Formules

Wij hebben wat betreft de forumules ervoor gekozen om voor elke formule een aparte klasse te maken. Al deze formules implementeren daarbij de abstracte klasse Formule. Deze abstracte klasse formule bevat de abstracte methode *executable(String[] a)*, die alle formule-klasses dus moeten hebben. In deze methode *executable(String[] a)* wordt voor elke formule de inhoudelijke werking gedefinieerd. Waarom we hiervoor gekozen hebben is omdat we natuurlijk een groot aantal formules kregen die allemaal op basis van de structuur hetzelfde deden: zij krijgen specifieke waardes binnen, vervolgens verwerken zij deze waardes op basis van hun functionaliteit en als laatste returnen ze een uitkomst. Hierin herkenden wij, op basis van wat wij geleerd hebben bij het vak OOP-programmeren van het vorige kwartaal, direct het implementeren van klasses en methodes. We verplichten alle formule-klassen de methode *executable(String[] a)* te hebben waardoor er nooit een formule zal zijn die niet werkt of niets doet.

Eerst hadden wij in gedachte om voor alle formules een aparte methode in één grote klasse te maken. Maar na suggestie van meneer Zaidman en onderlinge overleg leek het ons handiger de manier te gebruiken die wij nu hebben geïmplementeerd. Ten opzichte van ons eerste idee vonden wij dat dit namelijk de volgende voordelen had:

* Het toevoegen van nieuwe formules is eenvoudig. Het toevoegen van een nieuwe formule vereist namelijk alleen een nieuwe klasse aanmaken en de methode *executable(String[] a)* te definiëren. Als we later meer formules willen implementeren, gaat dat dus heel gemakkelijk. Hierbij komt nog dat meneer Zaidman vertelt had dat je met behulp van een String een klasse kon aanroepen, dit zou heel erg goed samenwerken met de structuur van onze methode.
* Het is overzichtelijker. In plaats van één grote klasse met 20+ methodes voor elke formule en daarbij nog eens aparte methodes die de formules ondersteunen wordt al snel onoverzichtelijk. Niet alleen qua documentatie, zoals javadoc, maar ook qua werkstructuur. Alle formules staan door elkaar en je kunt al heel snel een fout maken doordat onoverzichtelijk is, bijv: het is heel snel mogelijk haakjes heel slecht van elkaar te kunnen onderscheiden; welk deel sluit waar, welke deel begint waar. Bij onze methode heb je voor elke formule 1 kleine klasse en dus ook 1 javadoc per formule.
* Ook werkt dit naar onze mening fijner. Zo kun je snel in één opslag zien hoe ver je bent met een formule, welke formules nog niet af zijn, wat er nog moet gebeuren, etc. Hierdoor krijg je een betere indicatie van hoever je bent en hoeveel je nog moet doen.
* Verder was dit naar onze mening de meest efficiënte manier om dit te implementeren wat betreft onze kennis van java. Wij zijn allemaal beginners qua programmeren en hebben dus dit oordeel gemaakt op basis van wat wij geleerd hebben bij OOP in kwartaal 1. Wij hebben bijvoorbeeld voor een String-array gekozen in plaats van bijvoorbeeld een Hashmap omdat destijds de String-array één van de weinige manieren was waarmee wij wisten dat meerdere waardes in opgeslagen konden worden. Hierbij hadden wij afgesproken dat alle waardes als String opgehehaald worden uit de cellen en de uitkomsten ook als String gereturnt worden.

Wij zijn niet echt tegen grote problemen opgelopen. Voor de problemen waarvan wij niet zeker wisten hoe we ze moesten oplossen hebben we regels opgesteld waaraan wij ons hielden. Deze problemen zijn grootendeels hierboven allemaal beschreven. Overal waarvoor wij iets afgesproken hadden was oorspronkelijk een probleem of een struikelblok. Het contract/de regel die wij opgesteld hebben is hoe wij dachten dat dit het beste was om op te lossen.