Better code hub (BCH) Evaluatie

Naam: Mark Brinkman & Jari Kuipers

Studentnummer: 346387 & 342386

Klas: ITV2D

Boggle opdracht

# Write short unit of codes

Hierbij hebben wij nog net een voldoende. Dit komt omdat er een paar methodes met in de 20/30 regels code zijn.

Hiervoor heb ik gekozen om ze zo groot te laten omdat het moeilijk op te delen is of er niet duidelijker van wordt.

# Write simple units of codes

Op simple units of codes hebben wij ook net een voldoende omdat er meerdere methoden zijn die veel branch points. Dit komt omdat deze hier niet minder branch points kunnen hebben.

# Write code once

Er is geen dubbele code in dit project dus is dit een voldoende.

# Keep unit interfaces small

We hebben het aantal parameters in methoden in de code hebben we zoveel mogelijk klein gehouden en waar nodig in objecten gestopt. Daarom is dit voldoende.

# Separate Concerns in Modules

We hebben alle objecten netjes in verschillende klassen opgedeeld. En is daarom dus voldoende.

# Couple architecture components loosely

Op couple architecture components loosely hebben wij een onvoldoende omdat er veel calls zijn van en naar methodes in andere klasse worden gedaan. Hierdoor zijn de klasse dus niet loosely koppelt.

# Keep architecture components balanced

Onze code is momenteel goed gebalanceerd waarbij ons Model het grootste is met 191 regels aan code en onze kleinste is de Main waarin 41 regels aan code staan.

# Keep your codebase small

Omdat het programma nog maar heel klein is zit de codebase nog op voldoende. Dit is momenteel bij ons nog 0.08 man jaren.

# Automate tests

Er staat een onvoldoende voor Automatische tests omdat deze niet zijn toegevoegd omdat deze applicatie in verhouding nog heel klein.

# Write clean codes

Op clean codes hebben wij een voldoende omdat er geen opmerking zijn over dat er nog uit gecommenteerde stukken codes of andere rommelige code.

# Algoritme complexiteit

Het grootste algoritme in de code is het vinden van alle woorden in het bord.

Het algoritme waarmee alle woorden worden gevonden is een depth first search algoritme.

Dus de complexiteit van dit algoritme is O(b^d).

Waarbij b gelijk staat aan de hoeveelheid branch points er zijn en d gelijk staat aan de diepte waarin je gaat zoeken.