Contents

[Codding C# 2](#_Toc48417476)

[Naming convention 2](#_Toc48417477)

[Git Hub 2](#_Toc48417478)

[Dobra praksa 2](#_Toc48417479)

[Project - Tank 3](#_Toc48417480)

[Plan rada: 3](#_Toc48417481)

[Ostalo 3](#_Toc48417482)

[Struktura Calc promenjive tipa TCalc 4](#_Toc48417483)

[Komentari na ko di ostale teme 5](#_Toc48417484)

[001#: Explicitno zadavanje brojeva 5](#_Toc48417485)

# Codding C#

## Naming convention

1. Imena sto kraca. Najbolje **4-6 karaktera duzine**. Kraci nazivi znace krace linije koda i lakse citanje. Pogotovo je bitno za nazive promenjivih.
2. Za kapitalizaciju naziva najbolje da koristimo **PascalCase**. Spojene reci sa prvim velikim slovom. PascalCasenije bas C# praksa al deluje najpismenije od svih.

# Git Hub

## 

## Dobra praksa

1. Uvek ici sync/pull pre pravljenja grane da neostane nesto lokalno

# Project - Tank

## Plan rada:

1. Napraviti minimalni model kojic ce raditi. Za postetak samo proracun tanak na unutrasnji pritisak.
   1. Dummy funkciju koja ce popunjavati vrednosti prema excelu
   2. Dummy funkciju koja ce raditi deo proracuna
   3. Dummy funkciju koaj ce izbaciti report
2. Ispeglati postojeci program da omoguci ponavljanje unosa podataka iz tacke 1 ali sa funkcionalnim interfejsom. Takodje se uveriti da report ima sve podatke koje ce imati finalan verzija.
3. Preci na drugi deo proracuna iz excela koji ide posle internal proracuna. Pa ponoviti sve iz tacke 1 i 2.
4. Za listu shelova treba dodati dugme da se shell pomeri 1 red na gore i da se pomeri jedan red na dole (“Move up”, “Move down”).
5. Za listi shelova se moze dodati da “Add” nedodaje na kraj nego posle selektovanog shela. Tako bi imao funkciju Add/Insert u isto vreme.
6. Dodati check boks koji ce omoguciti da shell ima default koroziju.
7. Kod globalnih vrednosti treba uraditi tako da se one unose automatski kad izgube focus. Tj treba uradi ubacivanje vrednosti u nase klasu na “Leave” and “KeyDown” evente.
8. …

## Ostalo

1. Treba pratiti utrosene sate za kasnije evidenciju negde na lokalu. Da bi mogli napraviti neku procenu u slucaju potrebe. Za sad bi nas interesovalo:

* koliko je vremena potroseno
* Koja aktivnost je u pitanju (kodiranje, trazenje resenja, analiza proracuna…)
* Voditi evidenciju po danima
* Kolko je efikasnost u %. Kao opist toga koliko je vremena stvarno otislo na rad u unapredenju aplikacije a koliko je vremen na ucenje, resavanje problema kad zapne… Odnos korisnog vremena za aplikaciju i svega ostalog. Ovo je slobodna procena. Ako je 10% znaci slabo se radilo na aplikaciji al se mozda ucilo i istrazivalo ako je 80% znaci sve sto je radeno uslo je u aplikaciju pa makr bilo i pogresno :D. Tako nekako….

Kao primer: 2h kodiranje 70%,

2h API660 30%,

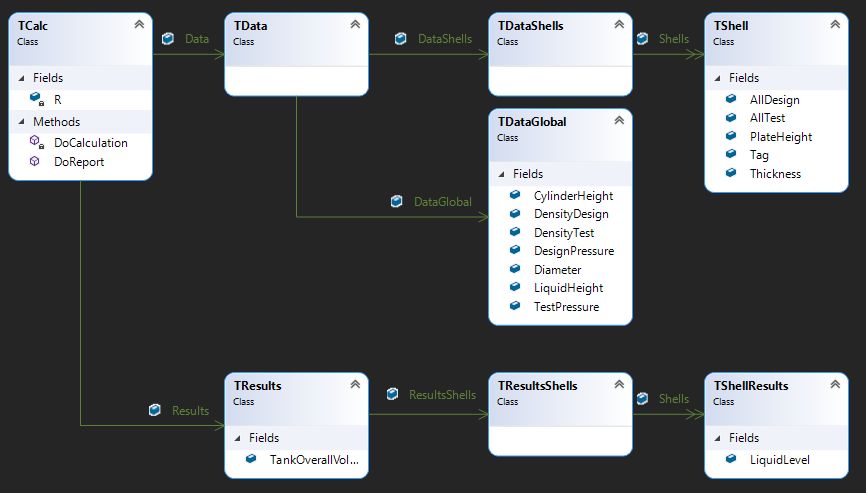
3h borba sa VS interfejsom 10%....

1. Trebalo bi koristi standardne SI jedinice svuda. Tipa mm za precnik i N za silu. Ako bas treba konverzija mm u m ili N u daN to treba jos razmotriti. Uvek treba raditi u istim jedinicama da nedodje da neocekivane greske u konverziji.

## Materijali

Materijal ID od 1 do 99 je rezervisan za custom materijale. Materijal index od 100 do 9999 se trenutno nekoriste. Svi materijali od 10000 pa nadalje treba da se koriste za materijale iz baze Tables.xlsx.

## Struktura Calc promenjive tipa TCalc



# Komentari i ostale teme

## 001#: Explicitno zadavanje brojeva



Ovo se radi samo kad je bitno explicitno naglasiti kog je tipa broj. Da je 3.2 float. Ako to nije bitno compiler ce sam to uraditi inplicitno na osnovu tipa promenjive. Kada je promenjiva float i vrednost float nece se nista posebno desiti. Ali kad se promenjivoj promeni tip u recimo double, vrednost je i dalje explicitno zadata kao float pa dodje do sitne greskee. Ovo je karakteristicno bas za realne brojeve.

3.2F u double promenjivoj ima vrednost 3.1999999. Dok samo 3.2 u double promenjivoj ima vrednost 3.2.