Качествен Програмен Код

Задача 1 – Есе на тема „Какво е Качествен Програмен Код“

Винаги ми е било трудно да отговоря еднозначно на този въпрос. Даже наскоро ме попитаха - „Опиши с няколко думи какво е КПК“. Ами какво да отговоря?

За мен КПК е нещо, което се разбира от пръв поглед - или го има, или го няма. С други думи - по някакъв повод отваряш съвсем произволен код, писан от „незнаен автор“ и в момента в който се визуализира - вътрешно разбираш дали е качествен. Естествено това е свързано с много детайли, които едновременно се набиват в очите.

Или ако трябва да опишем това в някаква последователност от елементи, то са:

- Правилно форматиране – кодът е така подреден, че позволява веднага да се оцени какво точно прави всеки един оператор, от къде до къде е тялото му, в кой елемент от по-горно ниво е включен (клас, метод, property getter/setter).
- Правилно именуване – всички имена са подбрани съобразно това, какви данни съхраняват/обработват и са достатъчно описателни, т.е. човекът разглеждащ кода да може веднага да се ориентира всяка една променлива/метод за какво се отнася.
- „По-завързаните“ и сравнително по-неясни фрагменти код са детайлно коментирани, без това в същото време да е прекалено претрупано, излишно и натоварващо. За всички останали случаи не е необходимо, тъй като кодът се „самодокументира“.

Всички неща, изброени до момента, помагат кода да бъде четим и лесно променян (поддържан) за в бъдеще от автора или от трети лица, а това винаги се налага в жизнения цикъл на даден софтуер (освен ако не е някакъв „скрипт“ за еднократна употреба, тогава няма никакъв смисъл от допълнителните усилия, свързани с писането на качествен код, така или иначе той е свършил вече своята работа).

Други също съществени елементи от КПК са:

- Кодът е документиран съгласно конвенциите на съответния език, т.е. всеки клас/метод има кратко описание за това какво прави, какви входни параметри приема и какви изходни параметри връща. Това е особено важно ако се пише библиотека или част от по-голям проект, който в последствие ще се използва от други хора с неясна квалификация. Тази документация чрез средствата на IDE-то ще “подсказва“ на бъдещия потребител как се използва предоставената функционалност.
- При изпълнението си кодът прави точно това, което се очаква от него и нищо друго, без значение от това дали входните данни са правилни или не. С други думи, количеството „бъгове“ е сведено до приемлив минимум, а входните данни винаги се проверяват за коректност. Винаги се приема, че потребителят неволно или умишлено може да подаде грешни данни и трябва да се реагира адекватно на тях. В такъв случай ако функционалността не може да бъде предоставена, то следва да се информира потребителят по съответен начин (съобщение за грешка, log и т.н.) и да се предостави възможност за корекция (където това е допустимо, разбира се).

- Кодът е написан по такъв начин (с използване на подходящи алгоритми, структури от данни, вече готови функционалности и т.н.), че да изпълни предназначението си в разумен интервал от време. Според мен „времето за реакция “ би следвало да бъде елемент от заданието на клиента или да се изхожда от естеството и обема на изпълняваните задачи, т.е. ако се управлява например индустриален процес в реално време то очевидно програмата трябва да „вземе решение“ за милисекунди, а ако се извършва например детайлно картографиране на обширен географски район, то резултатът „може и да се забави“).
- Кодът, в зависимост от своята сложност, трябва да бъде разделен на множество по-малки и по-прости модули на едно или няколко нива. Прилага се принципа „разделяй и владей“. По този начин се постигат няколко цели – решението на проблема е раздробено на много на брой прости и лесни за разбиране и коригиране стъпки. В същото време работата по този проблем може да бъде разделена на много хора, които да паралелно да съставят съответната част от общото решение. Постига се голямо бързодействие, гъвкавост и технологичния цикъл значително се съкращава. Тук естествено максимално трябва да се прилагат принципите на слабата свързаност между тези части (loose coupling), силната кохезия (strong cohesion) и скриване на вътрешните данни (encapsulation) в рамките на съответната част. По този начин всеки от разработчиците в такъв екип постига известна свобода в действията си, известна независимост от останалите и в крайна сметка възможност да се постигне високо качество на крайния продукт.

Като заключение – един основен белег за КПК е:

“Always deliver more than expected”.

(Всички прилики с действителността са случайни и неумишлени).