Cornateanu Laurentiu
ID 10105187
Object Oriented Design and Development (QHO543)
Assessment AE1 Object-Oriented Development

Cornateanu Laurentiu
ID 10105187
Object Oriented Design and Development (QHO543)
Assessment AE1 Object-Oriented Development

Table of Contents

Cornateanu Laurentiu
ID 10105187
Object Oriented Design and Development (QHO543)
Assessment AE1 Object-Oriented Development

Analysis

Use Cases

I started the analysis phase by reading through the case scenario and developing a functional overview use case diagram for the entire system. This involved identifying high level functionality and which actors would need to interact with them. The high-level functionality was then decomposed into smaller more specific chunks of functionality.

Each use case was then elaborated, in order to start identifying the entities involved in the system and how they would need to interact. These elaborations formed the basis of the sequence diagrams generated.

I opted to use Enterprise Architect (EA) as we are starting to use this tool at work, and I wanted to gain a better understanding of its capabilities. As I opted to use EA to model my software, I don't have robustness diagrams in the form demonstrated during lectures. I was, however, able to apply a UML stereotype to the entity lifelines in order to get EA to render them with the appropriate symbols. Additionally, EA doesn't allow individual methods to be represented as elements in the diagrams.

Using the entities generated during the elaboration, I composed a number of class diagrams to illustrate the relationships between each class.
As an aide to modelling the software I created a dummy interface for the JSP Views containing a display() and button_click(action).

Given more time, the sub-components, such as the controller, ticket machine and gate machine would be decomposed further. However, for the sake of this proof of concept I opted not to do this.

Assumptions

Based on the scenario, a number of assumptions were made including:

- Ticket Machines would be registered, on site, with the central controller using the requested REST API
- It was assumed that each ticket machine would use a hardware module with a unique ID embedded into it.
- A 'plug and play' methodology was also adopted for the installation and registration of the ticket machines.
- The controller would be deployed to localhost/lunderground
- The ticket machine would be deployed to localhost/ticket
- The gate machine test page would be included in the ticket machine webapp at localhost/ticket/gate

Cornateanu Laurentiu
ID 10105187
Object Oriented Design and Development (QHO543)
Assessment AE1 Object-Oriented Development

Design

The software uses various tried and tested design patterns to make the code base easier to manage, extend and test.
Examples include:

- Using a model view controller (MVC) design to separate the concerns of getting user input and displaying results from the underlaying business logic and from the model used to represent the system state.
- Using an additional Data Access Layer to abstract the interaction with the specific technology used to persist entities.
- Using a REST interface to allow operations offered by the controller to be accessed across the network
- Using the observer pattern to reduce the time between the ticket machine configuration being modified and the latest configuration stored on the controller being requested.
- The façade pattern was used to hide relatively complex operations behind a much simpler API. The controller façade took this a step further and used a faceted interface so that the same class could offer operations against 2 different interfaces, each tuned for a specific actor. The LundergroundServiceFacade is used to offer up the common controller operations and the DeveloperFacade added additional features to aide with initialisation and testing that normal uses should not have access to.
- A singleton pattern is used for many of the DAO objects to help reduce the chances of concurrent operations from various users conflicting and causing race conditions.

Cornateanu Laurentiu
ID 10105187
Object Oriented Design and Development (QHO543)
Assessment AE1 Object-Oriented Development

Implementation

For the controller and ticket machine web app implementations I opted to use the Spring framework to develop using the MVC pattern. As Spring MVC was not covered in the main lectures this required that I go and learn how to use it myself. This turned out to be quite a complex subject but enabled me to test large sections of the business logic that would have otherwise require significant manual effort.

I also chose to use the Spring framework in order to implement the REST API, this also turned out to be a good decision as it closely followed the Spring MVC style making the REST controller relatively simple.
In addition to using the Spring framework, I also investigated good practice for REST API implementation. Microsoft had a particularly succinct set of guidelines to follow, the particular points I implemented included:

- Including a version in the API path to make it easier to make improvements without impacting end users
- Using singular nouns when working with an entity
- Using appropriate HTTP methods to represent the different CRUD operations, e.g. GET to request an entity, POST to create an entity, PUT to update an entity, DELETE to delete and entity.

I tried to mainly use maven features to make it easier to move between IDEs, however towards the end I did realise that I was relying on an IntelliJ feature which overrides the web app deployment URL. I have opted not to fix this due to how close to the submission time it was noticed.

Cornateanu Laurentiu
ID 10105187
Object Oriented Design and Development (QHO543)
Assessment AE1 Object-Oriented Development

Testing

Unit

I started creating unit tests for the code to be implemented from the very beginning of the project. The design documents, especially the sequence diagrams, made it quite easy to see the seams, and the expected interactions, between the various layers. As such I heavily relied on them to guide happy path testing.
I also used a code coverage tool to report on the sections of code exercised, this gave me insights into other areas that should also be sanity checked.
Over time I added automated building, execution of unit tests and measurement of code coverage to GitHub using their Actions. This way as I pushed changes to GitHub I was able to detect any breakages within a few minutes, this saved me on a few occasions, especially when adding new modules to the project or adjusting the dependencies being relied on.

I also used the unit testing framework (Junit) to execute a number of slower integration tests that accessed the filesystem to generate database items and carry out XML marshalling to files. This too helped me to quickly identify when I'd made changed to the DAO implementations that were not quite right.

User Acceptance Testing

For user acceptance testing (UAT) I created a test plan from the identified use cases. For the project I mainly focused on the happy path due to the time it takes to manually run through the test plan. Given more time I would have liked to investigate using an automated tool, such as Selenium, to automate the UAT.
I would not have been able to run these directly on GitHub as they only allow the deployment of static web pages at this point in time. A separate CI/CD server, such as Jenkins, would be required to monitor the git repo and run the tests locally.

Quality

In addition to functional testing I also investigated, and implemented, GitHub's multi language SuperLinter Action, this helps to identify where I'm not following recommended best practices. I was unable to find a GitHub action that allowed static code analysis which would have enabled me to identify more serious issues, such as null pointer usage, much earlier on.

Cornateanu Laurentiu
ID 10105187
Object Oriented Design and Development (QHO543)
Assessment AE1 Object-Oriented Development

Deployment

The Ticketing system has been implemented as a maven multimodule project.

As such, it can be built from the command line using ‘mvn build’

This will build all of the modules, including the public / private crypto keys.

As per the assumptions mentioned above it is expected that

- The controller be deployed localhost/lunderground
- The ticket machine be deployed to localhost/ticket
- Both be available on port 80

I have included the IntelliJ project files and run configurations which will allow:

- All unit tests to be run
- Code coverage to be calculated
- Generate crypto keys
- Deploy and run the whole ‘lunderground’ system. NOTE: IntelliJ will only open a browser page to the ticket machine, you will need to manually open the controller page.

Cornateanu Laurentiu
ID 10105187
Object Oriented Design and Development (QHO543)
Assessment AE1 Object-Oriented Development

Reflection

On the whole, I've learnt a huge amount from this project. In the past I've tended towards bloated classes that are dealing with too many concerns.
Looking back, I believe that I underestimated the number of moving parts in the project and I was too ambitious with some of the extra technologies I opted to use. However, I feel like I have a much better appreciation of the different options that are available to better structure my future projects.

There are areas of the documentation I'm not happy with, mainly as I started running out of time to develop diagrams to the level of quality I started with, as a result I do think that the scope of the documentation is not as wide as it could be.

Cornateanu Laurentiu
ID 10105187
Object Oriented Design and Development (QHO543)
Assessment AE1 Object-Oriented Development

Appendix A. Use Case Diagrams

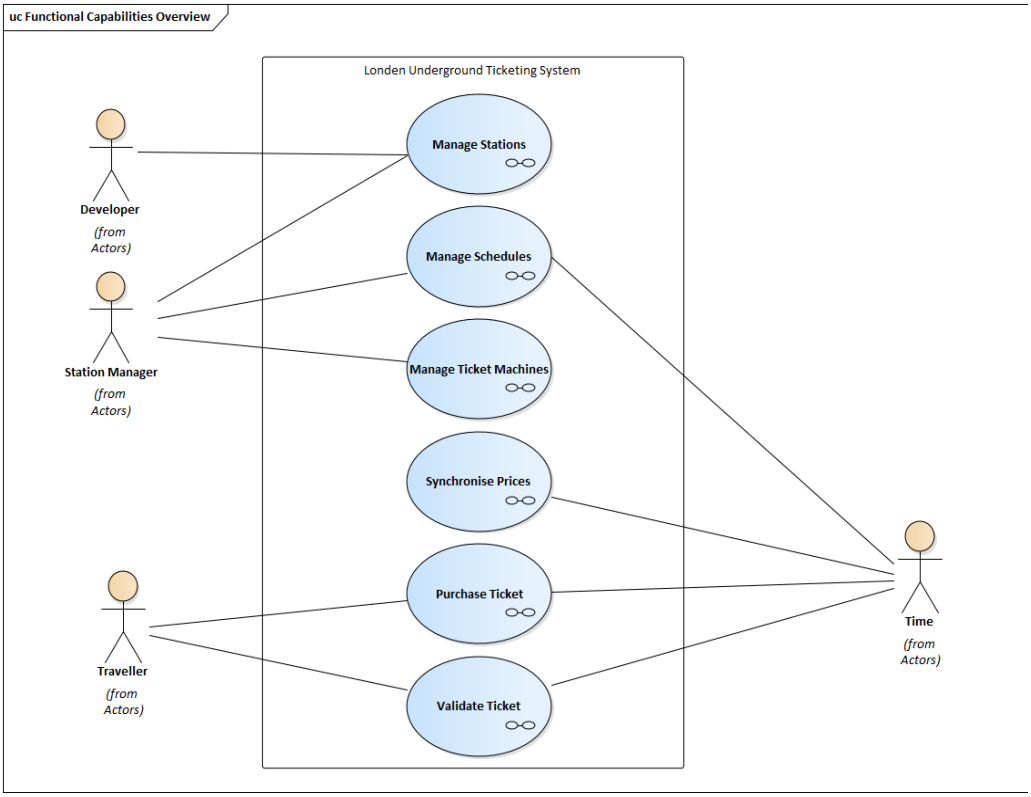


Figure 1: Function Overview

Cornateanu Laurentiu
ID 10105187
Object Oriented Design and Development (QHO543)
Assessment AE1 Object-Oriented Development

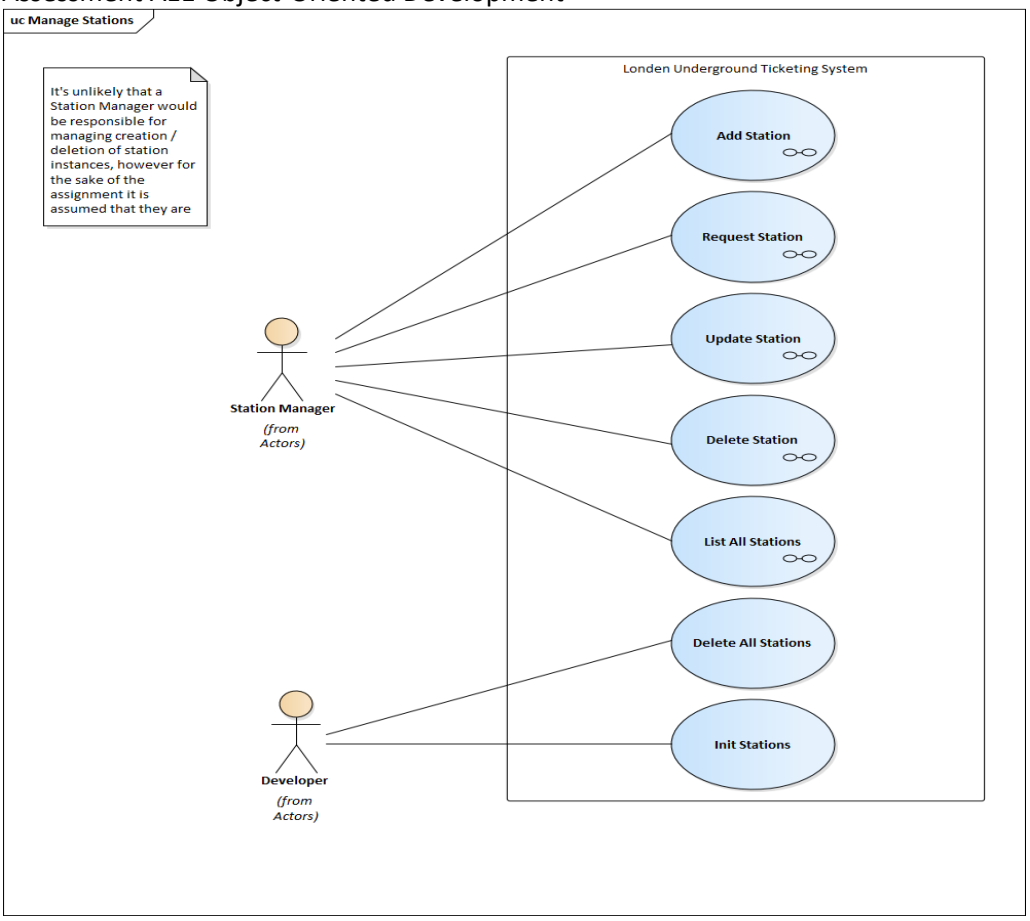


Figure 2: Manage Stations

Cornateanu Laurentiu
ID 10105187
Object Oriented Design and Development (QHO543)
Assessment AE1 Object-Oriented Development

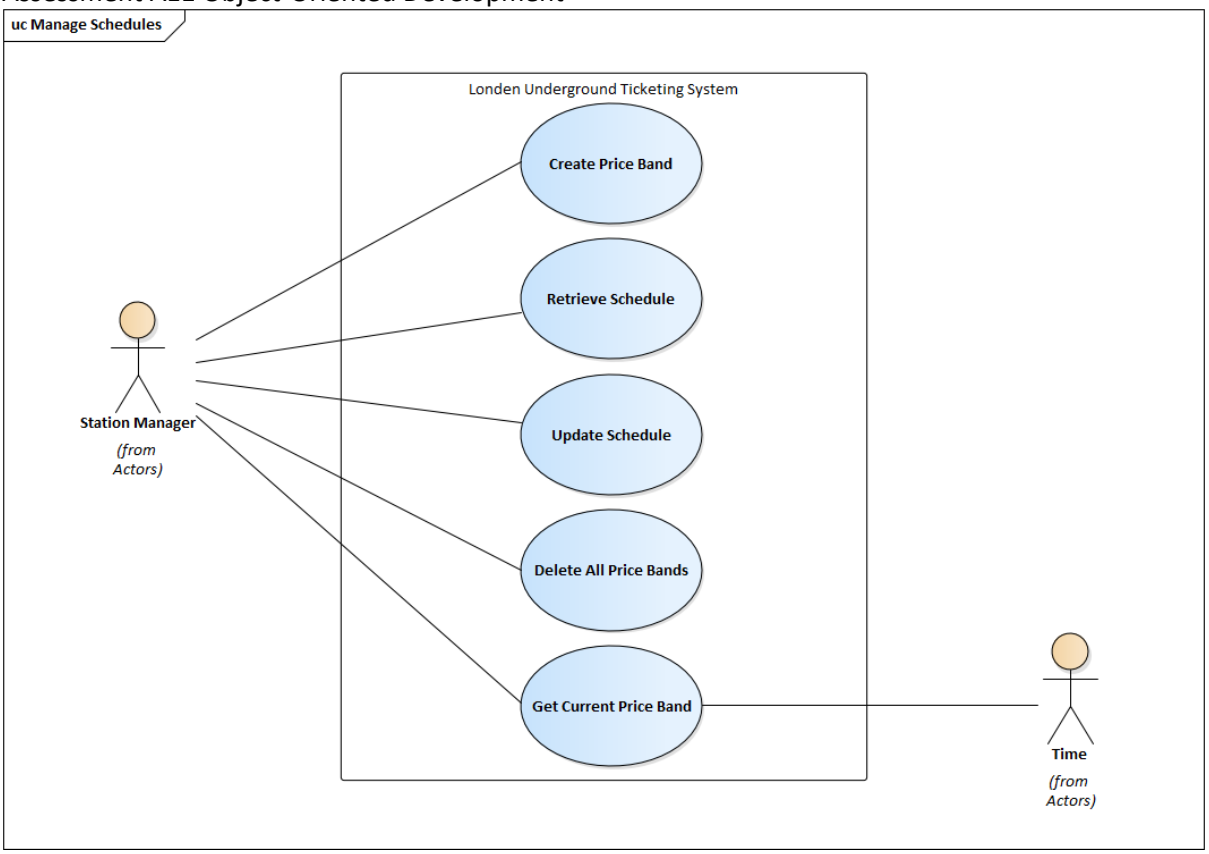


Figure 3: Manage Schedules

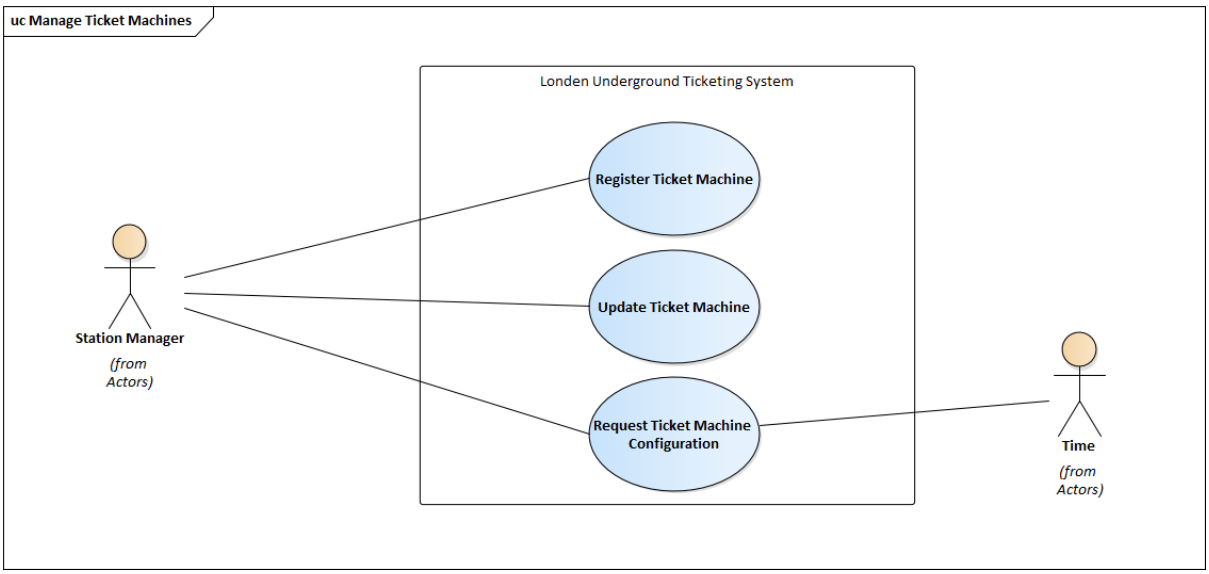


Figure 4: Manage Ticket Machines

Cornateanu Laurentiu
ID 10105187
Object Oriented Design and Development (QHO543)
Assessment AE1 Object-Oriented Development

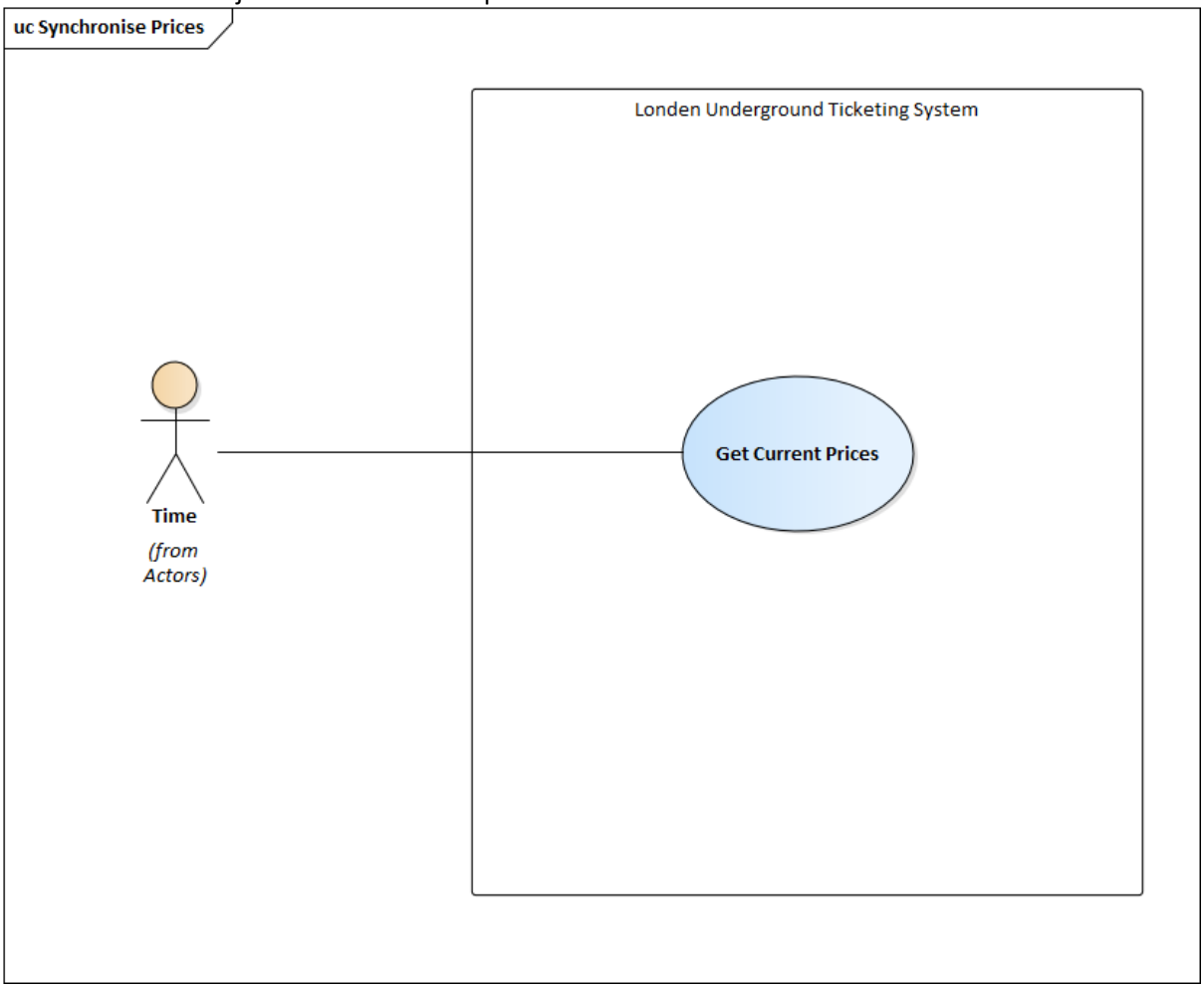


Figure 5: Synchronise Prices

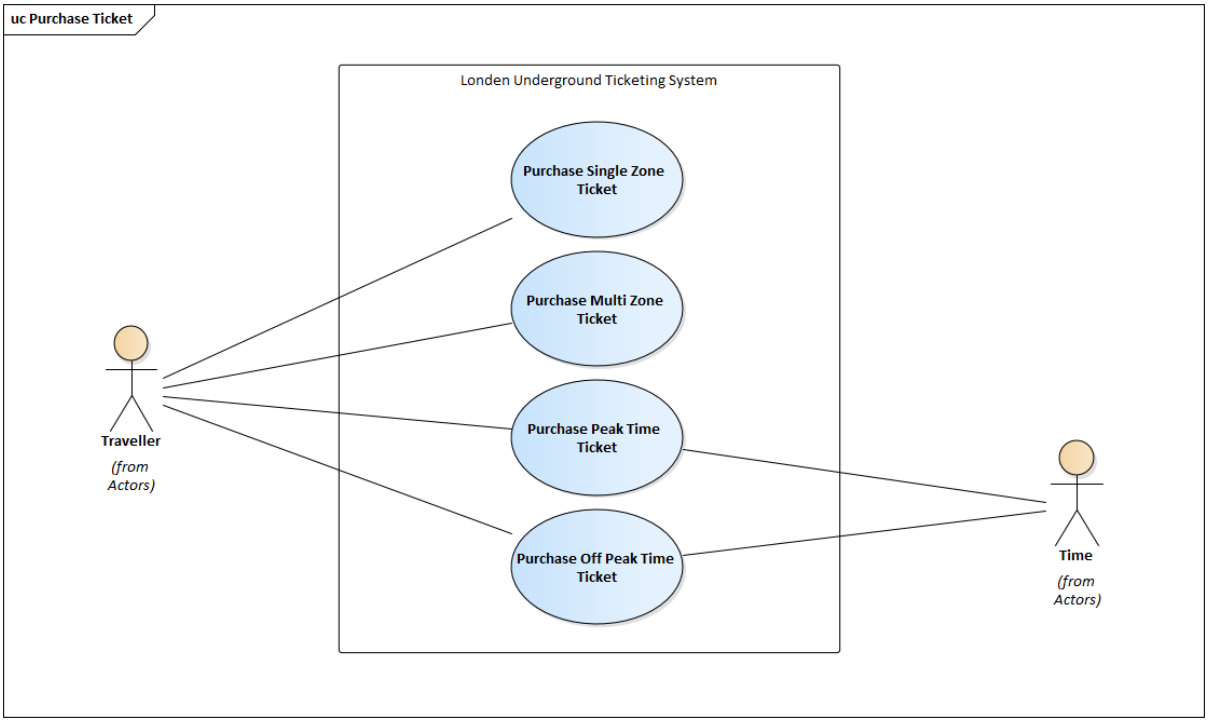


Figure 6: Purchase Ticket

Cornateanu Laurentiu
ID 10105187
Object Oriented Design and Development (QHO543)
Assessment AE1 Object-Oriented Development

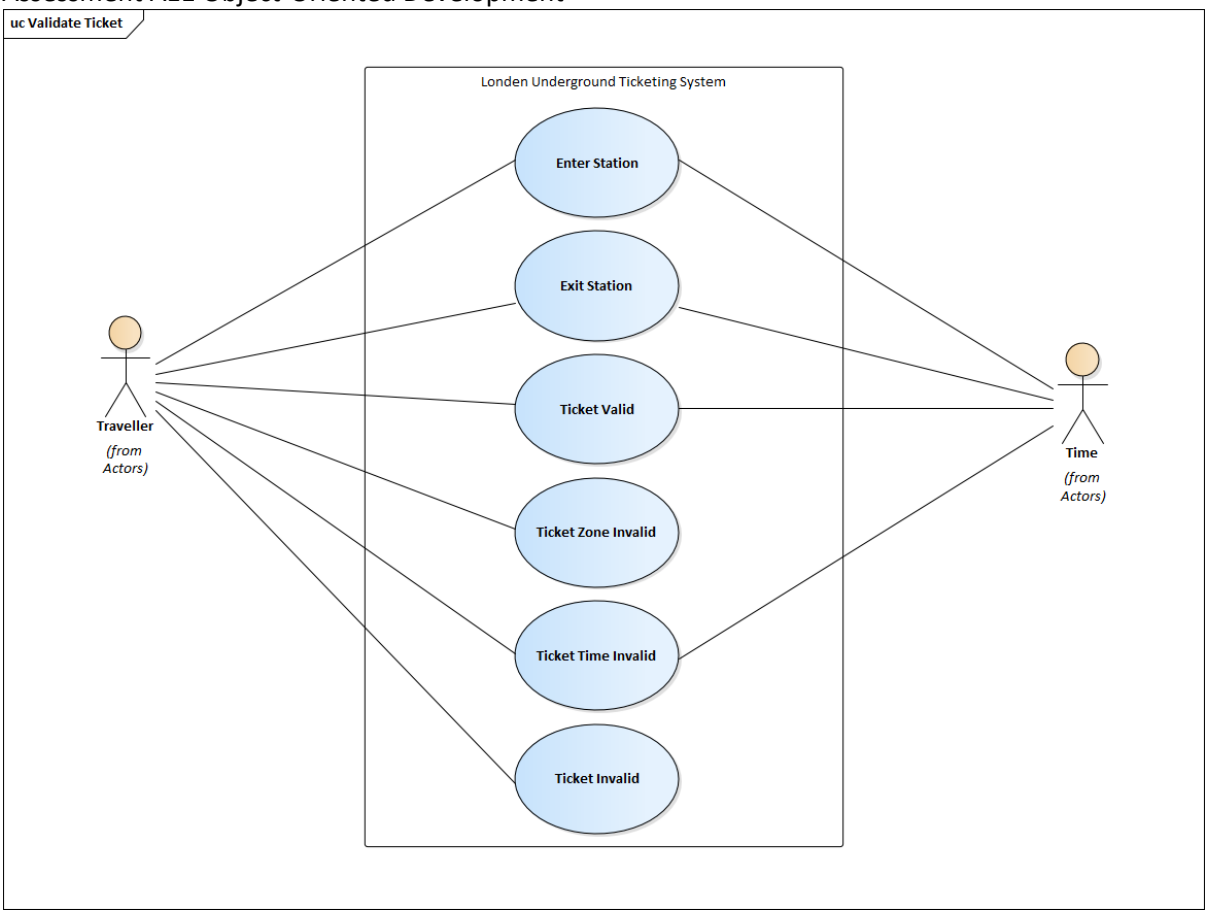


Figure 7: Validate Ticket

Cornateanu Laurentiu
ID 10105187
Object Oriented Design and Development (QHO543)
Assessment AE1 Object-Oriented Development

Appendix B. Use Case Elaboration Examples

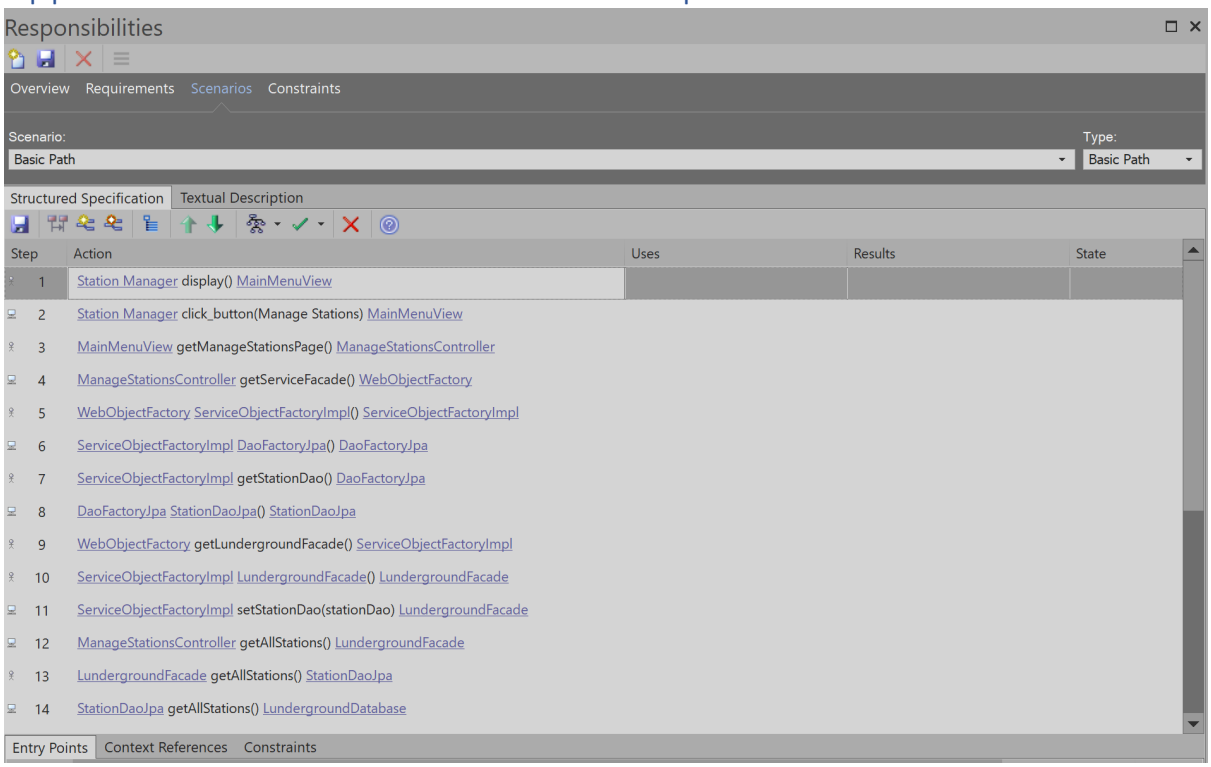


Figure 8: Add Station Elaboration

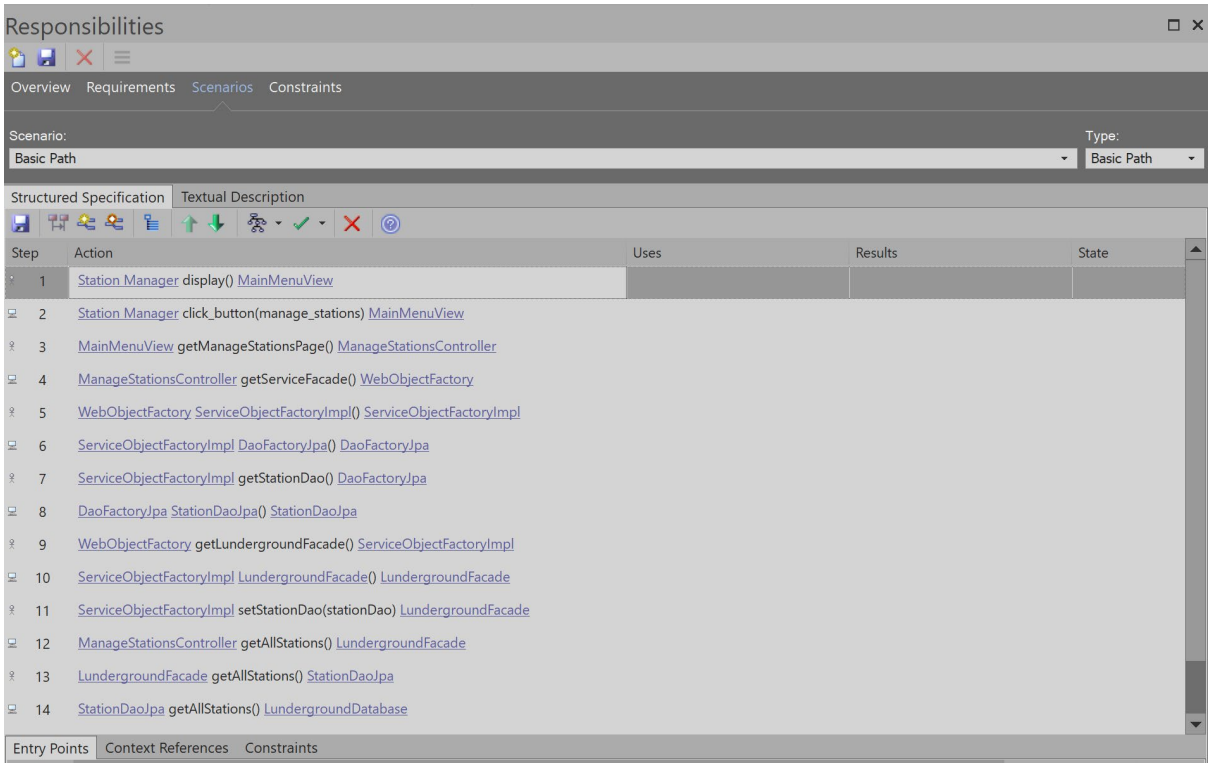


Figure 9: List Stations Elaboration

Cornateanu Laurentiu
ID 10105187
Object Oriented Design and Development (QHO543)
Assessment AE1 Object-Oriented Development

Appendix C. Sequence Diagram Examples

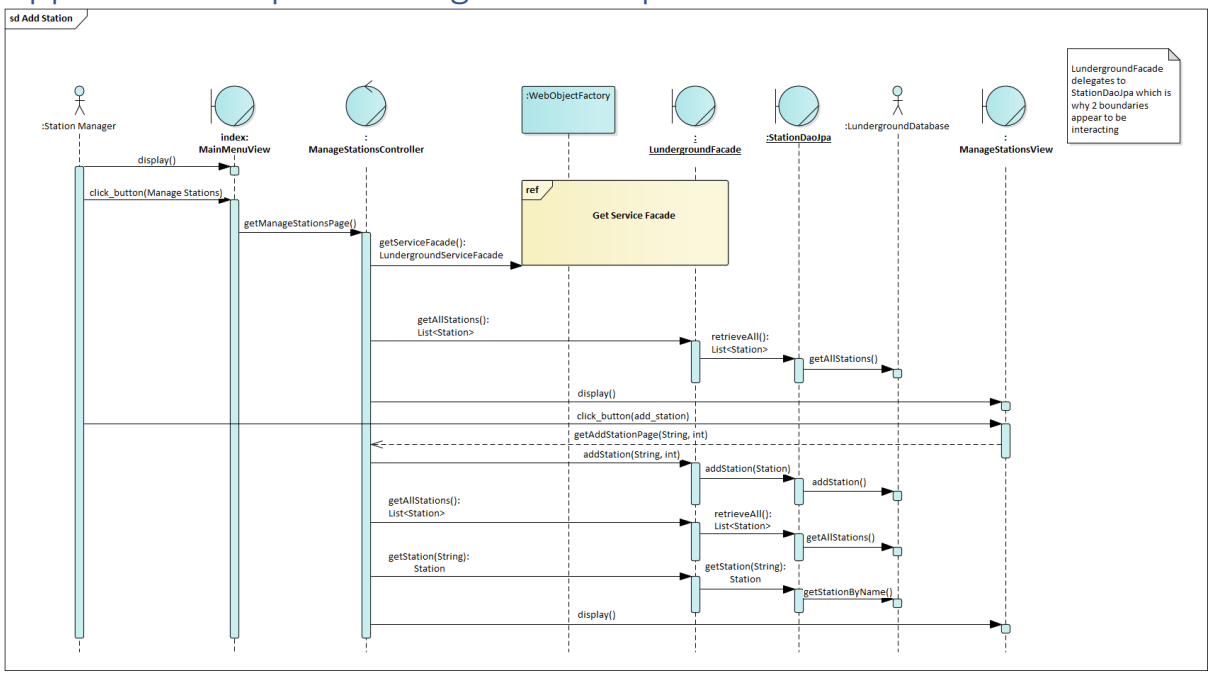


Figure 10: Add Station Sequence Diagram

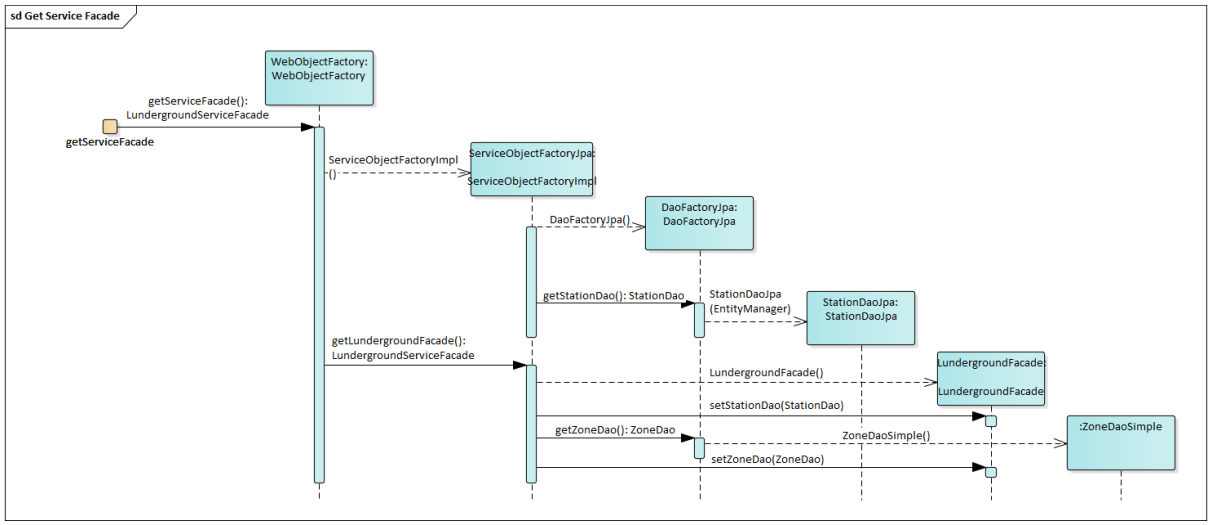
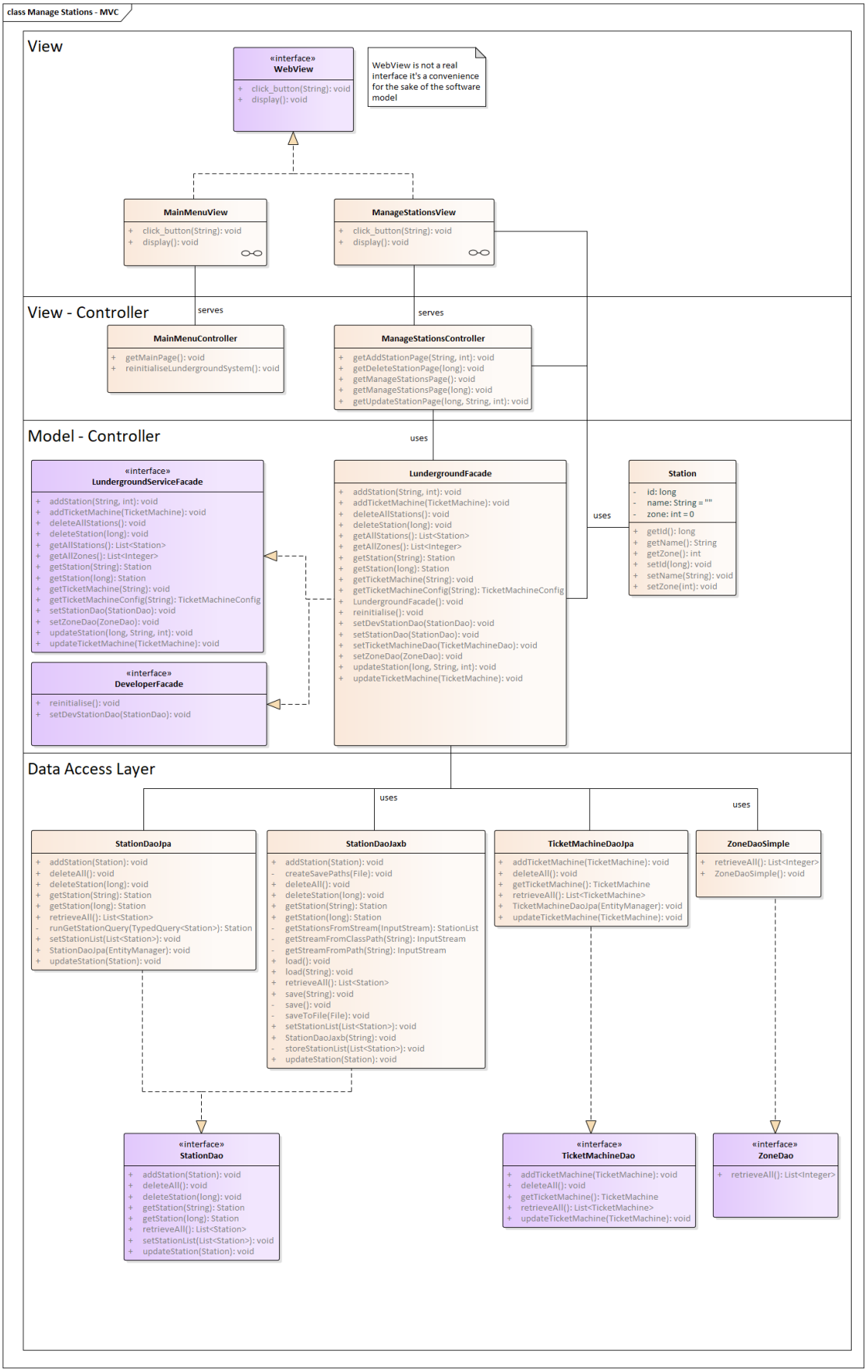


Figure 11: ServiceFacade factory sequence diagram

Cornateanu Laurentiu
ID 10105187
Object Oriented Design and Development (QHO543)
Assessment AE1 Object-Oriented Development

Appendix D. Class Diagrams



Cornateanu Laurentiu
ID 10105187
Object Oriented Design and Development (QHO543)
Assessment AE1 Object-Oriented Development

Figure 12: Controller service façade

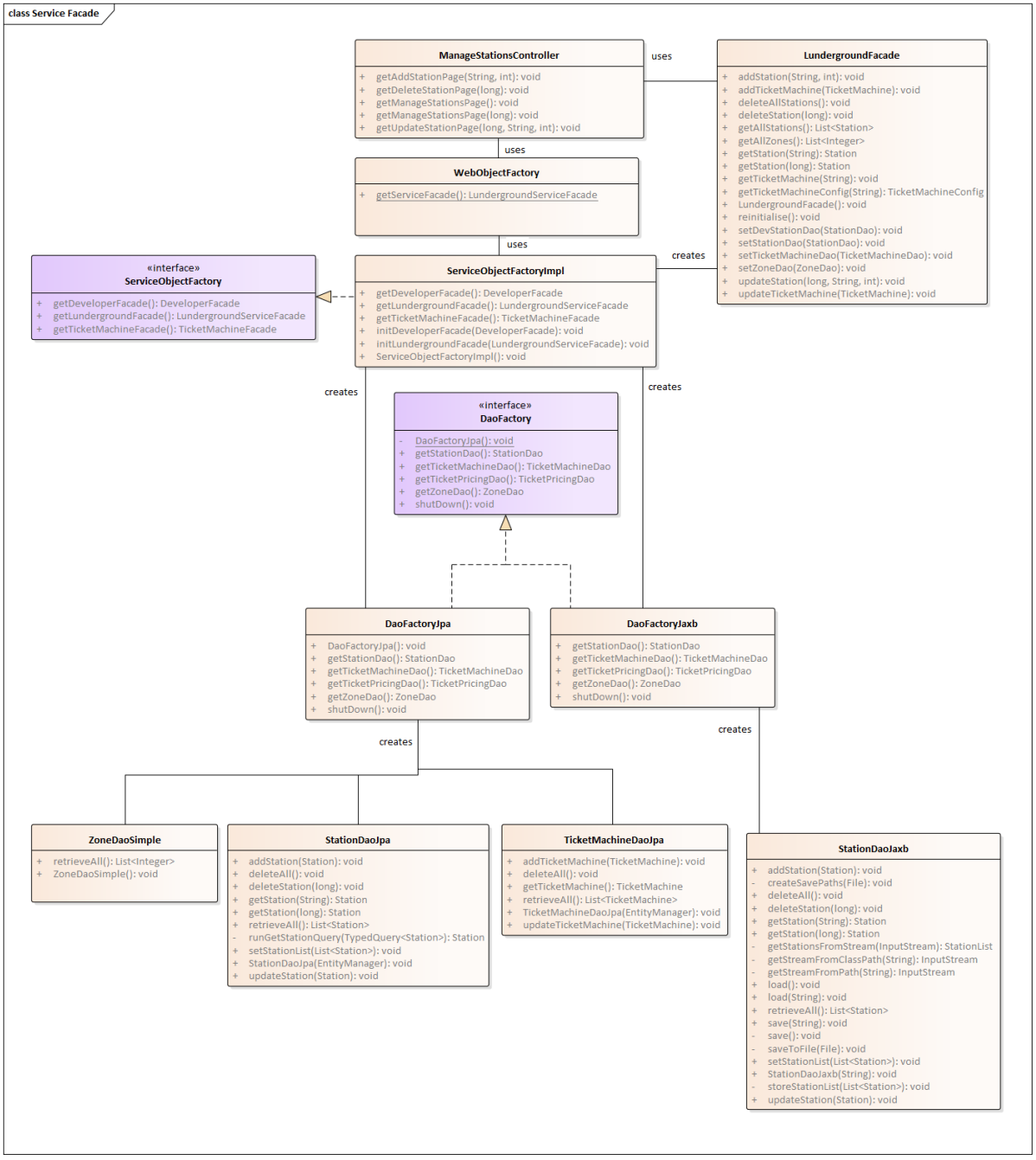


Figure 13: Creating object using factories