Rondje-hoe gaat het met Git?

- Hoe gaat het tot nog toe/ervaringen/fijn/lastig?
- Wat werkt wel?
- Wat werkt (nog) niet?

Leerdoelen (verversen/versterken)

- Repository kunnen maken en pushen
- Branches kunnen maken en deleten
- Kunnen klonen, fetchen, pushen, pullen, mergen – en het verschil daartussen weten
- Merge conflicten kunnen oplossen
- Fouten ongedaan maken (reset en revert)
- ONDERSTEUNEND:
  - Weten wat een commit, branch, staging en .gitignore zijn
  - Weten wat git trackt/in de gaten houdt

Maar eerst...

-IntelliJ kan ongeveer alles
-Maar is daarom niet overal erg goed in
-Heb reserve-tool
https://git-scm.com/downloads

-HANDIG: Git-gespecialiseerde GUI
https://git-scm.com/downloads/guis
Aanbevolen:
-https://gitextensions.github.io/ (Git Extensions (niet voor Mac! :( ))
-https://www.sourcetreeapp.com/ (SourceTree)
-https://www.syntevo.com/smartgit/ (SmartGit)

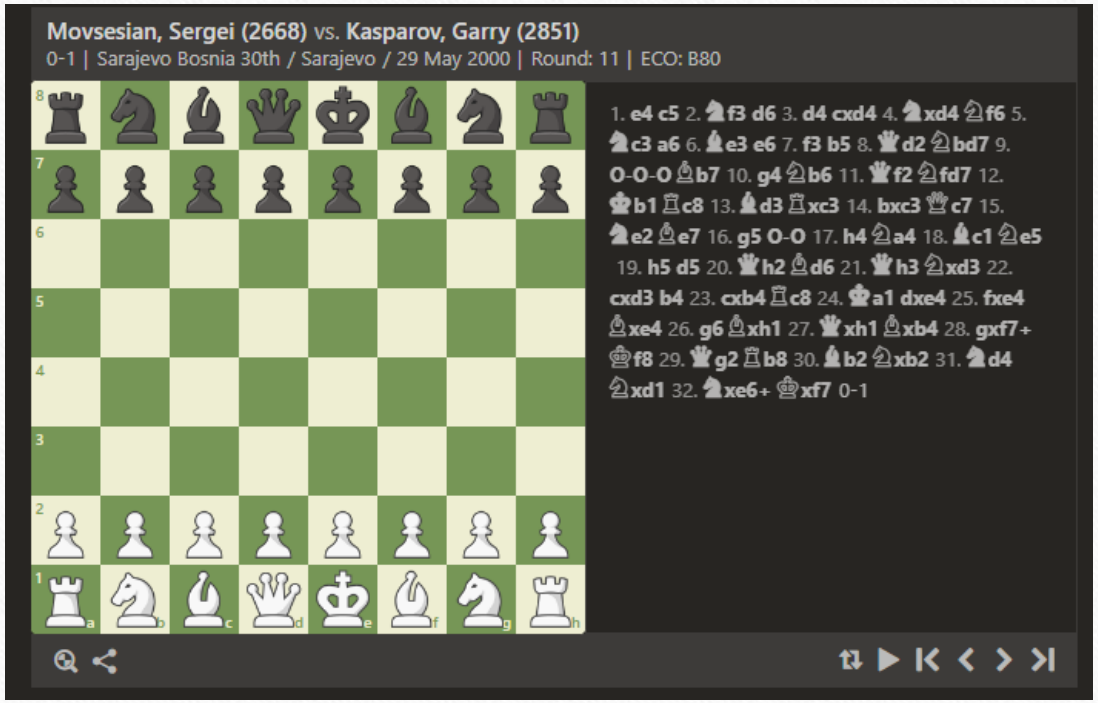
Ook omdat je commits checken handig is

```
private static List<Token> GetAncestors(ParsePosition position){
    List<Token> implemented = new ArrayList<>();
    if (position.CurrentTokenType() == TokenType.Colon) {
        position.Proceed();
        do {
            TokenType currentTokenType = position.CurrentTokenType();
            if (currentTokenType == TokenType.Identifier) implemented.Add(position.CurrentToken());
            else if (currentTokenType == TokenType.BracesOpen) break;
        } while (true);
    }
    return implemented;
}
```

Even stilstaan… waarom git?

- Leven zonder Git?

Git is vergelijkbaar met schaaknotatie



Schaken: Elk tussenbord kan worden afgeleid
Zonder 40x een schaakbord op te slaan
Git: elke tussenversie kan worden afgeleid
Zonder 40x alle bestanden op te slaan

Git kan makkelijk extra informatie toevoegen
1. e2e4 c7c5
Zo kan je ook terugzoeken van huidige toestand ipv telkens vanaf begin!

https://www.chess.com/openings/Sicilian-Defense

Nog even: hoe werkt Git?

- Git gebruikt
  - Een directory met 'normale' code ('huidige toestand')
  - En wat specifieke git-bestanden (bv .gitignore)
  - In de normale directory zit verborgen .git directory
    - Compact, want alleen changes worden opgeslagen
    - En alles gezipt!
  - Vaak een standaardserver waar team naar pusht
    - Heet "remote" of "origin"

Een commit is een bestandje met 4 dingen:

- De wijzigingen (regels, tekst)
- Een commit message (dus verplicht!)
- 0..n referenties naar de vorige commit
  - 0 voor eerste commit
  - 1 voor normale commit
  - 2 (of meer!) voor merge
- Heeft hash (als identifier)

En wat is een branch?

- Een pointer naar een commit
- Heeft een naam
- Kan worden geupdate om te wijzen naar een andere commit
  - Als dat niet kan heet het een "tag"
- "in een commit" waar geen branch naar wijst? 'Headless'

Git houdt verder in de gaten

- ...welke files getrackt zijn (interne lijst Git, niet makkelijk te zien)
- Welke files niet gevolgd hoeven worden (staan in .gitignore)
- "Op" welke branch je momenteel zit [OF welke commit de actuele is]

De spelers

- Commits
- Branches
- Locals (huisvesten .git repos)
- De origin/remote
- git zelf met .gitignore, tracked, active commit/branch
- Gebruiker

Het Git-proces

- Met clone of pull haal je de zipfile van de server
- Je wijzigt de code
- Git analyseert verandering in files
    - Althans, van files die niet in .gitignore staan

1] STAGE: Je selecteert welke (gewijzigde) files in de 'commit' komen

- Dit heet 'staging'

2] COMMIT: Git slaat de wijzigingen in de files op in lokale repository

- 'committing' - ook commit-boodschap nodig

3] PUSH: Git kan met 'push' commit naar centrale repository sturen

Het Git-proces: wat gebeurt er als je een branch maakt?

1. Git checkt wat de actieve branch is (AB) - zie .git/HEAD
2. Git checkt naar welke commit AB verwijst (Current Commit, CC)
3. Git maakt een nieuwe branch aan (NB)
4. Git zorgt dat NB nu naar de current commit gaat wijzen.
5. *Optioneel: maakt NB de actieve branch*

Laten we gaan gitten!

Troubleshooting/misc

- Troep in je repository (git rm –cached)
- De Detached head state...
- Amend
- Cherrypicking (ik heb in de verkeerde branch gecommit!)
- Rebase, Squash
- Zit je vast? Gebruik de commandline. Of een gespecialiseerde git-tool

Handigheidjes

- Als je klaar bent voor de dag: maak een commit!
- Als je klaar bent met een feature:
  - *Pull main
  - *Merge main in jouw branch (los conflicten op) (*=moet alleen bij meerdere branches tegelijkertijd)
  - Merge jouw branch in main
  - Push main/maak pull request
- Als je een nieuwe feature wil maken (of wil experimenteren)
  - Maak een branch! EN SWITCH ERNAARTOE

Referenties

- http://think-like-a-git.net/
- https://www.youtube.com/watch?v=1ffBJ4sVUb4 Git for people age 4 and up (video)
- https://www.atlassian.com/git/tutorials
- Azure devops-flow https://jd-bots.com/2021/01/23/push-your-code-to-azure-devops-repository-from-visual-studio/