Tehnici Avansate de Programare

Modularizare, `git` și evaluare

Petru Rebeja, Marius Apetrii

4 Martie 2021

Facultatea de Matematică
Universitatea Alexandru Ioan Cuza, Iași

Introducere

Informații

Avem spațiu dedicat cursului pe Discord.

Recapitulare

Recapitulare

- Tipuri de date în `.net`

Recapitulare

- Tipuri de date în `.net`
- Principiile programării orientată-obiect

Despre ce vom discuta azi

- Clase abstracte vs. interfețe.
- Acuplare și Coeziune.
- Fluxul de lucru GitHub
- Evaluare

Clase abstracte vs. Interfețe

Clasa abstractă

- Oferă o implementare implicită pentru metode și proprietăți.
- Nu se pot crea instanțe ale claselor abstracte.
- Este mai puțin abstractă decât înterfața.
- Poate impune anumite constrângeri (ex. constructorul).
- Un tip de date poate deriva dintr-o singură clasă abstractă.

Interfața

- Nu oferă nicio implementare.
- Cel mai mare grad de abstractizare.
- Nu impune constrângeri decât asupra semnăturii metodelor.
- Un tip de date poate implementa mai multe interfețe.

Modularizarea codului-sursă

Acuplare și Coeziune

Două atribute foarte importante ale unui produs software de succes sunt:

- Grad mic de acuplare,
- Grad mare de coeziune.

Acuplarea

Acuplarea este o măsură a gradului de interdependență dintre modulele unui produs software[1].

[1] https://en.wikipedia.org/wiki/Coupling_(computer_programming)

Acuplarea — exemplu

Implementarea clasică a șablonului `Singleton` este un exemplu de grad înalt de acuplare: clasele care folosesc metodele definite de `Singleton` sunt dependente de acesta.

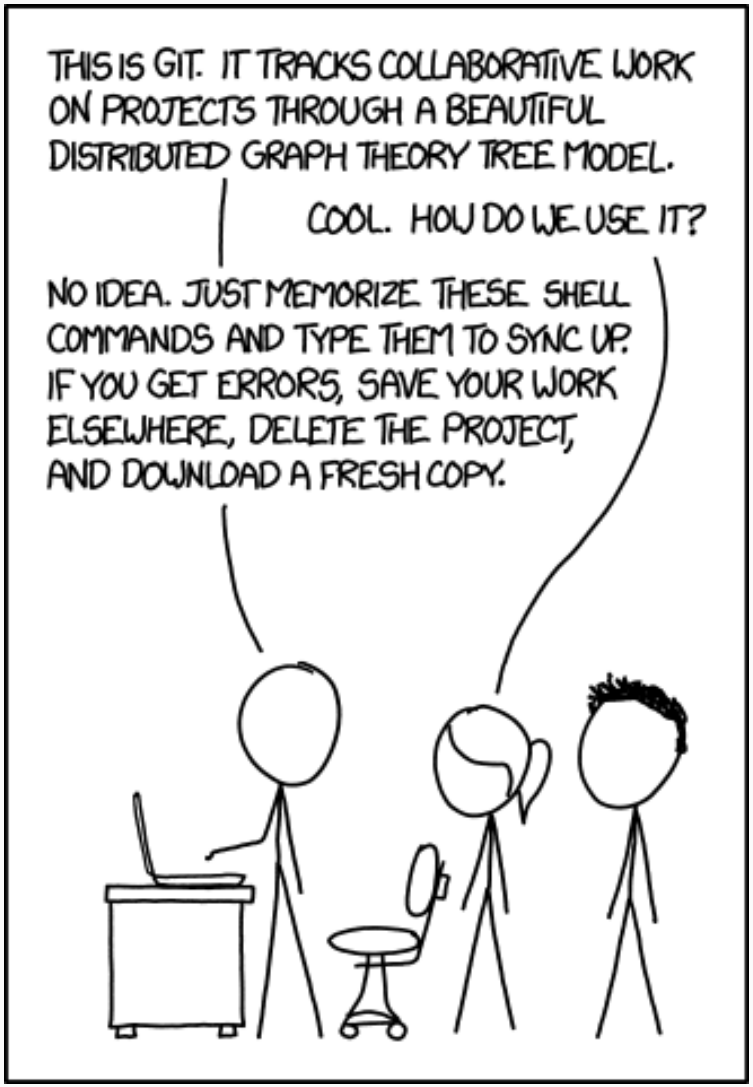
Coeziunea

Coeziunea este măsura în care elementele unui modul aparțin unul de celălalt[2].

[2] https://en.wikipedia.org/wiki/Cohesion_(computer_science)

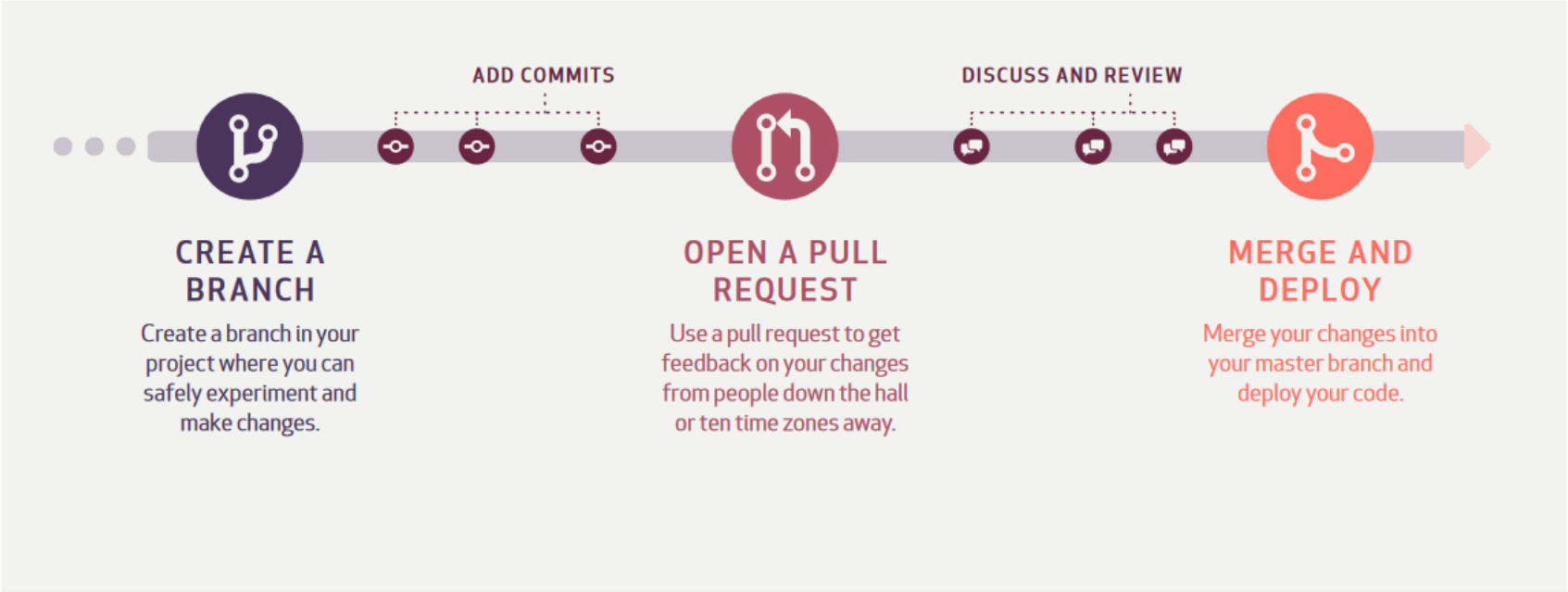
Fluxul de lucru Git

Git[3]



[3] https://xkcd.com/1597/

Fluxul de lucru GitHub[4]



[4] https://guides.github.com/introduction/flow/

Fluxul de lucru GitHub

Pentru proiectele la care aveți drepturi să faceți modificări:

- Creați o ramură nouă,
- Implementați cerințele prin modificări multiple și atomice,
- Creați un Pull Request din ramura nouă,
- Revizuiți și modificați dacă este cazul,
- Îmbinați ramura nouă cu `main`,
- Livrați funcționalitatea adăugată.

Fluxul de lucru extins[5]

Pentru proiectele la care nu aveți dreptul să faceți modificări:

- Creați un `fork` al proiectului,
- Adăugați modificările necesare în `fork`-ul propriu (folosing fluxul de lucru de mai sus),
- Creați un `Pull Request` din `fork`-ul propriu,
- Adăugați modificări suplimentare dacă este cazul
- O persoană desemnată va decide dacă modificările vor fi acceptate sau nu.

[5] `https://guides.github.com/activities/forking/`

Descrierea modificărilor[6]

[6] https://xkcd.com/1296/

Ghid pentru descrierea modificărilor[7]

- Lăsați o linie goală între sumar și restul descrierii.
- Sumarul trebuie să aibă maxim 50 caractere.
- Scrieți conform regulilor de gramatică și ortografie:
  - Începeți propozițiile cu majusculă,
  - Nu puneți punct după sumar.
- Descrierea trebuie să explice ce și de ce a fost modificat; nu *cum*.

[7] `https://chris.beams.io/posts/git-commit/`

Evaluare

Întrebări?

Întrebări legate de prima evaluare.

Încheiere

Recapitulare

- Clase abstracte vs. interfețe.
- Acuplare și Coeziune.
- Fluxul de lucru GitHub
- Evaluare

Vă mulțumesc

Succes la evaluare!