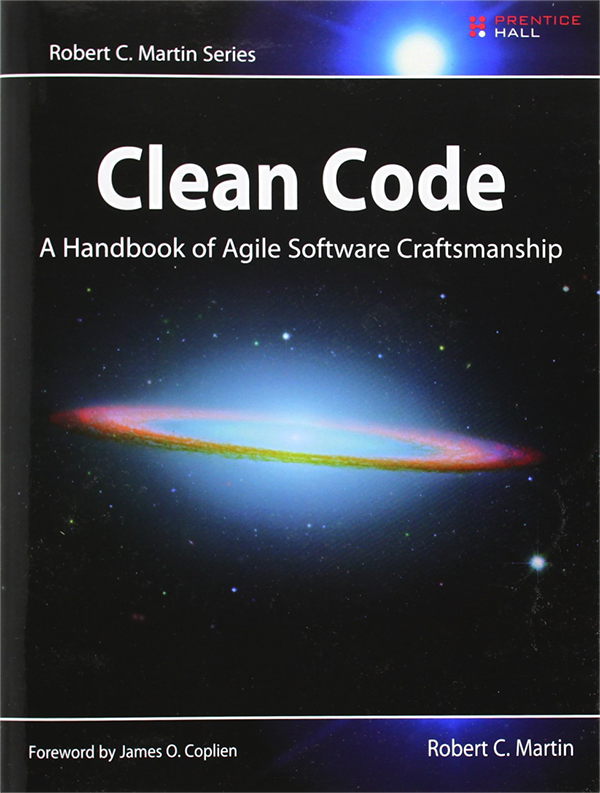
Ce înseamnă pentru mine conceputul de Clean Code

Să începem pentru prima dată cu modalitatea prin care am aflat pentru prima dată de conceptul de Clean Code. Acest concept l-am aflat destul de târziu aș putea spune, deoarece urmând un liceu de mate-info, în mod normal, unele concepte de bază, cum ar fi Clean Code, sunt printre primele lucruri cu care, profesorul ar trebuii să te familiarizeze, însă din păcate nu a fost așa... însă trecând cu vederea toate aceste lucruri, prima dată când am dat de acest concept a fost în primul meu an de facultate când trebuia să fac mici/mari proiecte, cu diferite grade de complexitate, unde chiar dacă totul mergea bine, era foarte greu de înțeles, câteodată de mine după ceva timp “ Oare de ce am declarat variabila asta?? “ dar mai ales de colegi.

Imaginează-ți o carte în care toate paragrafele sunt amestecate, mărimea caracterelor este diferită pentru fiecare dintre ele și o parte din ele au 20 de pagini. Cât de ușor ați putea citi această carte? Codul ar trebui să fie scris într-un fel în care să ofere oamenilor șansa de a-l citi ca pe o carte, de la început până la sfârșit, în care fiecare logică diferită este grupată separat.

**Sursa de învățare a unui scris *CLEAN***

Cred că este cea mai cunoscută și bună carte de *Clean Code* scrisă de **Robet C. Martin.** Prima dată când am văzut-o am zis, “oare ce m-ar putea învăța această carte cu sute și sute de pagini? “. Eii… chiar a avut cu ce mă învăța și cred că această carte trebuie să fie citită de orice persoană care dorește să ia contact cu tot ceea ce înseamnă programarea.

Pentru mine, recunosc a fost foarte greu să mă obișnuiesc cu scrierea unui cod curat, chiar și în momentul de față. Scrierea unui code curat este o muncă grea. Este nevoie de foarte mult exercițiu și de o foarte bună concentrare în momentul scrierii acesteia.

Cred că partea cea mai grea este pur și simplu de a face un obicei în a scrie clean code, dar odată cu trecerea timpului și îmbunătățirea abilitățiilor devine din ce în ce mai ușor. Scrierea unui cod curat, astfel încât și cele mai mici lucruri, cum ar fi schimbarea obiceiurilor pentru a numii variabilele fac cea mai mare diferență.

Caracteristicile unui cod curat:

* Ar trebuii să fie elegant – Codul curat ar trebuii să fie plăcut la citit. Citirea ar trebuii să facă cititorul să zică “Daa, uite, însfârșit un cod curat, unde pot să înțeleg tot ceea ce ai vrut tu să scrii, într-un sfârșit 😊)) “
* Ar trebuii să fie îngrijit – Cineva și-a făcut timp mai mult să îi confere un aspect cât mai simplu și ordonat
* Ar trebuii să fie simplu dar concentrat – Fiecare funcție, fiecare clasă. Fiecare modul expune o atitudine unică, care rămâne în întregime nedistată.

Cum să scriem un cod curat:

* Alegerea unui nume cât mai sugestiv pentru variabile – Alegerea unor nume cât mai bune, știu... necesită timp mai mult, dar economisește și mai mult timp decât este nevoie. Numele unei variabile, funcții sau al unei clase ar trebuii sa răspundă la toate întrebăriile cum ar fii “ Pentru ce e folosită variabila **x,** cum este folosită, etc”

Exemplu:

WRONG NAMING GOOD NAMING

Uint8 h; // hall sensor for SLV Uint8 hallSensor\_SLV ;

* Funcții

Aceasta este prima și cea mai importantă regulă legată de funcții. Ar trebui să le păstram cât mai scurte posibil. Explicația este destul de simplă: o funcție scurtă va face mai puțin (numai un singur lucru simplu). În plus de asta, va fi mai ușor de înțeles și de lucrat cu ea.

Dacă citești o funcție lungă, îți dai seama că face mai mult de un lucru acolo. De exemplu, în aceeași funcție deschizi conexiunea DB, execuți o cerere, transformi rezultatul într-un alt tip și te ocupi de cazurile speciale. Fiecare dintre lucrurile acestea ar trebui făcute separat.

De aceea o funcție ar trebui să facă numai un singur lucru. Dacă ai nevoie să faci mai mult de un lucru, atunci ar trebui să le împarți în funcții separate**.**

Întodeauna în momentul în care scriam o funcție mă gândeam oare câte linii de code ar trebuii scrise pentru ca funcția să functioneze?

* 500 de linii? – e cam mult
* 200 de linii? – oare e îndeajuns?
* 50 de linii? – totuși e cam puțin

Din păcate, nu putem avea un număr magic cum ar fi 5 sau 20, pentru că este destul de greu să generalizezi. Lungimea unei metode depinde de factori multipli, cum ar fi convențiile codului. De exemplu, cât de des apeși enter pentru a adăuga o nouă linie (pentru fiecare { sau pentru fiecare afirmație logică și așa mai departe).

* **Blocuri de indentare**

În legătură cu IF (dacă), ELSE (alt), WHERE (unde), REPEAT (repetare) și alte astfel de funcționalități, nu ai vrea să te trezești cu un IF de 10 linii. Ar fi destul de greu de citit și de înțeles. Pentru astfel de cazuri, ar trebui să extragi controlul (check) într-o funcție diferită și să îl apelezi din comanda IF. Aplicând această regulă, veți avea comenzi ca IF sau WHERE care necesită numai o singură linie de cod.

* **Argumentarea funcțiilor**

Să trecem la argumentarea funcțiilor. Cel mai optim număr de argumente al unei funcții ar fii 0 -> 1 -> 2 -> 3 -> *NOPE… cred ca ar fi de ajuns!* Întodeauna daca sunt mai mult de 3 argumente deja trebuie să te gândești dacă chiar se merită și dacă te ajută la ceva. O funcție cu un singur argument o folosim deseori ca și boolean sau punem un flag (0,1) pentru a verifica anumite lucruri. Pentru o funcție cu două argumente, justificarea acesteia poate fi mai grea deoarece, ambele argumente pot fi folosite pentru două lucruri total diferite cum ar fi, una să fie valoare unei variabile și cealaltă să fie un parametru

**REMEMBER!!!**

Chit că este doar o parte infimă descrisă în acest document despre Clean Code, unde mai pot fii scrise mii și mii de cuvinte despre tot acest mic univers al programării tot ce trebuie învățat este că  lucrurile mărunte pot face o mare diferență între o funcție bună și una proastă. Nu este nevoie să faci sau să știi te miri ce nebunii pentru a fi capabil să scrii funcții "fericite". Ținând cont de aceste mici descrieri vom putea scrie un software mai bun, care peste 10 ani va fi la fel de ușor de înțeles și de întreținut.