# 이클립스에서 Git 사용하기

(참고서적: 만들면서 배우는 Git GitHub 입문(6장)/윤웅식 저/한빛미디어)

덕성여자대학교 컴퓨터학과
최승훈

# 01. EGit 플러그인 설치

- EGit 플러그인 설치
  - 이클립스 > [Help] > [Eclipse Marketplace] > Egit 검색
  - 확인: Git 퍼스펙티브 클릭(활성화)

# 02. 저장소 생성

- 저장소 생성
  - 'Java' 퍼스펙티브 선택 > 'HelloWorld' 프로젝트 생성
  - 폴더 생성: C:₩Users₩csh₩git₩HelloWorld
  - 프로젝트명 > 팝업 메뉴 > Team > Share Project…
  - Configure Git Repository 창 > Create > Browse
    - C:₩Users₩csh₩git₩HelloWorld 선택
  - 확인:
    - 프로젝트명 옆의 아이콘 바뀌고 오른쪽에 저장소 정보 표시됨

# 02. 저장소 생성

- Git 관련 정보 확인
  - Git 퍼스펙티브 선택
  - Working Tree 선택
    - .git 저장소 디렉토리
    - Hello World 프로젝트 디렉토리

# 03. 첫번째 커밋

- HelloWorld.java 작성
  - System.out.println("Hello World");
  - 파일명 왼쪽 아이콘에 '?'
    - 아직 저장소에 추가되지 않았다는 의미

- 커밋
  - 프로젝트명 > 팝업 메뉴 > Team > Commit > 커밋 탭 열림
  - Commit Message 작성
    - create project. create hello world.
  - Unstaged Changes에서 모든 파일들을 Staged Changes로 추가
  - 'Commit' 버튼 클릭 > 파일명 왼쪽 아이콘의 '?' 사라짐

# 04. 새로운 브랜치 생성과 이동

- 새로운 브랜치 생성
  - 프로젝트명 > 팝업 메뉴 > Team > Switch To > New Branch
  - Branch name 입력
    - hotfix
  - 확인
    - 프로젝트명 오른쪽의 저장소 정보 변경됨

- HelloWorld.java 수정
  - 다음 줄 추가
    - System.out.println("World's End Dancehall");

# 05. 두 번째 커밋

- 두 번째 커밋
  - 프로젝트명 > 팝업 메뉴 > Team > commit
  - Commit Message 입력
    - add print line
  - Commit 버튼 클릭

# 06. master 브랜치와 병합

- 브랜치 변경
  - 프로젝트명 > 팝업 메뉴 > Team > Switch To > master
  - 확인
    - 프로젝트명 오른쪽 저장소 정보

- 브랜치 병합
  - 프로젝트명 > 팝업 메뉴 > Team > Merge
  - Merge ‘master’ 창
    - hotfix 브랜치 선택 > Merge 버튼 클릭

# 07. 각 브랜치의 독립성 확인

- master 브랜치 선택
  - HelloWorld.java 수정
    - 추가
      - System.out.println("Earth's End Dancehall");
  - 프로젝트명 > 팝업 메뉴 > Team > Commit
    - Commit Message 입력
      - master branch change a code
    - Staged Changes로 추가
    - Commit 버튼 클릭

# 07. 각 브랜치의 독립성 확인

- hotfix 브랜치 선택
  - HelloWorld.java 수정
    - 추가
      - System.out.println("Hello Dance World");
    - 프로젝트명 > 팝업 메뉴 > Team > Commit
    - Commit Message 입력
      - hotifx branch change a code
    - Staged Changes로 추가
    - Commit 버튼 클릭

- 각 브랜치는 독립적으로 코드 수정 가능함
  - 그러나, merge시 충돌이 발생하면 해결해야 함
    - 9절에서 실습

# 08. 불필요한 파일 및 폴더 무시

- .gitignore 파일
  - 저장소에 포함시키지 않을 파일 및 폴더를 지정할 때 사용함
  - 이미 저장소에 포함시키기 시작한 파일은 영향을 받지 않음
    - 저장소를 만들고 나서 바로 .gitignore 파일을 작성하는 것이 좋음
    - 저장소에 포함시켰던 파일을 무시하고 싶으면, 먼저 저장소에서 무시하고 싶은 파일을 모두 지우면 된다.
  - 예: *.class 파일을 저장소에 포함시키지 않으려면
    - 먼저 저장소에서 *.class 파일을 모두 지운 후
    - .gitignore 파일에 다음 문장을 추가함
      - *.class

# 08. 불필요한 파일 및 폴더 무시

- .gitignore 파일 생성
  - 프로젝트명 > 팝업 메뉴 > New > File
    - 파일명: .gitignore
  - https://www.gitignore.io/ 접속
    - windows, eclipse, java 입력 > 생성
    - 파일 내용을 .gitignore 에 복사 > 저장 > 커밋
  - 확인
    - Git 퍼스펙티브 선택 > Working Tree > HelloWorld 에서
      - .gitignore 확인
  - 테스트
    - Git 퍼스펙티브 선택 > Working Tree > HelloWorld > bin 에서 .class 파일 모두 삭제 > HelloWorld.java 수정 > 커밋
      - 이 때, *.class 파일은 unstaged changes 항목에 나타나지 않음

# 09. 충돌 해결

- master 브랜치 선택
  - 프로젝트명 >팝업 메뉴 > Merge > hotfix 브랜치 선택
    - 충돌 발생 메시지
  - HelloWorld.java 선택 > 팝업 메뉴 > Team > Merge Tool
    - 충돌이 발생한 두 파일의 비교를 도와줌
    - 이 탭을 아래에 위치시킨 후 참조하면서, HelloWorld.java에서의 충돌을 수동으로 해결함
    - 커밋

# 10. 기록 보기

- 커밋 기록 보기
  - 프로젝트명 > 팝업메뉴 > Team > Show in History
    - 커밋 id, 커밋 메시지, 커밋 메시지 작성자, 커밋 시간 등
    - 그 아래에는 커밋의 내역을 보여줌

# 11. 원격 저장소의 내용을 로컬 저장소로 가져오기

- 원격저장소를 클론하는 방법
  - 이클립스 > File > Import > Git 항목 > Projects from Git > Next > Clone URI > Next
    - GitHub의 저장소 URI를 입력함
      - 예: https://github.com/shchoi8347/seMainRepo01
  - Branch Selection 창: master 만 선택
  - Local Destination 창: 로컬 위치 지정
  - Select a wizard to use for importing projects 창
    - 새로운 프로젝트 선택
      - import using the New Project wizard
  - Select a wizard 창
    - 원격에서 가져올 프로젝트 유형 선택(Java Project 선택)
    - Project Folder Selection 창: Local Destination 창에서 지정한 로컬 위치 선택 > Finish

# 12. 로컬 저장소와 원격 저장소를 연결하기

- 빈 원격 저장소 생성하기
  - GitHub에서 빈 원격 저장소 하나 생성
    - 저장소 이름: eclipse_hello

- 원격 저장소와 연결
  - Git 퍼스펙티브 선택 > HelloWorld 폴더 > Remotes 팝업 메뉴 > Create Remote 선택
    - origin과 Configure push 확인 > create 클릭
  - Configure push for remote 'origin' 창 > Change 클릭
    - 원격 저장소 URI 입력 > Finish
      - 예: https://github.com/shchoi8347/eclipse_hello
    - Advanced... 버튼 클릭 > Add All Branches Spec 클릭 > Finish
  - Save and Push 클릭 > user name과 비밀번호 입력
  - GitHub에 가서 확인

# 13. 로컬 작업 내역을 원격 저장소에 올리기

- Hello World 프로젝트에 새로운 파일 생성
  - PushPrint.java의 main( )
    - System.out.println("Add Java Class File Push");
  - 저장 및 커밋
    - 커밋 메시지: remote repository add a PushPrint.java
  - 원격 저장소에 푸시
    - 프로젝트 팝업 메뉴 > Team > Push branch 'Master' > Preview > 사용자 아이디와 패스워드 입력 > Log in > Push

- GitHub에서 확인
  - PushPrint.java 가 추가되었는지 확인

# 14. 원격 저장소와 로컬 저장소의 간격 메꾸기

- 원격 저장소 수정
  - GitHub의 eclipse_hello에 접속
  - PushPrint.java 수정
    - 추가: System.out.println( " World with 42 " );
  - Commit changes
    - 커밋 메시지: PushPrint.java modified on GitHub
    - Commit changes 클릭

# 14. 원격 저장소와 로컬 저장소의 간격 메꾸기

- 로컬 저장소 수정
  - PushPrint.java 수정
    - 추가: System.out.println( " World with Miku " );
  - 커밋
    - 커밋 메시지: PushPrint.java modified on Local Repository

- 원격 저장소로 푸시 실행
  - 프로젝트 팝업메뉴 > Team > Push to Upstream
    - 실패 메시지 보임 (rejected)

- 원격 저장소로부터 Fetch 실행
  - Git 퍼스펙티브 > Remote > origin > Fetch

# 14. 원격 저장소와 로컬 저장소의 간격 메꾸기

- 로컬 저장소에서 병합 시도
  - 프로젝트 팝업 메뉴 > Team > Merge > origin/master 브랜치 선택
    - 충돌 발생 메시지
  - 충돌 해결 > 커밋

- 원격 저장소로 푸시
  - Git 퍼스펙티브 > 프로젝트 팝업메뉴 > Push Branch master > 사용자 이름과 패스워드 입력

- 원격 저장소에서 확인

# 14. 원격 저장소와 로컬 저장소의 간격 메꾸기

- 로컬 저장소에서 pull 사용
  - 목적: 원격 저장소의 변경 내용을 로컬에 직접 반영
  - 원격 저장소의 PushPrint.java 수정 및 커밋
    - 추가: System.out.println( " pull: remote 1 " );
  - 로컬 저장소
    - Git 퍼스펙티브 > 프로젝트 팝업 메뉴 > Pull 선택 > 풀 작업 결과 메시지 확인
    - PushPrint.java 수정 사항 확인