

Verziókezelés segédlet

Mi ez?

A verziókezelés olyan eljárások összessége, amelyek lehetővé teszik egy adathalmaz változatainak (verzióinak) együttes kezelését. Szoftverek esetében ez a szoftver életciklusa során a forráskódban végzett módosítások tárolását jelenti.

Miért van rá szükség?

A fejlesztés során a forráskód sok iteráción megy keresztül, így szükséges lehet, hogy esetleges probléma esetén vissza lehessen térni egy korábbi verzióra.

Egyszerű verziókövetés: minden változtatást elmentünk külön jegyzékekben. Megvalósítható, de nehézkes, időidényes, nem hatékony, nem átlátható. Megoldás: verziókövető rendszerek (pl. **Git**, Mercurial, stb.). Lehet használni helyi (local) illetve távoli (remote) repokat, ilyen távoli repok például a github, bitbucket stb.

A Git-et a használathoz telepíteni szükséges: <https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>

Hasznos GUI-val rendelkező Git kliens a **SmartGit**: <https://www.syntevo.com/smartgit/>

Alapfogalmak

Repository: röviden csak repo. Maga a tárolónk.

Working copy: A kód egy részének egy példánya, amelyen a fejlesztő éppen dolgozik a saját gépén

Commit: A kódon eszközölt változtatásokat úgynevezett commitok formájában érvényesíthetjük a tárolókon belül. A tárolók mintegy pillanatképként tartalmazzák azokat, illetve projektünk aktuális állapotát. Célszerű minden nagyobb módosítást követően commitolnunk. Az adott kommithoz általában megjegyzés is írható, hogy milyen módosítás történt az adott commit hatására.

Revision: verzió

Checkout: Lokális másolat készítése valamely verziókezelt fájlról.

Head: a legfrissebb commitot (verziót) jelöli, az aktuális ág teteje.

Push: adatok feltöltése a központi repoba

Pull: változások letöltése

Diff/Change/Delta: két file között változás megtalálása/mutatása.

Branch: fejlesztési ág

Merge: összefésülés. A fejlesztési ágak létrehozása mellett lehetőségünk van ezek egyesítésére is.

Conflict: Ágak összefésülése során keletkező jelenség. A két ág verziója olyan kódot tartalmaz, amit nem lehet automatikusan összefésülni

Clone: repo tartalmának lehozása lokálisan (adott jegyzékbe)

Forrás: Dr. Mileff Péter, Szoftverfejlesztés, Verziókövetés, Verziókövető rendszerek, jegyzet
https://users.iit.uni-miskolc.hu/~mileff/szf/Verziokezeles_V5.pdf

Github

<https://github.com/>

Távoli repository, amely a forráskódok távoli tárolását valósítja meg (részben ingyenes), használata regisztrációhoz kötött. Mérete folyton csak nő (forráskódból való törlés esetében is), hiszen minden egyes commit-olt verziót eltárol és bármelyik verzióra vissza lehet térni.

Regisztrációs felület:

Join GitHub

Create your account

Username *

Email address *

Password *

Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter.
[Learn more.](#)


Email preferences

☐ Send me occasional product updates, announcements, and offers.

Verify your account

Kérjük oldja meg a kirakóst hogy megtudjuk Ön valós személy-e

Ellenőrzés



Regisztráció után repo létrehozása szükséges, amely lehet **public vagy private**. Public esetében a weben bárki láthatja a repo tartalmát, private esetében csak az adott jogkörrel rendelkező személyek.

Új repo létrehozása:

Repositories



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner *



ttspeaker88 ▾

Repository name *

repo_neve



Great repository names are short and memorable. Need inspiration? How about [solid-spork?](#)

Description (optional)

repo leírása, mit fog tartalmazni



Public

Anyone on the internet can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

☐ Add a README file

This is where you can write a long description for your project. [Learn more.](#)

☐ Add .gitignore

Choose which files not to track from a list of templates. [Learn more.](#)

☐ Choose a license

A license tells others what they can and can't do with your code. [Learn more.](#)

Create repository

Létrehozott repo-k a felület bal oldalán jelennek meg:

Repositories



New

Find a repository...



[PasztorBence/vacation_calendar](#)



[nagydani98/Szakdolgozat](#)



[ttspeaker88/FRI_webpage](#)



[pipityu/thesis](#)



[bv-erika/OOP-Java-gyakorlat](#)



[ttspeaker88/sweng2018](#)



[ttspeaker88/oopJava](#)

Working with a team?

GitHub is built for collaboration. Set up an organization to improve the way your team works together, and get access to more features.

Create an organization

Az adott repo beállításai a repo-ra való kattintás után a Settings menüpont alatt találhatók:

ttspeaker88 / FRI_webpage Private

<> Code

🕒 Issues

🔗 Pull requests

🔄 Actions

📁 Projects

🛡️ Security

📈 Insights

⚙️ Settings

Options

Manage access

Security & analysis

Branches

Webhooks

Notifications

Integrations

Deploy keys

Actions

Secrets

Settings

Repository name

FRI_webpage

Rename

☐ **Template repository**

Template repositories let users generate new repositories with the same direc

Social preview

⚠️ You can upload a social image, but it will not be visible public

Upload an image to customize your repository's social media preview. Images should be at least 640×320px (1280×640px for best display).[Download template](#)

A repo típusa (public/private) az ablak alsó részében állítható (Change visibility):

Danger Zone

Change repository visibility

This repository is currently private.

Change visibility

Transfer ownership

Transfer this repository to another user or to an organization where you have the ability to create repositories.

Transfer

Archive this repository

Mark this repository as archived and read-only.

Archive this repository

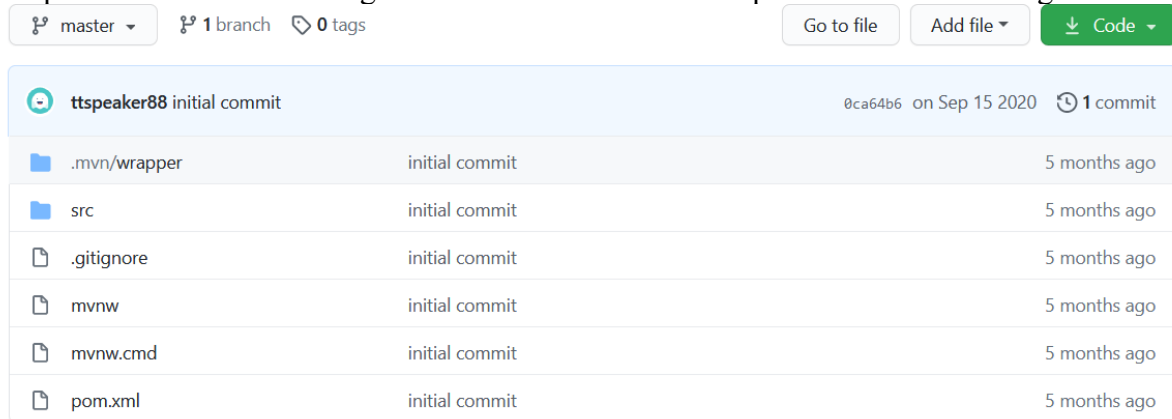
Delete this repository

Once you delete a repository, there is no going back. Please be certain.

Delete this repository

Manage access menüpont alatt (Invite a collaborator) lehet hozzáadni a private repo-hoz kollaborátorokat (résztvevőket), akik láthatják a repo tartalmát és commit-olhatnak is oda:

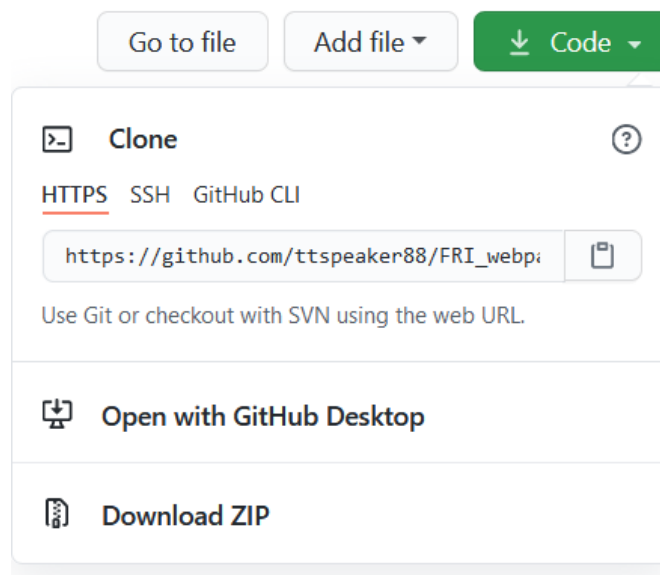
A repo tartalma és elérhetőségét biztosító link a Code menüpont alatt található meg:



The screenshot shows the top of a GitHub repository page. At the top, there are buttons for 'Go to file', 'Add file', and a green 'Code' button. Below this, a commit summary for 'ttspeaker88' is shown, including the commit hash '0ca64b6' and the date 'on Sep 15 2020'. A table lists the files in the repository:

File	Commit	Time
.mvn/wrapper	initial commit	5 months ago
src	initial commit	5 months ago
.gitignore	initial commit	5 months ago
mvnw	initial commit	5 months ago
mvnw.cmd	initial commit	5 months ago
pom.xml	initial commit	5 months ago

A zöld Code földre kattintva jelenik meg a link és itt tölthető le zip-elve a repo tartalma.



The screenshot shows the 'Code' dropdown menu. It has three tabs: 'HTTPS' (selected), 'SSH', and 'GitHub CLI'. The URL 'https://github.com/ttspeaker88/FRI_webp...' is displayed in a text box. Below the URL, it says 'Use Git or checkout with SVN using the web URL.' There are three main options: 'Clone' (with a terminal icon), 'Open with GitHub Desktop' (with a desktop icon), and 'Download ZIP' (with a ZIP file icon).

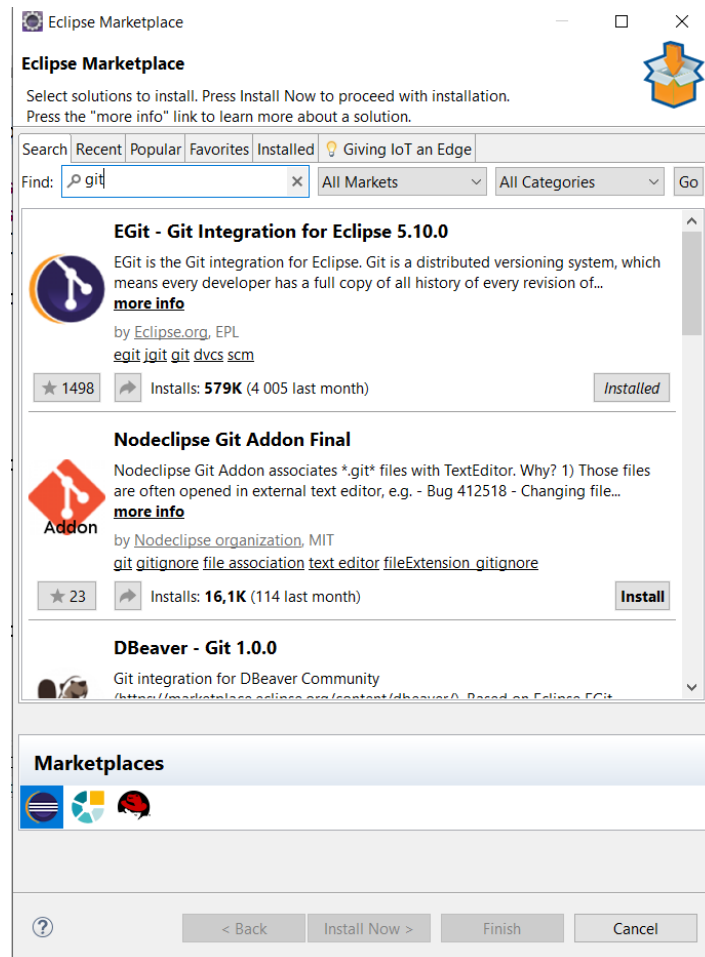
Hasznos Git tutorial: <https://www.tutorialspoint.com/git/index.htm>

Eclipse Git plugin telepítése és használata

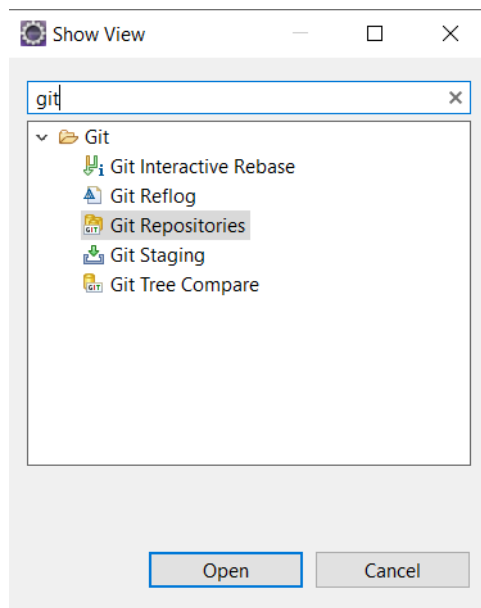
A Git plugin formájában telepíthető az adott IDE-be, amely által megvalósítja az adott IDE-ből (jelen esetben Eclipse) történő használatot.

Eclipse plugin: <https://www.eclipse.org/egit/>

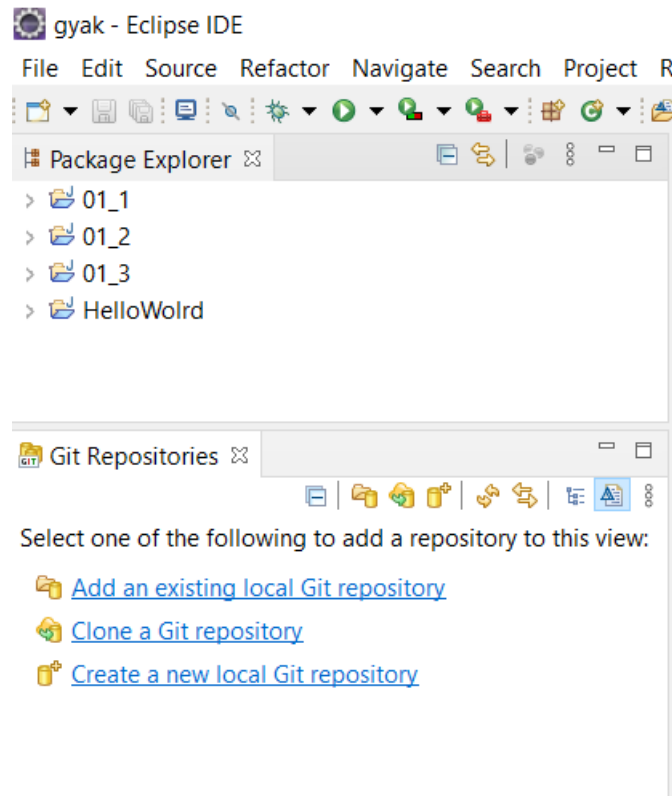
Telepítése Eclipse-ből: **Help -> Eclipse Marketplace -> Find -> git**



Telepítést követően meg kell nyitni a Git kezelőfelületét a **Window -> Show view -> Other -> Git -> Git Repositories** pontot kiválasztva:



Ezt követően az Eclipse ablakának bal oldalán a következő fülnek kell megnyílnia (Git repositories):



Itt az alábbi funkciók láthatók:

- *Add an existing local Git repository*: helyi fájlrendszeren lévő repo hozzáadása
- *Clone a Git repository*: távoli repo lehozása helyi fájlrendszerre
- *Create a new local Git repository*: lokális Git repo létrehozása a fájlrendszeren

Nekünk a projektek távoli repo-ba (Github) való felöltéséhez/leszedéséhez a *Clone a Git repository* opciót kell megnyitni, majd ezt követően hozzáadni az előző lépésekben létrehozott (Github) repo-t:

Clone Git Repository

Source Git Repository

Enter the location of the source repository.

Location

URI: Local Folder... Local Bundle File...

Host:

Repository path:

Connection

Protocol:

Port:

Authentication

User:

Password:

☐ Store in Secure Store

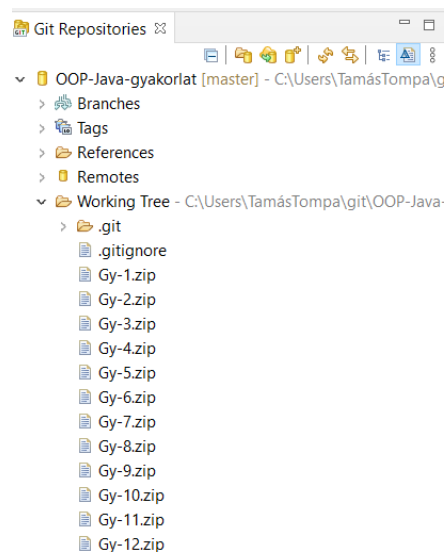
< Back Next > Finish Cancel

Az URI szöveges mezőbe kell beilleszteni a repo elérhetőségét, link-jét. Ennek hatására automatikusan felismeri, hogy milyen távoli repo-ról van szó (jelen esetben Github) majd kiegészíti a további *Host* és *Repository path* mezőket szükséges adatokkal.

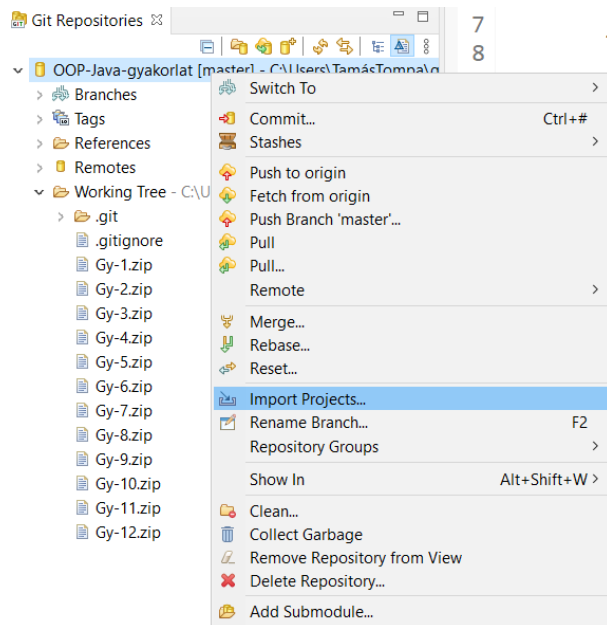
Az *Authentication* mezőben kell megadni az adott repo-hoz tartozó felhasználói név és jelszó párost (amit a Github regisztrációkor létrehoztunk).

A *Finish* gombra kattintva ennek hatására az adott jegyzékbe letölti a rendszer az adott repo tartalmát. Ez lehet akár üres is (kezdetben az), majd a későbbiek során ide fogjuk feltölteni (push) a forráskódot.

Ha a művelet sikeres volt akkor a Git ablakban az alábbiakat, azaz az adott repo tartalmát kell látni:

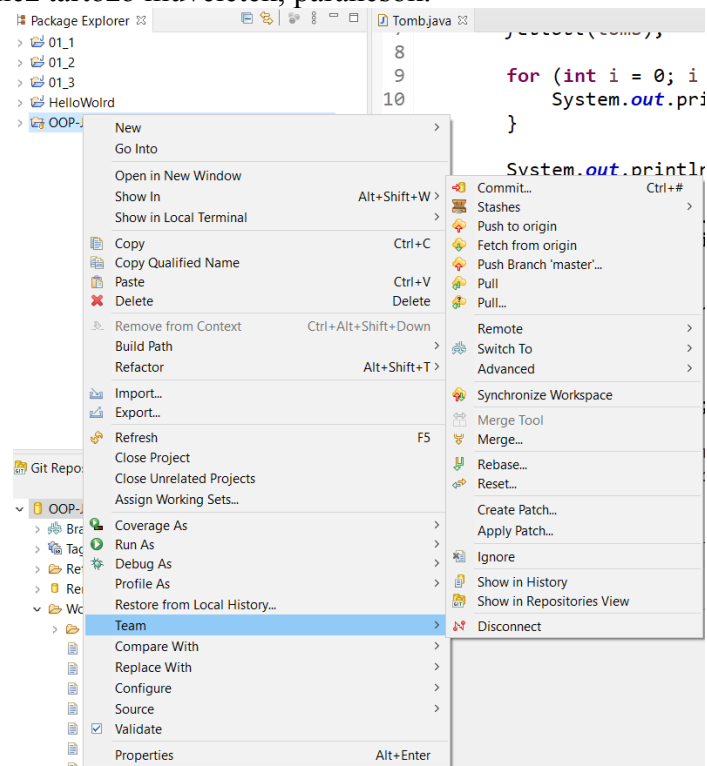


Jobb gombbal való kattintás hatására az *Import Projects* parancsot választva tudjuk a repo tartalmát importálni az Eclipse-be projektként. (Az Eclipse felismeri, hogy milyen projektről van szó és megnyitja az annak megfelelő nézetet).



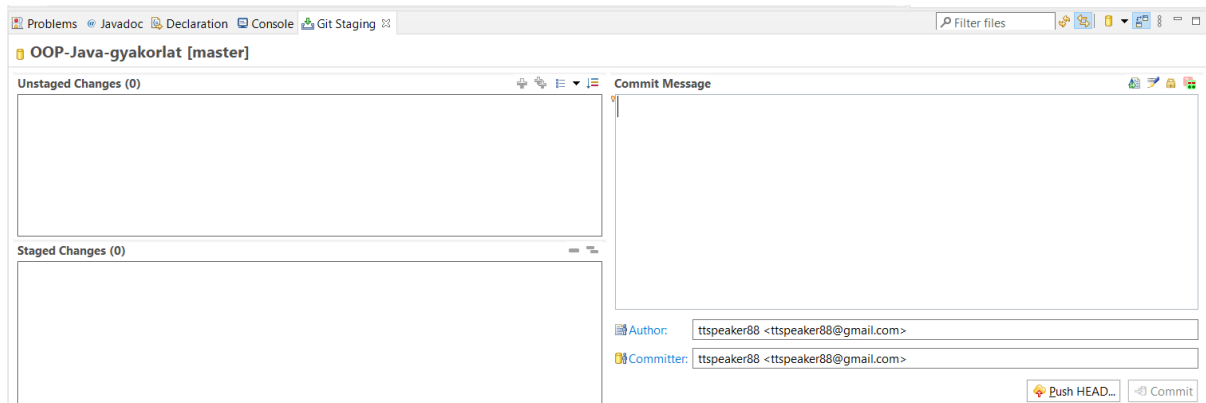
Ha sikeres volt a művelet, akkor az Eclipse *Package Explorer* ablakában megjelennek az importált (és így már futtatható) projektek.

Az adott projektre jobb gombbal kattintva, a legördülő menüből a *Team* menüpontot választva nyílnak meg a Git-hez tartozó műveletek, parancsok.



Ezek közül nekünk a *Commit*, *Pull* és *Push* műveletek kellenek elsősorban.

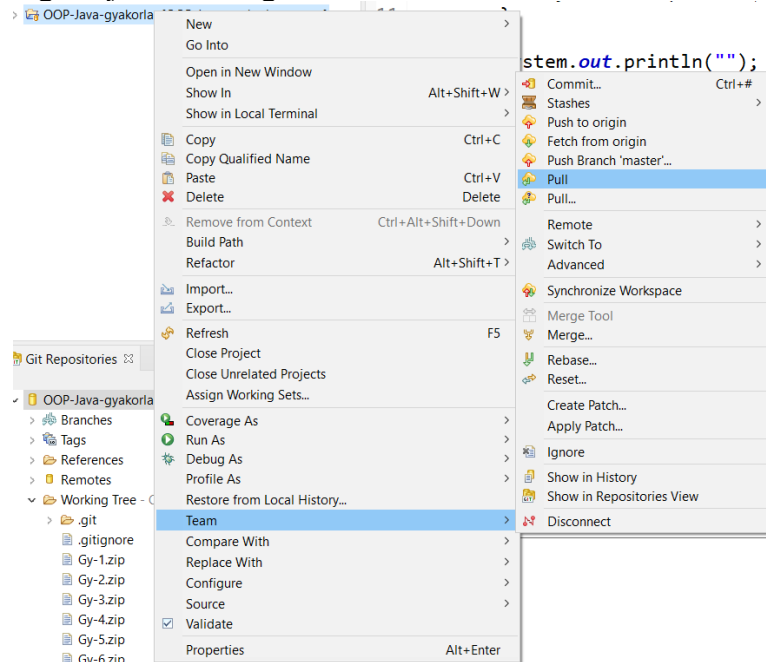
Commit kiadása a *push* (azaz a végzett módosítás feltöltése a repo-ba) művelet előtt szükséges. Ennek hatására a *Git Staging* fül fog megnyílni az Eclipse-ben.



Az *Unstaged Changes* részben az elvégzett, de még nem Commit-tolt módosítások láthatóak (jelen esetben most ez 0, mert nem volt módosítás).

A *Commit Message* szövegmezőben az adott kommithoz tartozó leírása megadása szükséges, tehát, hogy milyen módosítás történt az adott kódrészleten, például: „Emberek osztály hozzáadása”. Ennek hatására még nem történik feltöltés a távoli repo-ba, ez a módosítás (és commit) még csak a lokális fájlrendszeren él. Ahhoz, hogy a végzett módosítás ténylegesen felmásolódjon a repo-ba is a *Push/Push HEAD* művelet végrehajtása szükséges.

Ha valaki más által, az adott projekten végzett módosításokat szeretnénk letölteni a saját fájlrendszerünkre, az aktuális és legfrissebb verzióra frissítve a forráskódot, akkor a *Pull* művelet kiadása/végrehajtása szükséges:



A forráskódon történő munka előtt minden esetben célszerű végrehajtani a *Pull* műveletet, hogy a munka megkezdése előtt mindig a legújabb verzió legyen nálunk. (ha ugyanazon a kódrészen végzek módosítást, amelyen előtte már valaki és nem hozom le a legfrissebb verziót akkor az általam végzett módosítás felül akarja írni a más által végzett módosítást, ami ellentmondáshoz *Conflict*-hez vezet.)

A SmartGit segítségével GUI-val rendelkező felületről tudjuk kezelni repo-inkat. Az alkalmazás felülete:

