Better Code verslag - Jesse Tijsma & Mark Dissel - ITV2D - OOP3

Dit verslag is bedoeld rondom de twee OOP3 opdrachten die ingeleverd moesten worden aan het eind van deze werkcolleges. Op Code Better heeft de code vijf van de tien punten behaald. Dit is het minimum van de vereisten. De verbeterpunten zullen hieronder uitgelegd en beschreven worden.

Onjuist: Write Short Units of Code

Bij een aantal methoden hebben wij iets te veel code gebruikt, dat wellicht ook korter kon via loops of andere statements. Op het moment van schrijven hadden wij dit niet door, maar nu wij achteraf erop terug keken kan dit in de toekomst wellicht anders. De Boggle code bevatte één methode die hij als rood beschouwde. Dit is de ergste vorm, maar bij nader inzien hebben wij geen andere oplossing kunnen verzinnen.

Onjuist: Write Simple Units of Code

Volgens Better Code was er één methode vanuit de Boggle opdracht niet correct. Dit was dezelfde methode die ook in de eerste punt niet goed was. Hier geld dus net zo dat wij hier geen andere oplossing zien, maar hierdoor twee punten onvoldoende zijn.

Juist: Write Code Once

Dit punt was volgens Code Better voldoende. Echter was er wel één regel code dat niet correct was. Er kwam twee keer een aantal regels voor, dit konden wij beter als methode neerzetten en twee keer aanroepen, zodat duplicate code vermeden wordt.

Onjuist: Keep Unit Interfaces Small

Dit punt was volgens Code Better onvoldoende. Dit komt omdat we soms gekozen hebben voor functies die veel input nodig hebben. Een voorbeeld hiervan is de Quicksort methode. Deze heeft een Integer nodig om als beginpunt te zien, eindpunt (einde van de Array) en een list waar hij door moet “loopen”. Omdat recursie van toepassing is, kunnen we niet 1 keer alles initialiseren en het daarbij laten. Bij de Slider kunnen we het misschien nog anders aanpakken, maar persoonlijk vinden wij het handig dat je in 1 keer een slider aan kan maken, en helemaal zelf kan maken en / of aanpassen. Verder hebben we nooit meer dan 4 parameters gebruikt, en was 3 altijd de max (op 1 situatie na, wat door tijdgebrek niet af kwam).

Juist: Separate Concerns in Modules

Volgens Better Code hebben wij de afhankelijkheid tussen modules goed afgewerkt. Dit klopt ook, aangezien alle klassen bij ons apart staan van elkaar. Een opdracht bestaat uit maar één klasse.

Juist: Couple Architecture Components Loosely

Eigenlijk geld hier hetzelfde als de vorige punt en dit hebben wij ook correct gedaan.

Onjuist: Keep Architecture Components Balanced

Het is vrij terecht dat dit als slecht wordt beschouwd. We hebben alle code in 1 Java bestand gemaakt. Graag wilden we het spreiden in verschillende bestanden, maar wegens tijdgebrek zijn we hier niet meer aan toegekomen. Als we een weekend langer de tijd hadden gehad, hadden we dit zeker nog geïmplementeerd.

Juist: Keep Your Codebase Small

Volgens Code Better hebben wij voor twee java bestanden relatief weinig code gebruikt. Dit is voordelig zodat onderhoud en eventuele updates makkelijker doorgevoerd kunnen worden.

Onjuist: Automate Tests

Dit is een puntje van aandacht bij ons. We hebben momenteel een grafiek die willekeurig wordt gegenereerd. Dit betekent dat deze altijd andere waarden heeft, waardoor het zo is dat als je je code wilt testen, je altijd een andere grafiek gebruikt. Hierdoor is het testen niet efficiënt, en dit hebben we voornamelijk gemerkt bij het testen van QuickSort.

Juist: Write Clean Code

We hebben hier aan het einde zeker nog aandacht aan besteed. Tijdens het testen hadden we veel onnodige comments, niet gebruikte variabelen en / of methodes, prints en overig testmateriaal of overbodige code. Dit hebben we allemaal verwijderd, waardoor de code er nu toch een stuk beter uit ziet. Verder hebben we nog gedacht aan nette documentatie waar het nodig is.