다.
- 마우스 우클릭을 통해 파일의 현재 상태별 수행 가능한 git 명령어 정보를 확인할 수 있다.

(4) 커밋 패널

- 주요 git 기능을 제공하는 패널로 git 저장소에서만 열 수 있다.
- *중앙 패널*과 마찬가지로 파일의 상태 정보를 제공한다. 이때 파일의 상태는 크게 *staged*와 *unstaged*로 구분하며 *untracked, modified* 등 세부 상태는 아이콘으로 표시한다.
- *unstage*, *staged* 버튼을 통해 특정 파일을 staging area에 올리거나, staging area에 있는 파일을 restore 할 수 있다.
- commit message를 작성하고 하단의 *Commit* 버튼을 통해 staging area에 있는 모든 파일을 *committed(unmodified)* 상태로 바꿀 수 있다.
- 한 파일에 대한 여러 상태 정보를 제공한다.

(5) git 저장소 버튼

- *Create Repository* 버튼으로 일반 폴더를 git repository로 만들 수 있다.
- *Commit Menu* 버튼으로 커밋 패널과 브랜치 패널을 오픈할 수 있다.

(6) 브랜치 패널

- git repository안의 모든 브랜치 리스트를 보여주는 패널로 git 저장소에서만 열 수 있다. 리스트에는 로컬 브랜치와 원격 브랜치의 모든 목록을 제공하며 원격 저장소의 브랜치인 경우, 원격 저장소의 별칭이 함께 제공된다(ex. origin/master).
- 단, *create repository* 버튼을 통해 repository를 생성한 경우, 초기 브랜치는 master(main)으로 지정되지만 브랜치 리스트는 확인할 수 없다. 사용자가 첫 커밋을 보낸 후에야 브랜치 패널이 업데이트 되고 브랜치 리스트 확인 및 메뉴 선택이 가능하다.

3. 주요 기능

Git Repository 관리 서비스를 제공하기 위해 추가된 기능을 소개한다.

3.1. 브랜치 관련 기능

(1) 브랜치 관리

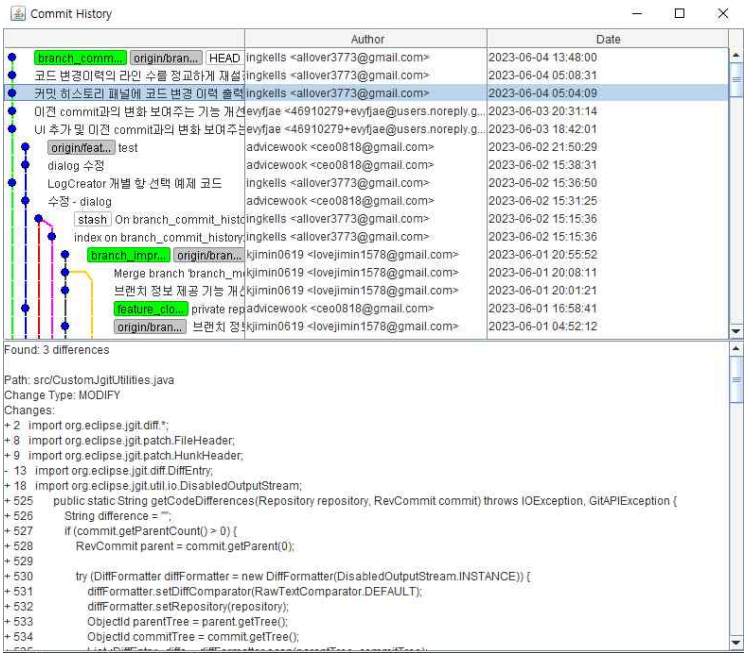
- 브랜치 패널 안 마우스 우클릭을 통해 브랜치 관리 메뉴를 선택할 수 있다. 메뉴에서는 *create, delete, rename, checkout* 기능을 제공하며, delete는 현재 브랜치에서는 불가능하다.
- 요청받은 메뉴를 수행할 수 없다면 에러 메시지를 띄운다.

(2) 브랜치 merge

- 브랜치 패널의 *merge* 버튼을 클릭하여 merge 가능한 브랜치 리스트를 확인하고 원하는 브랜치와 merge 할 수 있다.
- merge 성공 시 success 문구를 보여주며, 실패 시 error message와 unmerged path를 제공한 후 자동으로 abort 된다.

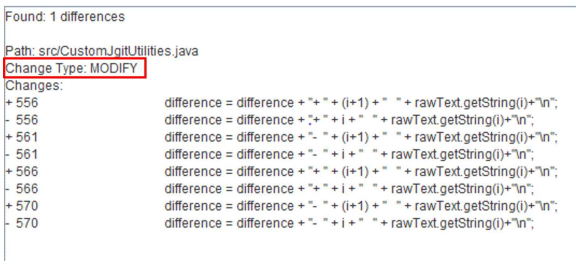
3.2. Commit History

- git repository에서 *Commit Menu* 버튼 클릭 시 commit history를 보여주는 팝업창이 나타난다.



<그림 4> commit history 그래프

- 기본적으로 commit message, author, 생성일자에 대한 정보를 확인할 수 있다.
- 그래프의 commit object를 클릭하면 해당 object에 저장된 변경사항을 확인할 수 있다. 변경사항은 크게 ADD, MODIFY, DELETE으로 구분한다. 이전 커밋과 비교했을 때 현 코드에 추가된 내용이 있거나(ADD), 코드 파일이 삭제되었거나(DELETE), 코드 내용에 변경사항이 있다면(MODIFY) 이에 대한 정보가 제공된다(<그림 5> 참고).
- merged된 commit object의 경우에는 현재 브랜치에서의 이전 커밋과의 변경사항을 제공한다.



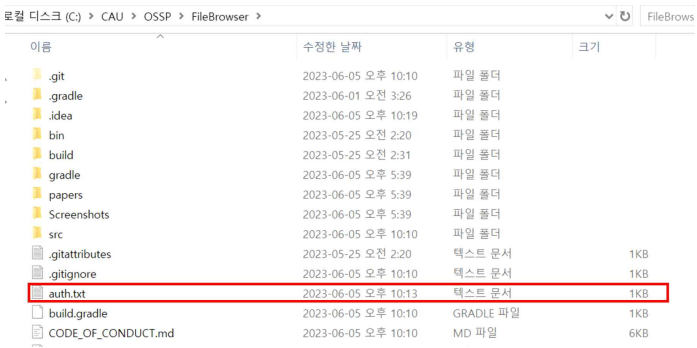
<그림 5> commit object 변경사항 예시 : MODIFY

3.3. Git clone from GitHub

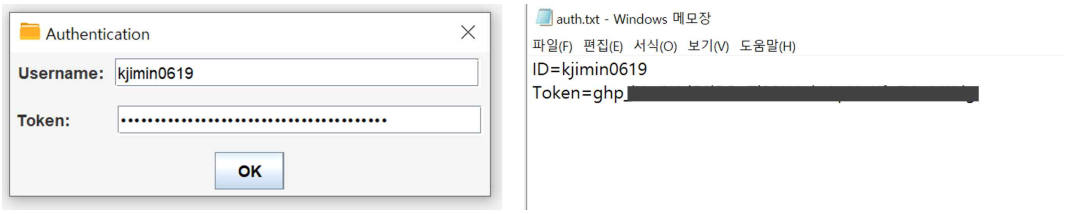
- *clone* 버튼을 통해 원격 저장소 안의 repository를 clone 할 수 있다. *clone* 버튼은 git repository가 아닌 폴더에서만 나타난다.
- 사용자는 clone 하고자 하는 repository의 address를 입력해야 한다.
- private repository를 clone하는 경우 사용자는 추가로 user ID과 token을 입력해야 한다.
- 사용자는 토큰 생성 시 접근 범위를 지정해야 한다. token이 없거나 유효하지 않은 경우, 사용자는 'github > Settings > Developer settings > Personal access tokens'에서 지정된 접근

범위의 access token을 생성해야 한다.(<그림 8> 참고).

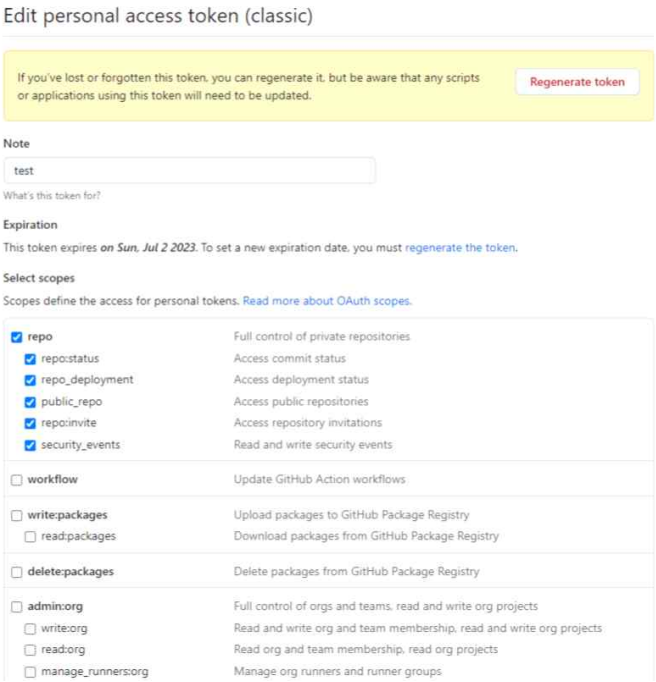
- private repository의 clone 성공 시 ID와 token을 추후에 입력 없이 사용할 수 있도록 프로젝트 폴더 하위의 auth.txt에 저장하고, 프로그램 재실행 시 사용한다.



<그림 6> auth.txt 파일



<그림 7> private repository clone시 authentication dialog(좌)와 auth.txt 내용 예시(우)



<그림 8> token 생성 시 필요한 access scopes

III . Guideline to install and run the service

본 프로젝트는 Windows를 타겟 플랫폼으로 하여 InteliJ IDEA Ultimate 환경에서 개발되었다. JAVA, JDK-17을 사용하고 gradle 8.0을 빌드 도구로 채택하였으며 이와 동일한 개발환경에서 실행하는 것을 권장한다.

프로젝트를 실행하는 순서는 다음과 같다.

1. Github Repository 복제
2. IntelliJ 환경 설정
3. Run

실행 과정에 대한 자세하고 구체적인 가이드라인은 깃허브 README 문서의 How to Execute에서 확인할 수 있다.

IV . Collaboration history

1. 협업 방식

- 노션을 활용한 회의록 작성 및 관련 문서 아카이빙
- 주 1회 오프라인 회의 진행

2. 역할 분담

	GUI	git 주요 기능 구현	기타
김지민	• branch 패널 전반적인 gui 구현 • 파일 텍스트 패널 branch 정보 제공		• 제출 문서 및 리드미 작성
장민호	• commit history graph 하단 패널 ui 생성	• branch merge 기능 구현	
조언욱	• clone gui 구현	• public, private repository clone 기능 구현 • 사용자 id, token 정보 저장 후 auth.txt 파일 생성	
차현호	• commit history graph 및 워크플로우 gui 구현	• branch create, delete, rename, checkout 기능 구현 • commit history 오브젝트의 author name, message, change 정보 출력	

<표 1> 역할 분담

3. 타임라인

날짜	진행 상황
23-05-26	branch 패널 UI 완성
	repository clone 기능 구현
23-05-27	repository clone 기능 개선 - private repository clone

23-05-29	branch merge 기능 구현
	branch create, delete, rename, checkout 기능 구현
23-05-31	commit history graph 초안 구현
23-06-01	branch 패널 기능 취합 및 개선
	repository clone 기능 개선 - 토큰 처리
23-06-03	commit history graph 추가 ui 구현
23-06-04	commit history graph 기능 구현
23-06-05	branch merge 기능 오류 개선 - 충돌 파일 경로 정보 제공, cancel 버튼 기능 추가
	repository clone 기능 개선 – 사용자 id, token 정보 파일(auth.txt) 생성
23-06-08	문서 및 리드미 작성

<표 2> 개발 타임라인

프로젝트 개발은 팀원별 브랜치를 만들어 기능을 구현하고 완료 후 merge하는 방식으로 진행되었다. 진행하는 과정에 있어 생성된 브랜치는 9개로 각 브랜치에 대한 설명은 아래와 같다.

- master : 배포 브랜치
- develop : 전체 기능 취합 브랜치
- feature_clone : repository clone 기능 구현을 위한 브랜치
- branch_commit_history : commit history graph 구현을 위한 브랜치
- branch_menu : branch 패널 gui 구현을 위한 브랜치
- branch_merge : branch merge 기능 구현을 위한 브랜치
- branch_management : branch create, delete, rename, checkout 기능 구현을 위한 브랜치
- branch_improvement : branch_menu, branch_merge, branch_management 브랜치 merge 후, 오류 및 기능 개선 작업을 위한 브랜치
- docs : 문서 작업을 위한 브랜치

4. git commit history

브랜치별 구체적인 작업 과정 및 전체 commit history는 아래 주소에 접속하여 확인할 수 있다.
https://github.com/advicewook/FileBrowser/commits/master