Какво е качествен програмен код?

Преди няколко години нещо доста интересно ми направи впечатление, докато се мотаех из нета и по сайтовете с обяви за работа. Навсякъде се търсеха „софтуерни инженери“. А що е туй? Защо не се търсят „програмисти“? И каква е разликата между двете, ако има такава? Това са въпросите, които мигновено изникнаха в главата ми и едва наскоро успях да си отговоря. Разликата е колкото малко, толкова и огромна. Програмистът е човек, който е усвоил основите на един език и може да „изпрограмира“ едно приложение. Софтуерният инженер, обаче, не бърза толкова – той създава архитектурата на приложението, следвайки определени езикови конвенции и практики. Той е знае как да правилно да софтуер, който не само върши работа, но и е лесен за поддръжка, разширяване и разбиране от други софтуерни инженери. Те умеят да пишат „КАЧЕСТВЕН ПРОГРАМЕН КОД“.

Усвояването на основите е задължително за всеки, който е решил да започне тази нелека професия, но качеството на написаният код е това, което различава просто програмистите от софтуерните инженери. Високото качество и добре тестваният програмен код улеснява работата, както на самият инженер, така и на всеки, който след това ще работи по неговия код. Добре написаната програмата има няколко основни характеристики:

* Добро, логическо и последователно именуване на променливите, методите, класовете и всички останали елементи на програмата. Ако името е достатъчно добре избрано то носи в себе си всичката необходима информация, като например каква функция изпълнява метода или какво държи в себе си дадена променлива. При качественият код не ни интересува толкова “как”, а по-скоро “какво”, т.е. не ни интересува как един метод си върши работата, а какво точно върши.
* Колкото и тривиално да звучи правилното форматиране е проблем при доста начинаещи програмисти. Има доста хора, които в бързината да свършат работа колкото се може по-скоро натракват нещо и смятат, че са приключили. Това обаче далеч не е така, защото лошо форматираният код означава трудна и бавна поддръжка, което от своя страна означава повече разходи за работодателя (или загубено време и пропуснати ползи за фрилансъра). Има определени конвенции за различните езици, но обща черта за повечето (ако не и за всички), е че всеки вътрешен блок трябва да по-навътре (по-индентиран ако мога да използвам направо тази чуждица) от предишния. Това отново създава едно много добро логическо разделение на различните части от програмата.
* Дебъгинг и юнит тестове, т.е. трябва да сме сигурни че софтуерът не само работи, а работи правилно и няма (или има минималното възможно количество) бъгове. Има различни подходи към разработката на софтуер и един от по-популярните е т.нар. test-driven development при който първо се пишат тестове, а след това се пише самият код. Едно от най-големите предимства на този начин на работа лежи в начина на мислене на самият програмист/инженер. Когато ни е представен даден проблем инстинктивно се замисляме как да го решим, но се случва да пропуснем някои аспекти на проблема, като някои частни случаи, например. При разработката, фокусирана към тестване първо се пишат тестовете, а чак след това се пише самият код – това е полезно, защото знаем какво точно трябва да се изпълни и намаляваме шанса да пропуснем някой важен аспект/функционалност/частен случай, когато започнем да пишем самата имплементация.

Следвайки конвенциите и правилата за писане на качествен програмен код ние успяваме да направим нещо много важно – успяваме да приравним програмният език към човешкия. Една програма, независимо колко сложно, която следва тези правила има елегантността, логическата последователност и яснотата на едно съчинение или роман написан на разбираем човешки език. Различните аспекти на качествения код могат логически да се приравнят към човешкия език по следния начин:

именуване = речникът на нашето приложение

форматиране = пунктуацията, която допринася за последователното разбиране на кода

тестване/дебъгинг = граматиката, гарантираща, че няма логически грешки (по-познати в ИТ сферата като бъгове)

Колкото по-близки са тези аспекти към вече познатите ни граматични и логически правила, толкова по-лесна за разбиране е тази програма. Английският и българският имат общи правила или черти, но имат и много различия помежду си – граматически, пунктуационни и др. По същият начин различните програмни езици се различават помежду си – в синтаксис, в функционалност и в доста случаи – по начин на мислене. Помислете за разликите в начина на мислене при C# и Lisp, които наподобяват на разликата между английски и китайски – и в двата езика може да се напише съчинение(програма), което да изразява (прави) едно и също и на двата езика, но реализацията би била коренно различна. Но при програмния език следвайки определени конвенции и пишейки качествен код можем да размием малко тази граница между различните езици – нещо което не е толкова възможно при човешките езици.