Java és web programozás

Kovács Kristóf

Budapesti Műszaki Egyetem

2015. 05. 06.

10. Előadás

Tiszta kód

- Az előadás az olvasható és tiszta kód írásáról fog szólni.
- Miért érdemes olvasható kódot írni?
  - A nagyvilágban nem egyedül írják az emberek a programokat, hanem csapatokban. Nagyon frusztráló tud lenni, amikor órákat kell azzal tölteni, hogy valakinek a kódját megértsd.
  - Egyszer kész lesz a kód ekkor az ember örül és elteszi, hogy soha többé ne kelljen hozzányulni. De mi van akkor ha mégis hozzá kell? Fájdalmas lesz, ha már egy év eltelt a letétel óta.
  - Könnyebben észrevehetők a hibák.
- Mikor nem érdemes tiszta kódot írni?
  - Ha annyira a haladás útjába áll, hogy több idő megy el a szépítéssel, mint a valódi fejlesztéssel.
  - Ha a kód eldobandó, csak egy specifikus feladatra van írva.

Indentálás

- Nagy vita tud lenni, hogy melyik indentálási stílus a legjobb
- K&R

```
int main(int argc, char *argv[])
{
    while (x == y) {
        ...
    }
}
```

Indentálás

- Nagy vita tud lenni, hogy melyik indentálási stílus a legjobb

- Allman

```
int main(int argc, char *argv[])
{
    while (x == y)
    {
        ...
    }
}
```

Indentálás

- Nagy vita tud lenni, hogy melyik indentálási stílus a legjobb
- BSD KNF

```
int main(int argc, char *argv[]) {
    while (x == y) {
        ...
    }
}
```

Változó- és adattagnevek

- Egy változó vagy metódus **scope**-ja a kód azon része ahol ez a változó vagy metódus előkerülhet.
- Ha egy változó scope-ja kicsi, azaz csak rövid ideig használjuk, akkor a neve legyen rövid. Például ciklusváltozók, ideiglenes változók. Ezek a változók általában egyértelműk, hogy mire használtak, így nem kell leíró név.
- Ha viszont egy változó scope-ja nagy, akkor minél leírobb nevet próbáljunk neki adni
- A változók neve legyen mindig főnév, kivétel ha igaz-hamis változó, ebben az esetben melléknév.
- Ne tartalmazzon semmit ami a típusára utal, ez felesleges, a fejlesztői környezet tudatja velünk.

Függvény- és metódusnevek

- Ha egy függvény scope-ja kicsi, azaz csak nagyon kis körben használjuk, akkor a neve legyen hosszú. Ekkor általában speciális dolgot csinál amit átláthatóbb ha részletesen leírunk.
- Ha viszont egy függvény scope-ja nagy, akkor minél rövidebb nevet próbáljunk neki adni. Sok helyen fogjuk használni és nem a legszebb dolog amikor 50 karakteres függvényneveket kell végigolvasnunk.

- A függvények neve legyen többnyire ige, írja le, hogy mit csinál a függvény.
- Nem kell a visszatérési értékére vagy a paramétereire utalnia.
- Ha igaz-hamis értéket ad vissza, akkor legyen igen-nem-el megválaszolható kérdés.

Függvények, metódusok terjedelme

- A cél az, hogy a függvényeink legyenek minél rövidebbek, ha egy függvény 20 soros az biztosan olvashatatlan.
- 10 soros függvénynél még mindig gondolkodnunk kell, hogy megértsük.
- 4-5 sor lenne az ideális, ennél hosszabb ne legyen egy függvény.

- Ha egy függvényben van egy feltétel, ne habozzunk kiemelni a feltételen belüli programrészt és nevezni *eredetiFuggvenyNeveHaValamiteljesul* néven, átláthatóbb lesz a kód.

Interface-ek

- Használjuk az interface-eket és osztályokat arra amire valók.
- Rejtsük el a program többi része elől, hogy hogyan működik egy osztály.
- Ha egy osztály használ egy másikat, úgy hogy annak belső működését is kihasználja, akkor gondban lehetünk.

- Legyenek jól elhatárolható részei a programnak amiket az interface-ek mentél "el tudunk vágni". Így a kód külön feleivel kényelmesen foglalkozhatnak különböző emberek

Refactor

- Ha már látjuk, hogy a kódunk kezd egyre csúnyább lenni, hosszú függvények, érthetetlen változónevek, publikus adattagok, akkor refaktorálnunk kell.
- Eclipse-ben érdemes tanulmányozni a Refactor menüpontot. A következők a leglényegesebbek:
  - Extract method
  - Rename
  - Generate...