Groepsopdracht OO Ontwerpen 2019-2020

1. Inleiding

Deze opdracht bestaat uit het ontwikkelen van een kassasysteem om een verkoop van een klant in een winkel af te handelen

Concreet bestaat de opdracht uit drie delen:

- Een volledig uitgewerkt klassendiagram opstellen voor het programma.
- Het schrijven van een verslag aan de hand van de gegeven template.
- Een Java implementatie van dit klassendiagram schrijven.
    - Let bij je oplossing vooral op een goed gebruik van design patronen en principes!
    - Zorg voor een degelijk ontwerp, zodat je code leesbaar, onderhoudbaar en uitbreidbaar is.

2. Afspraken

Algemeen:

- Deze opdracht telt mee voor 5 punten (van de 20) voor het opleidingsonderdeel OOO
- Deze opdracht is **in groep (3 studenten)** te maken.
- Merk op dat indien er vastgesteld wordt dat je het project in overleg met andere groepen gemaakt hebt, of indien code fragmenten te gelijkend op elkaar in verschillende projecten teruggevonden worden, dit aanleiding kan geven tot een fraude dossier
- Je beheert je code in een GIT repository. De link naar dit project zet je in je verslag .
- Deze opgave is een **richtlijn**. Voel je zeker vrij om, in lijn met de geziene principes, uitbreidingen toe te voegen of bepaalde beslissingen aan te passen.
- Pas patronen correct toe in de code. Laat zien dat je ze beheerst.
- Voel je vrij om elementen/functionaliteit toe te voegen waar deze jou gepast lijken om de patronen aan bod te laten komen.

Deadlines:

- Je checkt, tijdens het ontwikkelen, je code steeds in op GIT.
- Je dient op het einde een zip van je java source files (als backup) en het verslag in via Toledo, ten laatste op maandag 23 december 2019 om 23h59. Hierna wijzig je niets meer aan de code in je GIT repository.
- In de laatste lesweek wordt je groepswerk tijdens het labo besproken. Aanvullingen en verbeteringen zijn nadien nog mogelijk.

3. De applicatie

Inleiding

We gaan een sterk vereenvoudigd kassa systeem bouwen voor het afhandelen van een verkoop aan de kassaterminal van een winkel.
Veronderstellingen:

- Er is slechts één kassa en één kassier die tevens administrator is van het kassasysteem
- We registreren geen klantgegevens bij een verkoop (anonieme verkopen)
- De klant kan alleen cash betalingen doen aan de kassa
- Normaal heeft elk artikel een streepjescode die kan gescand worden aan de kassa. Omdat wij niet beschikken over een scanner gaan de artikelcodes intypen (in een tekstvak) en houden we de artikelcodes eenvoudig
- Alle artikelen hebben een btw percentage van 6%

Database

De artikelen (met code, omschrijving, artikelgroep, verkoopprijs (btw in) en actuele voorraad zitten ofwel in een tekstbestand ofwel in een Excel bestand (naar keuze van de winkelier). Deze artikelen worden bij het opstarten van de app ingelezen in een in memory database (hashmap) die gedurende gans de sessie wordt gebruikt. Bij het afsluiten van de app worden de voorraadgegevens van de artikelen in het tekst of Excel bestand bijgewerkt. Artikelen toevoegen/ verwijderen, prijsaanpassingen, … gebeuren niet via de app maar rechtstreeks in het tekst/excel bestand.

User interface

Je maakt een JavaFX project aan. Je schrijft zelf all JavaFX instructies (niet laten genereren door een tool!)
Maak eenvoudige maar toch aantrekkelijke (gebruiksvriendelijke) schermen!

We maken 2 vensters (stages) die tegelijkertijd zichtbaar zijn op het computerscherm (zie labo dobbelsteen oefening betreffende observer design pattern).

In **venster 1** (bedoeld voor kassier/admin) zijn er 4 tabs :
Tab1: het eigenlijke kassascherm
Bevat een tekstvak voor het intypen van de artikelcodes van de artikelen uit de winkelkar van een klant. Telkens als een code is ingetypt en op enter gedrukt, wordt dit product toegevoegd aan de lijst van gescande producten (in een tableview) en wordt de totale prijs van de verkoop bijgewerkt (zichtbaar op het scherm)

Tab2:
Bevat een lijst (tableView van alle artikelen uit de in memory database). De artikelen zijn alfabetisch geordend. Deze tab wordt alleen gebruikt om artikelen te raadplegen. Na elke

verkoop (als alle artikelen uit de winkelkar van een klant zijn gescand) wordt de voorraad van de artikelen op deze tab bijgewerkt.

Tab3:

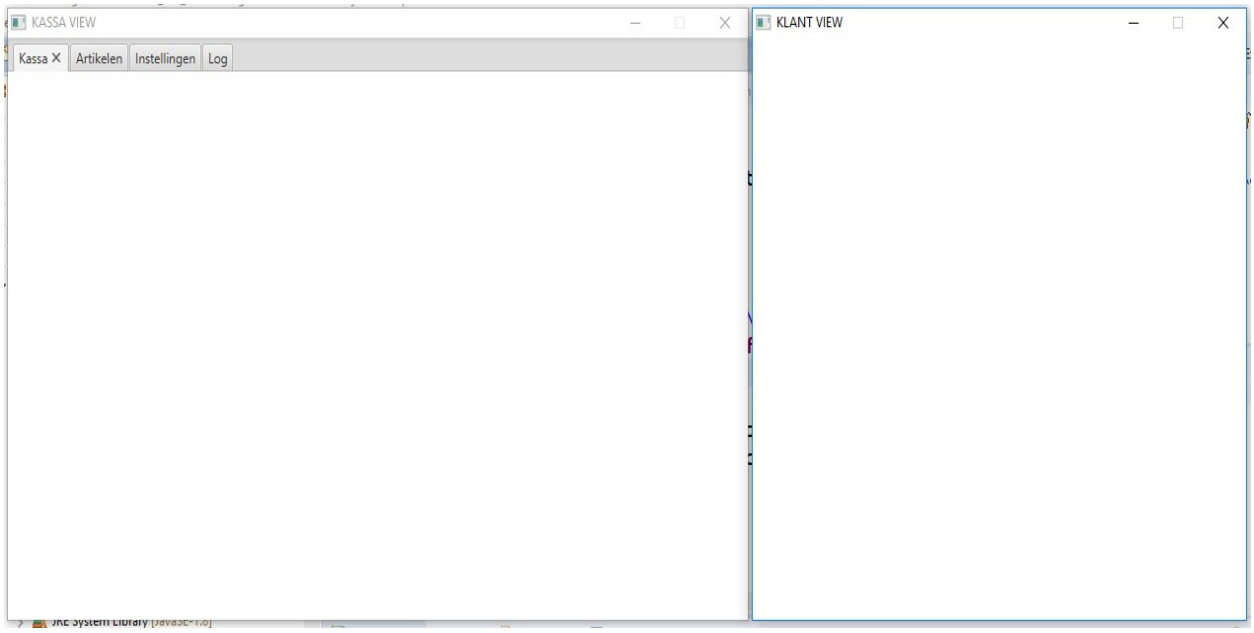
Hier kan de winkelier instellingen voor de app registreren:

- artikelen bijhouden in tekst of Excel formaat
- toepasbare kortingstrategie + korting%, kortingdrempels, …
- layout kassabon

Tab4:

Log - > geeft een overzicht van alle verkochte artikelen met datum en tijdstip. Log gegevens worden na elke verkoop aangevuld. De log gegevens hoeven bij afsluiten van de app niet te worden opgeslagen!

Venster 2 (bedoeld voor de klant) geeft een lijst van alle gescande artikelen uit de winkelkar van de klant (per gescand artikel wordt deze lijst bijgewerkt) en de totale prijs van de gescande artikelen (wordt ook per gescand artikel bijgewerkt)



Deze schermen zijn gemaakt met de code in StartCode.zip. Je mag deze code gebruiken in je project.

4. Stories

Op de volgende bladzijden vind je de verschillende *User Stories* die de gewenste functionaliteit beschrijven.

Het voornaamste doel van de opdracht is dat je kan illustreren dat je de SOLID design principes en de besproken OO patronen kan toepassen. Mocht het je niet lukken om de volledige opdracht klaar te krijgen, weet dan dat wij vooral hierop zullen beoordelen.

- **Anticipeer op eventuele uitbreidingen**. Zo kunnen er later andere soorten vragen of andere vormen van feedback gevraagd worden.
- Implementeer het updaten van views aan de hand van het **Observer pattern**.
- Voorzie minstens een **Facade** voor de functionaliteit aangeboden door de spellogica (=model).
- Hou het **MVC** patroon in gedachten. Doe dit zoals uitgelegd in The MVC pattern_Example_Important.docx (les 6 uit cursusdocumenten van Toledo cursus)
- Bedenk of en waar je de andere design patronen die we besproken hebben kan gebruiken (**Strategy**, **Factory**, **Adapter**, **Singleton**, **Decorator**, **State**).
- Denk ook aan andere zaken die aan bod zijn gekomen tijdens de lessen (**Enum**, **reflection**, **properties**).

Story 1. Overzicht artikelen tonen

Beschrijving

Maak een in memory artikel database klasse die alle artikel uit het artikel.txt tekstbestand inleest in een hashmap. Deze artikelen worden getoond in tab2 (artikelen) in een tableview met als kolommen de artikelcode, de omschrijving, de artikelgroep, de prijs en de actuele voorraad. De artikelen toon je in alfabetische volgorde van omschrijving. Er moeten geen artikelen kunnen worden toegevoegd, verwijderd of worden aangepast via je app . De actuele voorraad van de artikelen uit een kassaverkoop moet na afsluiten van elke kassaverkoop van een klant wel worden aangepast (automatisch).
Voorzie een load (inlezen tekstbestand) en een save (schrijven tekstbestand) methode in je artikel database klasse. De load methode geeft een ArrayList van Artikel objecten terug en de save methode schrijft een ArrayList van Artikel objecten weg.

Opmerkingen

1. Gebruik het meegeleverd artikel.txt bestand (met symbolische codes, omschrijvingen en groepen)!

2. Omdat het lezen en schrijven van tekstbestanden altijd verloopt volgens een vast stappenplan maak je een TekstLoadSaveTemplate klasse (gebruikt makende van het Template pattern).
   Je ArtikelTekstLoadSave klasse erft hiervan over en implementeert alleen nog de abstracte methodes uit de template klasse. Later kunnen andere klassen (voorbeeld een tekstbestand met klantgegevens van deze template klasse gebruik maken)

3. Voorzie in je architectuur tevens dat de in memory database later kan vervangen worden door voorbeeld een relationele database (de code om artikelen uit een relationele database te gebruiken hoef je niet te implementeren, wel de architectuur (met strategy pattern daartoe voorzien)

Story 2. Overzicht artikelen tonen met keuze uit tekstfile of Excel file

Beschrijving

Idem als story 1 maar nu moet je ook de mogelijkheid hebben om artikelen uit een Excel bestand te kunnen inlezen.
Maak een LoadSave strategy interface met een load en save methode en refactor je code uit story 1 door gebruik te maken van het **strategy design pattern** (later gaan we artikelen misschien ook willen lezen uit een serialized bestand, …)
Voor het werken met Excel bestanden maak je in je oplossing gebruik van 2 jar bestanden :

- jxl-2.6.12.jar (een algemene jar om met Excel bestanden te werken)
- ExcelPlugin.jar (een jar die gebruik maakt van de vorige jar en een klasse bevat met code om de gegevens van een Excel werkblad te lezen en te schrijven)
- ExcelPlugin.java (de java code van de klasse uit vorige jar – Aan deze code mag je niets wijzigen!)

Door gebruik te maken van het **adapter design pattern** zorg je ervoor dat de ExcelLoadSaveStrategy (adapter klasse die gebruik maakt van ExcelPlugin) kan ingepast worden in de LoadSave strategy van je app.
In tab3 (instellingen) kan de winkelier de opslagstrategie (tekst of Excel) kiezen. Zijn keuze wordt opgeslagen in een properties file. Bij het opstarten van de app wordt gebruik gemaakt van de gekozen strategie.
Maak gebruik van het **Factory design pattern** (indien mogelijk met reflection) voor het aanmaken van de juiste LoadSave strategie.
Maak **singleton** klassen van je factories

Opmerking

Gebruik het meegeleverd artikel.xls bestand (met symbolische codes, omschrijvingen en groepen)!

Story 3. Registreren van een kassaverkoop van een klant

Beschrijving

Het registreren van kassaverkopen gebeurt in tab1 (kassa)
Voor alle artikelen uit de winkelkar van de klant typen we in een tekstvak de artikelcode in (vereenvoudiging voor het scannen van de streepjescode van het artikel). Telkens wanneer je een code intypt en op enter drukt

- wordt het artikel opgezocht in de artikeldatabase en indien gevonden wordt van dit artikel de omschrijving en de prijs toegevoegd in een tableview. Het aantal is altijd 1 (als er 3 stuks van het zelfde artikel wordt verkocht typen we 3 maal de code en verschijnen er 3 lijnen voor dit artikel in het overzicht)
- wordt het totaal te betalen bedrag opgehoogd en getoond (in een label)
- indien artikelcode niet gevonden wordt toon je de boodschap “niet bestaande code”

Story 4. Overzicht van gescande artikelen aan klant tonen

Beschrijving
Telkens wanneer de kassier een artikel scant (code intypt) wordt op het klantenscherm (Venster 2) de artikelomschrijving, het aantal (=1) en de prijs aan de artikellijst toegevoegd. Wanneer een zelfde artikel een tweede keer gescand wordt zal dit niet worden toegevoegd aan de lijst maar zal het aantal opgehoogd worden. Het totaal te betalen bedrag wordt ook op dit scherm getoond en wordt bij elke scan bijgewerkt.
Maak hier gebruik van het **Observer design pattern**

Story 5. Verwijderen van een artikel uit de kassaverkoop

Beschrijving
De kassier moet een artikel uit de lijst van gescande artikelen kunnen verwijderen. Het totale bedrag moet dan worden bijgewerkt en tevens moet het klantenscherm worden bijgewerkt

Story 6. Een kassaverkoop on hold zetten

Beschrijving

Wanneer een klant aan de kassa merkt dat hij nog één of meer artikels moet gaan bijhalen in de winkel en er al een wachtrij van klanten aan de kassa is moet de kassier de reeds gescande kassaverkoop (lijst van gescande artikels) on hold kunnen zetten (deze kassaverkoop wordt bewaard) maar het kassascherm en klantenscherm wordt klaargemaakt voor het registreren van aankopen van een volgende klant.
Er kan maar één kassaverkoop on hold gezet worden.
Een on hold kassaverkoop moet (na afrekening van een andere klant) terug actief kunnen gezet worden en afgehandeld worden.

Story 7. Berekenen van korting bij een kassaverkoop

Beschrijving
De winkelier moet de mogelijkheid hebben om flexibel van dag tot dag te kunnen beslissen welke korting acties hij wenst toe te passen.
Enkele voorbeelden van korting acties:
Groepkorting: vb. 5% korting op alle artikelen van een bepaalde groep
Drempelkorting: vb 5% korting op een aankoopbedrag hoger dan 100 €
Duurstekorting: vb 25% korting op duurste artikel uit winkelkar

Zo zijn er nog vele andere acties mogelijk. De cijfers in de acties (5%, 25%, 100€) moeten instelbaar zijn.

Maak gebruik van het **strategy design pattern** voor het flexibel implementeren van deze korting strategieën
Zorg ervoor dat de winkelier de toe te passen kortingsactie kan kiezen en de nodige cijfers (vb korting%)kan ingeven (in tab3 – instellingen).
Gebruik **factories, enums, reflection** indien mogelijk

Story 8. Een kassaverkoop afsluiten

Beschrijving
Als alle artikelen uit de winkelkar van een klant gescand zijn drukt de winkelier op de *AFSLUIT* knop en wordt op zijn kassascherm en op het klantenscherm naast het totale bedrag (zijnde de som van de prijzen van de gescande artikelen) ook de totale korting en het te betalen bedrag (totale bedrag – korting) getoond.

Story 9. Een kassaverkoop betalen

Beschrijving

Scenario 1

Als de klant genoeg geld bijheeft en betaalt drukt de kassier op de *BETAALD* knop.

- Het kassierscherm wordt dan klaargezet voor een nieuwe verkoop te registreren of voor het terughalen van een verkoop on hold.
- De voorraad van de artikelen wordt bijgewerkt in de database en de actuele voorraad in tab3 wordt aangepast.
- De verkoop wordt gelogd: aan de lijst in tab4 –log wordt een lijn toegevoegd met als inhoud: datum en tijdstip van betaling – totaal bedrag – korting – te betalen bedrag. In ons vereenvoudigd systeem moeten de verkopen niet opgeslagen worden in een database. Dus als we de app terug opstarten in de lijst van logs terug leeg.

Scenario 2

Als de klant niet genoeg geld kan de kassier nog gescande artikels verwijderen (bedragen in beide vensters worden dan aangepast) en dan alsnog op de *BETAALD* knop klikken.

Scenario 3

Als de klant geen geld bijheeft kan de kassier op de *ANNULEER* knop klikken. Kassa en klantvenster worden dan klaargezet voor verwerken verkoop van een volgende klant, voorraad wordt natuurlijk niet bijgewerkt, verkoop wordt niet gelogd.

Maak voor de Verkoop klasse (de klasse voor een actuele verkoop van een klant) een UML toestandsdiagram (state diagram) met alle mogelijke toestanden van een verkoop en alle mogelijke events (van story 3 tot en met story 9). Pas het **State design pattern** toe om dit diagram te implementeren.

Story 10. De klant krijgt een kassabon

Beschrijving
Als de klant betaald heeft zal er een kassabon worden afgeprint. In de vereenvoudigde app hoef je niet te printen, je toont de kassabon op de console.

Op deze kassabon staat steeds volgende informatie:
Een lijn met labels (omschrijving, aantal en prijs). Prijs is de prijs voor 1 aantal zonder korting
Daaronder de aangekochte artikelen
Daaronder de betaalde prijs (inclusief korting)

Voorbeeld van een kassabon

```
Omschrijving            Aantal   Prijs
*****************************
Artikel1                   1   12.50
Artikel17                  3    3.75
Bartikell6                 1    2.50
*****************************
Betaald (inclusief korting) : 24.50 €
```

Optioneel kan de winkelier kiezen om een aantal header en footer lijnen toe te voegen aan de kassabon. Hij moet deze optionele lijnen aanvinken in tab 3 instellingen.
Mogelijke optionele headerlijnen (af te printen bovenaan de bon boven de vaste info van hierboven:

- Een algemene boodschap (die de winkelier kan ingeven in tab 3 – vb Zondag 31 maart open van 9.00 tot 12.00
- Datum en tijd dat bon is afgeprint

Mogelijke optionele footerlijnen (af te printen onderaan de bon onder de vaste info van hierboven:

- De totale prijs zonder korting met daaronder de totale korting
- De betaalde prijs exclusief BTW (Btw is 6 % voor alle artikelen) met daaronder het BTW gedrag
- Een algemene afsluitlijn (vb Dan je voor je aankopen)

Deze optionele instellingen voor de kassabon aangevinkt in tab 3 (instellingen) moeten natuurlijk worden opgeslagen (properties file).

Maak gebruik van het **decorator design pattern** om de kassabon af te drukken.
Zorg voor een goede uitlijning van de kolommen van de kassabon zodat de bon goed leesbaar is.