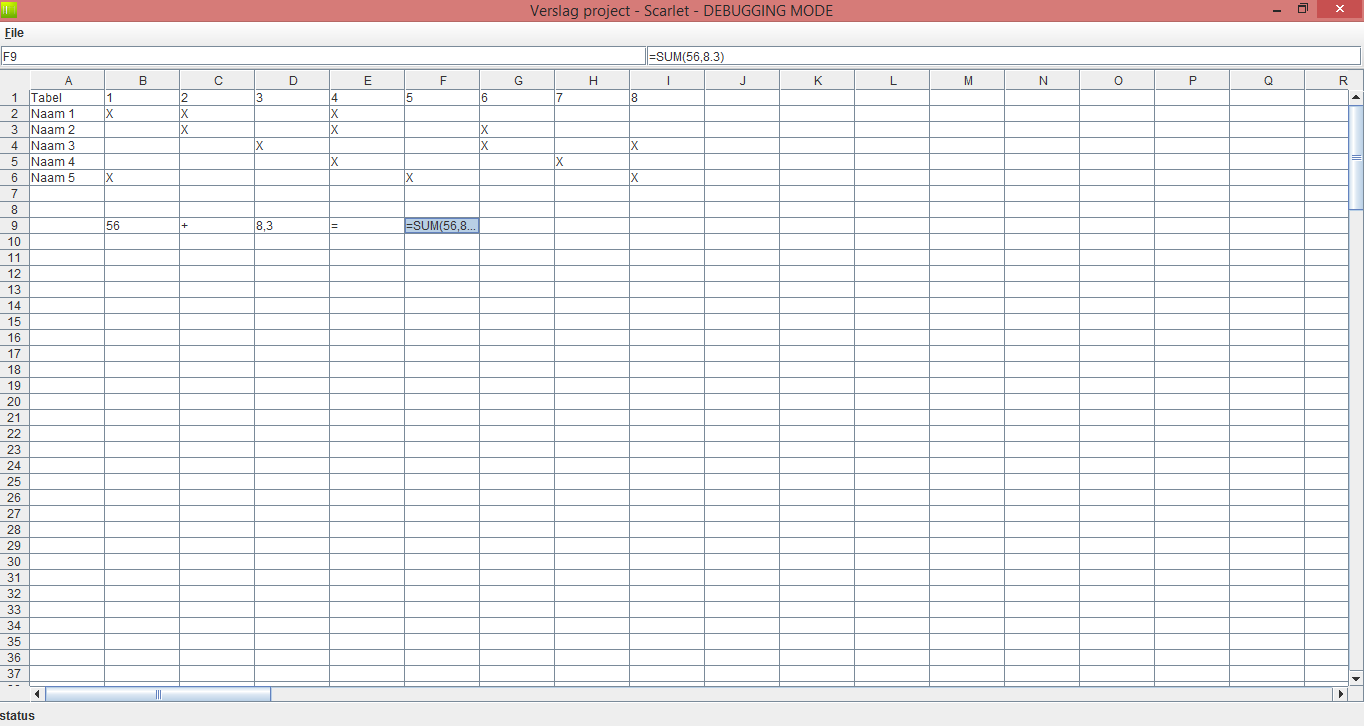
Spreadsheet in Java

Door Team Scarlett



Martijn Gribnau - 4295374

Mitchell Olsthoorn - vul in

Roy Klip - 4293908

Ike Rijsdijk - vul in

Robin Borst - vul in

Alan van Rossum - 4293932

Bachelor Technische Informatica, 2013-2014

OOP Project

**Inhoudsopgave**

1. Algemeen: hoe is het project verlopen?

2. Ontwerpproces

3. Verbeterpunten

4. Individuele feedback

**Algemeen: Verloop van het project**

**Planning**

Door de vergaderingen elke maandagmiddag, kregen we een mooi beeld van de taakverdeling en de planning. Ook met behulp van de planning die beschikbaar is op blackboard, konden we zelf een goede planning in elkaar zetten. Vooral in het begin was iedereen enthousiast en werd er hard gewerkt. We liepen toen zelfs voor op de planning. Maar naar mate de weken vorderden en er minder taken over bleven, werd er minder hard gewerkt. Gelukkig hadden we in het begin een voorsprong opgebouwd, zodat dit voor ons geen problemen opleverden. Helaas werd vaak, ondanks dat we telkens een week de tijd hadden, toch pas zondag wat gedaan. Het goede is, dat er die zondag wel vaak werd afgemaakt wat er af moest voor maandag.

Tijdens de maandagmiddag werd vrij weinig geprogrammeerd. Die tijd besteedde we aan vergaderingen en ander overleg. Het programmeren gebeurde dus bijna alleen thuis.

**Samenwerking**

De samenwerking binnen het team verliep goed. Ondanks dat er natuurlijk wel verschillende niveaus van programmeurs in het team zaten, was de rolverdeling voor iedereen gunstig. Doordat iedereen kon doen waar hij goed in is of wat hij leuk vindt, was iedereen tevreden. Hoewel er soms wel hulp kwam van andere rollen, deed eigenlijk ieder zijn ding.

Elk groepje bestond uit 2 à 3 personen en de taken van de groepjes waren als volgt verdeeld: GUI, UML, verslag, back-end (formules, parsers, testen enzovoort). Sommige teamleden speelden een rol in meerdere groepjes om het beter te verdelen.

Bij het samenwerken in de groepjes zelf, werd ook bijna alleen maar thuis gewerkt. De eerste weken werd er nog wel samengewerkt op maandag tijdens de projecturen, maar zoals ook al bijna de planning uitgelegd, werd er de laatste weken niet meer zo veel gedaan op maandag. Kortom het meeste werk werd thuis gemaakt.

**Communicatie**

Voor de communicatie bij het project waren vele mogelijkheden. Na wat te hebben geprobeerd via github en Trello, bleek uiteindelijk de makkelijkste oplossing de mobiel te zijn. Daarom hadden we besloten een Whatsappgroep aan te maken voor de algemene communicatie.

Van Trello hebben we na één week geen gebruik meer gemaakt. We vonden het niet nodig om van een extra programma gebruik te maken om de planning bij te houden. Ook omdat we elke week vergaderden en omdat daarbij notulen werden gemaakt, was dat eigenlijk een betere manier om de planning bij te houden.

Verder vond verre weg de meeste communicatie face-to-face plaats. Niet alleen tijdens de vergaderingen op de maandagmiddag, maar ook tijdens pauzes of tijdens colleges werd zo nu en dan overlegt.

Voor versiebeheer maakten we gebruik van github. Dit werd geadviseerd tijdens één van de colleges. Daarna hadden we vrij snel besloten om dit te gebruiken. Github is voor ons een enorm handig programma, omdat met dit programma elkaar gemakkelijk op de hoogte kunnen houden van de laatste veranderingen in het project. Ook voorkomt github verwarring als bijvoorbeeld meerdere mensen in dezelfde klasse iets hebben veranderd.

**Ontwerp proces**

**Ontwerpkeuzes**

“Maak een spreadsheet in Java”. Deze opdracht laat vrijwel alle keuzes over aan de ontwerpers. We hebben daarom ons best gedaan om voor elk onderdeel van het project de best passende methode te gebruiken. Hier volgen onze belangrijkste keuzes.

**De opslag van inhoud van de cellen**

Dit was een van de eerste problemen waar we tegenaan liepen. Eerst lag het voor de hand om een tweedimensionale Array van Cell-objecten te maken, in de vorm van Cell[][]. Dit leek ons in eerste instantie de beste optie, omdat het zo makkelijk was om de inhoud van een willekeurige cel op coördinaten (x,y) op te vragen. Je kan dan namelijk gewoon Cell[x][y].getValue opvragen, en zo zeer snel de benodigde waarde ontvangen.

Het nadeel van deze methode was dat we de hele Array vol moesten zetten met null-values voor lege cellen. Nu kwam het idee van een tweedimensionale ArrayList tevoorschijn. Een ArrayList in een ArrayList dus. Het voordeel hiervan was dat je alleen de waardes van volle cellen hoeft op te slaan. Zodra een Cell een value heeft, voeg je hem toe aan de ArrayList. Het nadeel was echter dat elke Cell nu ook een x en y-coördinaat moet hebben, aangezien de index van de Cell niet altijd overeenkomt met de x- en y-coördinaten van de Cell. Het grote nadeel is dat het niet makkelijk is om de inhoud van een willekeurige ccel op coördinaten (x,y) op te vragen. Er moet namelijk gebruik worden gemaakt van een for-loop in een for-loop, die elke index afgaan en checken of de Cell.getX en Cell.getY de coördinaten zijn die zijn opgevraagd. Dus een ArrayList gebruikt minder geheugen, maar is wel langzamer.

In week 4 leerden we echter de ultieme methode kennen: de HashMap. In een HashMap link je Objecten niet aan een index, maar aan een key. Door in deze key een x- en y-coördinaat te zetten, bijvoorbeeld in de vorm van (11,2), link je elke Cell-object aan zijn betreffende coördinaten. Hierdoor is een willekeurige cel op coördinaten (x,y) snel op te vragen, zonder loops die elk Object in de lijst afgaan. Bovendien slaat een HashMap alleen de Cellen op met een waarde, net als een ArrayList, dus is een HashMap qua geheugengebruik ook veel efficiënter.

Onze HashMap heeft dus de snelheid van een Array[][], en de geheugencapaciteit van een ArrayList<ArrayList<>>.

**Git of SVN**

- Git is gedecentraliseerd

**Aparte klassen voor formules**

- modulair

**Parser**

- Patterns / recursive voor nested formulas

**UML, hoe heeft het wel/niet geholpen**

**test coverage(inclusief screenshot package of klasse niveau plus beargumenteer klassen die heel goed of niet heel goed gecovered zijn)**

**Verbeterpunten**

**Hoe kan onze software verbeterd worden?**

Voor zover onze software, valt er nog veel te verbeteren. Zeker als we kijken naar een professioneel programma als Excel. Aan de andere kant, voor wat we hebben werkt de software goed. Wat de software wel zou verbeteren, is extra mogelijkheden. Als je kijkt naar de GUI is te zien dat er slechts één menu is en dat is ‘file’. Dit is vrij weinig voor een spreadsheet programma. Opties als grafieken is een goede extra mogelijkheid.

Ook de opmaak van cellen zoals kleur of lettertype zou een grootte verbetering zijn van onze spreadsheet. Vooral omdat dit meer duidelijkheid brengt in bijvoorbeeld tabellen die je maakt in het programma.

Als laatste verbeterpunt kunnen we de hoeveelheid cellen noemen. Onze spreadsheet heeft namelijk een beperkt aantal cellen. Hoewel het niet vaak zal voorkomen dat er zo enorm veel cellen worden gebruikt bij één spreadsheet, is het handig als het programma de grootte van de spreadsheets dynamisch maakt, zodat het aantal cellen als het ware onbeperkt is.

**Hoe kan het vak OOP Project verbeterd worden?**

Het vak OOP Project kan hier en daar nog worden verbeterd. Bijvoorbeeld in het begin was het lastig om te bedenken hoe je nu aan het project moet beginnen. Vragen als: ‘moeten we al aan de GUI beginnen’, ‘wat moet er allemaal worden getest’ en ‘Hoe moeten de XML files worden geparst’, werden vaak gesteld.

Verder was er ook onduidelijkheid rondom de UML. Ondanks dat het in een college wel werd behandeld, was het nog niet duidelijk hoe je dat nou precies in elkaar zet. Er had wat meer aandacht aan mogen worden besteed. Een eventuele optie is om ons, de studenten, een opdracht te laten maken en de opdracht houdt dan in om een UML te maken van een programma. Aan de hand daarvan kan dan feedback worden gegeven. Bij ons ging het bijvoorbeeld fout bij het maken van de relaties en bij het neerzetten van de methodes in de klassen. Het was ons niet duidelijk welke methodes we wel en welke we niet in de UML moesten plaatsen.

**Hoe kan ons proces verbeterd worden (planning, overleg, commits, uml gebruik)**

Doordat dit ons eerste project was, was het proces niet perfect. Het vergaderen liep vaak chaotisch en er werd niet altijd naar de planning gekeken. Eén van de grootste verbeterpunten is dat we van te voren een UML moesten maken. Dit wisten we niet. Als je namelijk de UML eerst maakt, dan weet je waar je naar toe moet werken en hoe de verschillende klassen en packages met elkaar samenwerken. Eigenlijk heb je met de UML als een bouwtekening van je project en bouwtekeningen worden altijd voor de bouw gemaakt.

Het tweede punt dat tijdens ons project verbeterd kan worden, is het vergaderen. Zoals al eerder gezegd, verliepen die meestal chaotisch. In het begin is het natuurlijk wennen, maar zelfs de weken daarna was het vergaderen nog niet professioneel. Hoewel de vergadering altijd wel volgens de planning liep, werd er toch vaak wat gewisseld tussen de onderwerpen. De planning van de vergadering werd ook meestal erg laat verstuurd. Soms de avond van te voren, som de dag zelf en soms zelfs helemaal niet. Dit geldt ook voor de notulen, want die werden tegelijk met de planning van de vergadering gestuurd.

**Individuele feedback**

**Reflectie op functioneren binnen het team pp, sterke/zwakke punten, conflicten etc.**

**Martijn**

**Mitchell**

**Roy**

**Ike**

**Robin**

**Alan**