DESIGN PATTERNS INTRODUCTIE

WEEK 4, 5 EN 6

Design Pattern Opdracht: “Koffiemachine”

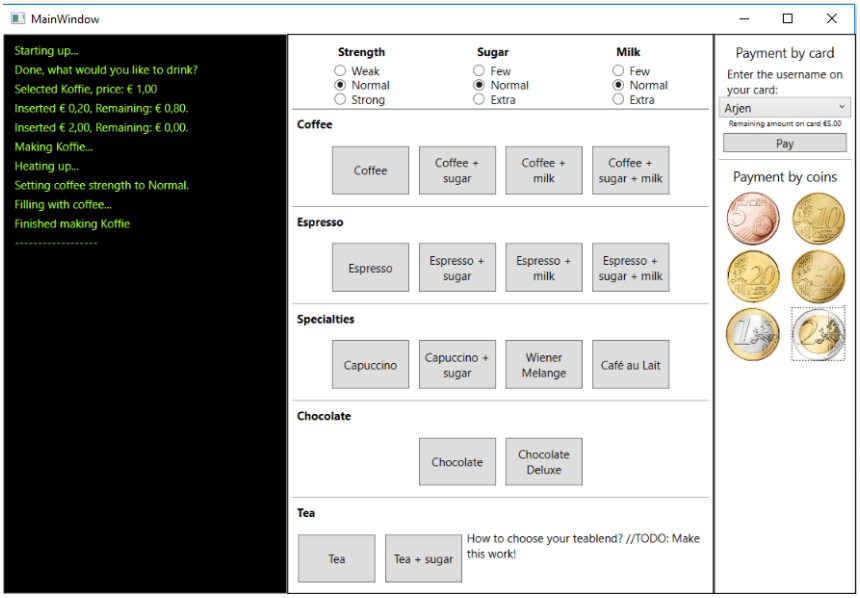
Deze weken ga je aan de slag met een grotere applicatie. Deze applicatie simuleert een koffiemachine. Je kan koffie bestellen naar jouw voorkeur, vervolgens kan je contant of per bankpas betalen en ten slotte wordt je drankje gemaakt. De code van deze machine is echter niet heel netjes opgesteld en je bent gevraagd om dit te refactoren. Maak gebruik van je eerder opgebouwde kennis van design patterns.

Deze opdracht zal de stappen in grotere lijnen uitleggen dan de vorige opdrachten. Het is de bedoeling dat je hier naar eigen inzicht de code refactoret en de juiste patronnen toepast.
Wanneer de docent vindt dat je dit voldoende hebt gedaan zal je opdracht afgetekend worden.

1. Clone de repository en run de code

Clone het project https://github.com/Avans/DPINT_Wk456_Koffiemachine
Hier zie je een window die alles controleert. Dit scherm bestaat uit drie delen:

- In het midden zie je knoppen om je drankje samen te stellen;
- Links zie je een output window die precies vertelt waar het apparaat mee bezig is;
- Rechts zie je een scherm met bovenin de mogelijkheid om met je pas te betalen of onderin knoppen om met contant geld te betalen.
- Let op: Chocomel en Thee werken nog niet, het is de bedoeling dat je deze later nog implementeert.



DESIGN PATTERNS INTRODUCTIE

WEEK 4, 5 EN 6

2. Bekijk de code

We hebben twee projecten, het WPF-project Dpint_wk456_KoffieMachine en de class library KoffieMachineDomain.

De domeinklasses bevatten de verschillende soorten drankjes met hun configuraties, hun prijzen en ze kunnen tekst uitprinten om te laten zien hoe ze gemaakt worden.
Er is later nog wel een eis bij gekomen dat er geld gerekend werd voor suiker en melk. De ontwikkelaar kende de code helaas niet goed genoeg en heeft mede door de werkdruk deze prijzen daarom in het WPF-project laten berekenen.

Het WPF-project bevat twee helperklassen voor de XAML. Hier hoef je niets aan te veranderen, want deze code is gekopieerd van Stackoverflow en de vorige ontwikkelaar heeft deze code al heel vaak in projecten geplakt zonder dat het ooit mis ging.
De rest van de logica zit allemaal in het MainViewModel. Dit begon ooit als 'even snel implementeren' en is uitgegroeid tot wat het nu is. De business logic zit hier dus ook allemaal in verweven helaas.

3. Bugfixing en refactoring MainViewModel

De code is nu een wirwar. Er staan veel switch/case statements in die erg op elkaar lijken en ongeveer hetzelfde zouden moeten doen. Echter zie je bijvoorbeeld dat de namen in de logs niet allemaal overeenkomen met de namen in de buttons en dat de prijzen niet altijd hetzelfde worden geformatteerd.
Probeer eens met de eerder geleerde patronen deze code op te schonen en los te trekken uit de MainViewModel.

We hebben ook geleerd dat een klasse verantwoordelijk moet zijn voor één ding. Hoe kan het dan dat de algoritmes om drankjes te maken en betalingen af te handelen in dezelfde klasse staan? Kan dit niet gescheiden worden? En zouden de pasbetalingen wel bij de contantbetalingen moeten staan?

4. Refactoring domeinklasses

De logica in de domeinklasses lijkt wel heel erg op elkaar. Vaak worden dezelfde if-statements gebruikt, stel dat we in de toekomst ook nog een crème-topping zouden kunnen kiezen dan moeten we al deze klasses aanpassen. Kan dit niet beter?

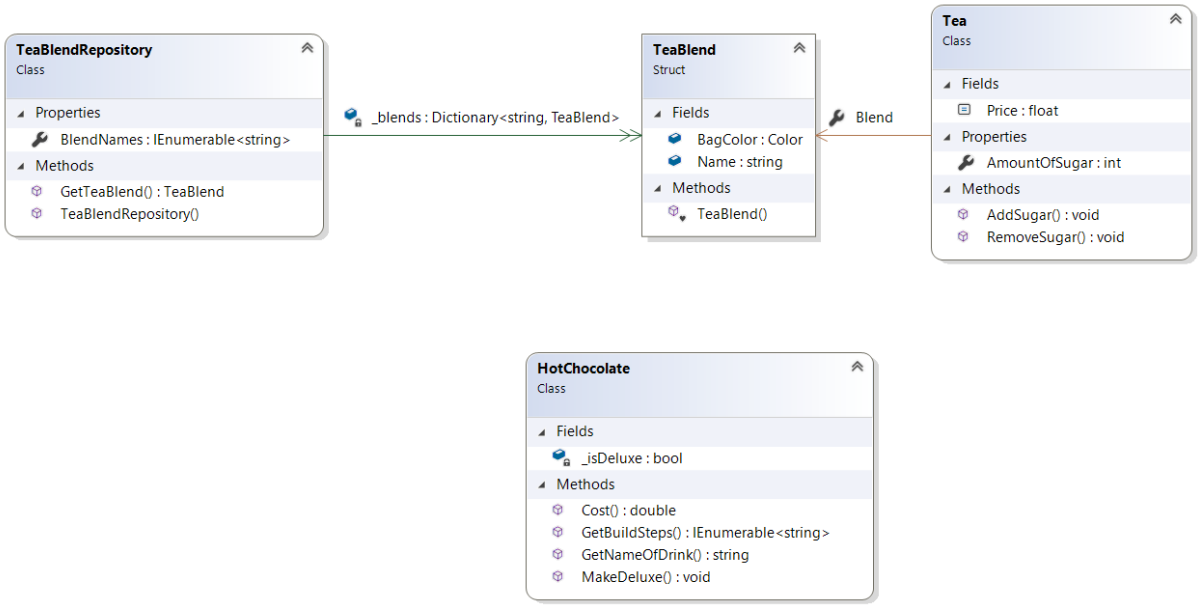
DESIGN PATTERNS INTRODUCTIE

WEEK 4, 5 EN 6

5. Nieuwe drankjes

De automaat krijgt een nieuwe dispenser erbij. We kunnen nu dus ook de drankjes van de onderste knoppen maken. De leverancier van de dispenser heeft een library aangeleverd die de klasses bevat voor de nieuwe drankjes, helaas matchen de interfaces van die klasses niet met die van ons. Kunnen we onze software toch minimaal aanpassen zodat we de dispenser kunnen gebruiken? Hier bestaan vast handige patronen voor.
Gelukkig hebben we wel een klassediagram van de leverancier gekregen, die is hieronder afgebeeld.

Maak een referentie aan naar de externe library en zorg ervoor dat je code de nieuwe drankjes HotChocolate en Tea (inclusief Blend) ondersteunt.



6. Bijzondere drankjes

De automaat heeft weer een nieuwe dispenser, echter is hier nog geen third party library voor dus moeten we dit zelf schrijven. Gelukkig zijn het wel de drankjes die je op je werkplek zou willen zien:

Special coffees!

Maak het mogelijk om een special coffee te bestellen, denk hierbij aan **Irish Coffee** (Whisky, koffie, suiker, slagroom), **Spanish Coffee** (Cointreau, cognac, koffie, suiker, slagroom), **Italian Coffee** (Amaretto, koffie, suiker,slagroom).

Om toekomstgericht te kunnen werken dient de **configuratie** van deze soorten koffie in een los bestand gedaan te worden. Dit kan bijvoorbeeld met een **XML-, JSON- of CSV-formaat** gedaan worden. Dit maakt het mogelijk om een Grand Café met vele honderden soorten koffie te kunnen helpen.