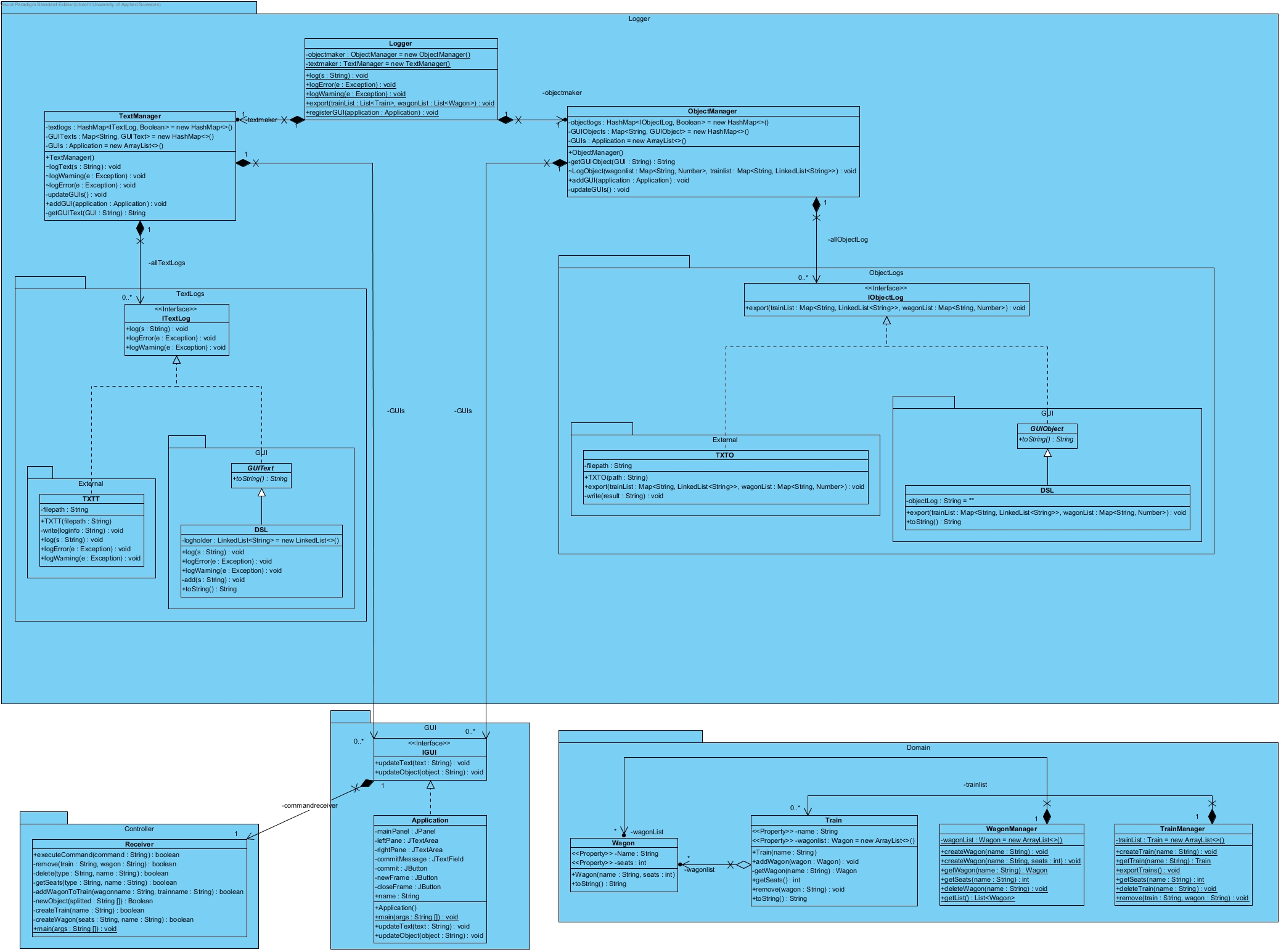
Motivatie Code



(Klassendiagram: model.jpg)

**Voor een beter kwaliteit en inzoomen, zie model.jpg.**

# GUI

*Interface IGUI zorgt voor makkelijk gebruik van verschillende GUI’s zonder dat er wijzigingen buiten deze layer nodig is. Door een observer pattern met Textmanager en ObjectManager worden de GUIoutputs up-to-date gehouden. Dit samen zorgt ervoor dat er voor het toevoegen van een andere soort GUI alleen een klasse hoeft aangemaakt worden aan IGUI en (als de binnenkomende Text en Object anders gegeven moet worden) een klasse aan GUIText en GUIObject. (Open-close principle)*

IGUI vormt een laag tussen de GUI’s en de back-end applicatie. Hierdoor kunnen er verschillende GUI’s simpel op de applicatie gebouwd worden door de interface te implementeren, er zijn dan geen aanpassingen in de applicatie meer nodig.

Doormiddel van een observer pattern vanuit ‘Textmanager’ en ‘Objectmanager’ wordt de tekst in de gemaakte GUI actueel gehouden. Als de informatie anders weergegeven moet worden kan dit aan GUItext of GUIobject gehangen worden. (Strategy pattern & open-close principle)

# Controller

*Receiver krijgt commando’s binnen van GUI’s, decodeert ze en stuurt de acties door naar de juiste domein-klassen.*

Als een gebruiker in de GUI een command geeft (new/add/delete etc.) dan krijgt de ‘Receiver’ klasse deze binnen. De commando’s worden dan door deze klasse gedecodeerd en vervolgens doorgezet naar de verantwoordelijke domeinklasse.

# Domain

*Trainmanager en WagonManager gebruiken de factory pattern op train en wagon voor het beheren van objecten van deze klassen. Dit zorgt er ook voor dat alle klassen maar één taak hebben (Single-responsibility Principle)*

*Acties die effect hebben op trains of wagons sturen een commando door naar logger om de overzichten (object genoemd) up te daten.*

Het factory-pattern wordt gebruikt door ‘TrainManager’ en ‘WagonManager’, zodat het beheren van de treinen en wagons gescheiden is van de objecten. Hierdoor hebben de klassen maar één taak (Single-responsibility Principle)

De overzichten worden geüpdatet doormiddel van commando’s naar de logger die de overzichten bijwerkt.

# Logger

*Logger stuurt de Textoutputs door naar de textmanager en de Objectoutputs naar de objectmanager. Ook GUI observer registraties worden hier doorgestuurd naar beide managers. Dit samen zorgt dat de logger de enige aanspreekpunt is voor klassen buiten deze layer, die de rest van deze layer aanstuurt.*

Logger krijgt Textoutputs binnen, deze worden doorgezet naar de ‘TextManager’. Objectoutputs worden doorgezet naar ‘ObjectManager’. Ook de geregistreerde observer GUI’s worden naar beide managers meegestuurd. Alles samen zorgt ervoor dat ‘Logger’ het enige aanspreekpunt is voor buitenaf en de aansturing van de laag verzorgt.

*Omdat Textoutputs en Objectoutputs voor GUI’s en externe files dezelfde kennis (mogen) hebben en binnenkrijgen, worden deze (als ITEXTLog en IObjectLog) bijgehouden door dezelfde manager (TextLogs en ObjectLogs natuurlijk wel gescheiden) (Dependency Inversion principle). Omdat Externe log-schrijvers direct wegschrijven en GUIlogs intern opslaan om door te sturen naar de GUI’s worden GUI’s gescheiden door ze een sub-interface GUIobject/log te geven (Liskov substitution principle & Interface segregation principle). Bij nieuwe outputs sturen de managers de informatie gelijk door naar de subscribed GUI’s via het observer-pattern.*

Textoutputs en Objectoutputs mogen voor GUI’s en externe bestanden dezelfde kennis hebben en ontvangen. Bij de manager (ITextLog en IObjectLog) wordt aangegeven of een item extern of naar de GUI moet worden weggeschreven (Dependency Inversion principle). GUI’s hebben een extra (eerdergenoemde) strategy pattern laag ertussen zitten, omdat GUI-logs intern opslaan om door te sturen en externe log-schrijvers het direct wegschrijven (Liskov substitution principle & Interface segregation principle). De manager stuurt nieuwe outputs ook gelijk door naar de subscribed GUI’s. Dit gebeurt met het in “GUI” uitgelegde observer pattern.

*IObjectLogs en ITextLogs zijn adapters voor de data om ze op de juiste manier weg te schrijven of door te geven aan de GUI. (Open-closed principle).*

IObjectLog en ITextLog volgen een adapter pattern om de ontvangen informatie op de juiste manier weg te schrijven of door te zetten aan de GUI. (Open-close principle)

**De SOLID Principles zijn vaak meerdere keren op dezelfde plekken terug te vinden.**