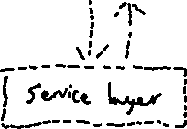
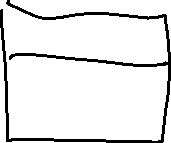
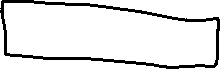
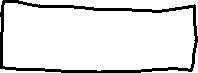
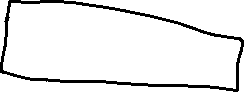
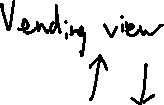
Initial MVC for vending machine (28/06/2022)



1. View.DisplayMenu()
2. GetUserInput() (Money in, and item requested)
3. VendItem(Item requested) returns Transaction DTO
4. If transaction is successful, deincrement stock and output change (call view.displayChange(double), converts to currency denominations)
5. If not, “**display a message indicating insufficient funds** and then **redisplay the amount the user had put into the machine**“ ask Eugene if we go straight to asking for a new item to vend or ask for more change
6. Head back to the start



Diagram

Description automatically generatedFlowchart:

**Above here are bits done before the start of programming, and below here is the review of the project.**

Spec requirements:

1. Apply spec to application

The program runs as intended with all requirements of the spec fulfilled, excluding 2:

* 1. MVC pattern – while MVC system design has been followed, a service layer was not added to the program. This was done to reduce time to completion and led to functionality bleeding into both the DAO and View layers. In the future, properly implementing the service layer and keeping business logic encapsulated within the service layer is probably for the best. Additionally, I did not use constructor-based dependency injection (at least not to my knowledge (I don’t fully understand it)). Need to recap and look into more either in my own time or by asking lots of questions about it in Zoom.
  2. Full set of unit tests – Some unit testing has been done, but with full disclosure in mind, I wasn’t sure how to select the methods/sections of the system to write unit tests for. This will be something I’ll bring up during a zoom meeting I imagine, alongside questions on the best way to instantiate all of the test variables (e.g. manually setting up treemaps for all tests in the ChangeTest file seemed cumbersome and I feel I’m missing a trick somewhere)

1. Dependency injection

See part 1a

1. MVC

Separate packages were built for dao, view, control, dtos. Control class handled main execution of program, while view handled CLI and dao handled file I/O. DTOs transferred data between these. There are a few instances where data is transferred outside of a DTO (i.e. transferring all items from itemstore to view in VendingController:line 39).

(I realise I’m putting in a lot of detail, the next parts of the spec are going to be covered either through code screenshots or a simple yes/no on whether it was done)

1. I/O operations

Graphical user interface, application, website

Description automatically generated

File I/O in VendingItemStorage and VendingAuditStorage

1. Service layer

See 1a.

1. Unit tests

See 1b.

1. Graphical user interface, text

   Description automatically generated with medium confidenceText

   Description automatically generatedBigDecimal

BigDecimal used in DTOs for monetary representation

1. Text

   Description automatically generatedLambdas and Streams

🡨

1. Enums

Text

Description automatically generated

1. Audit with DateTime

Text

Description automatically generated

Taken from VendingAuditStorage.SaveToFile

1. Explain Service Layer

The service layer contains the business logic of a system and allows the system to keep the logic encapsulated and out of the other layers.

1. Explain Unit Testing

A unit is a section of the system being developed. The idea of unit tests is that by breaking the program into these units and validating that each one of these performs correctly, the sum of the units (i.e. the whole system) will also perform correctly. Accomplishing this in its entirety would mean testing every function which is often not practical. However, creating unit tests for critical functionality will cover the majority of situations where invalid behaviour may occur.

1. Commented code, naming conventions

I believe I have followed naming conventions to a suitable degree. My comments within functions outline the flow of execution for these functions, and other comments are not too bad in my opinion.

Overall: fun project, would recommend.