CSSE220: Design Problems 1 - 2 Solutions

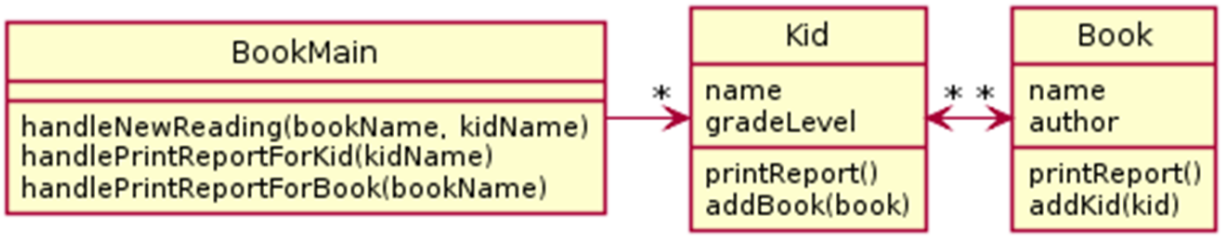
For maximum benefit, I encourage you to attempt to solve the problems yourself before peeking at this solutions document.

All of these problems relate to Design Principle 1 and 2

Table of Contents

Books (in-class exercise SETechniques)- Solution

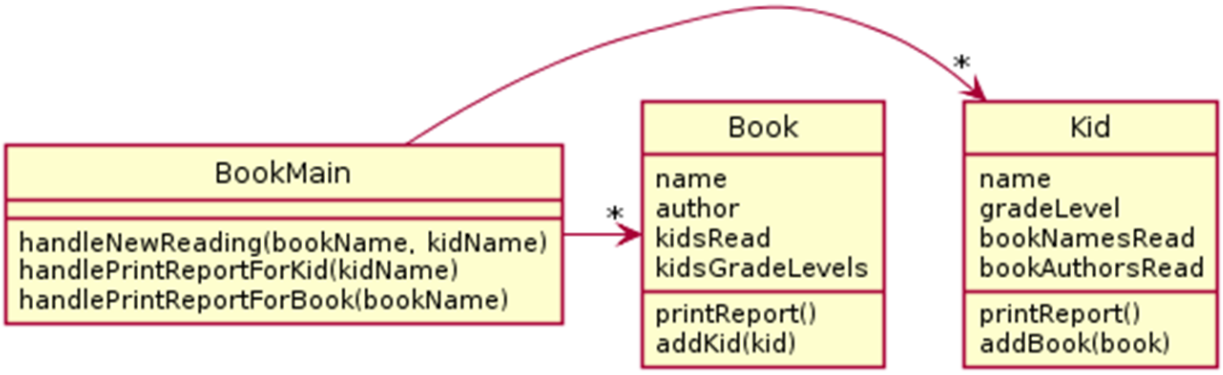
Bad Design A:



Problem: 1b - This design does not function.

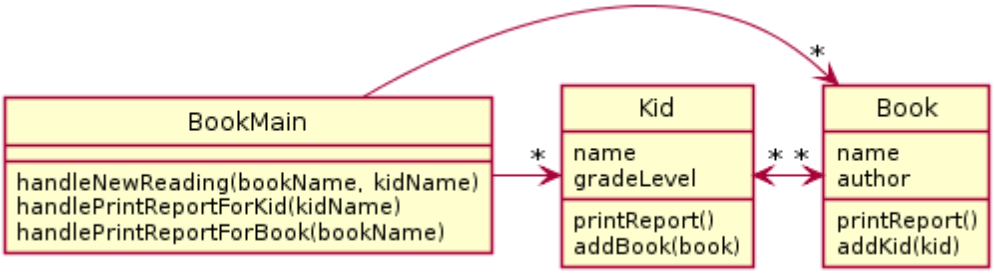
There is no (sane) way to look up a book for printing a report or for associating with a Kid.

Bad Design B:



Problem: 1c - This design functions but there is a very large amount of duplication – which in general we want to avoid. In particular, the author/title information in the kid is duplicated and the name/grade level information in the book is duplicated.

Good Solution:



All functionality fo rhandler methods easily provided.

Company Accounts (in-class exercise SETechniques)- Solution

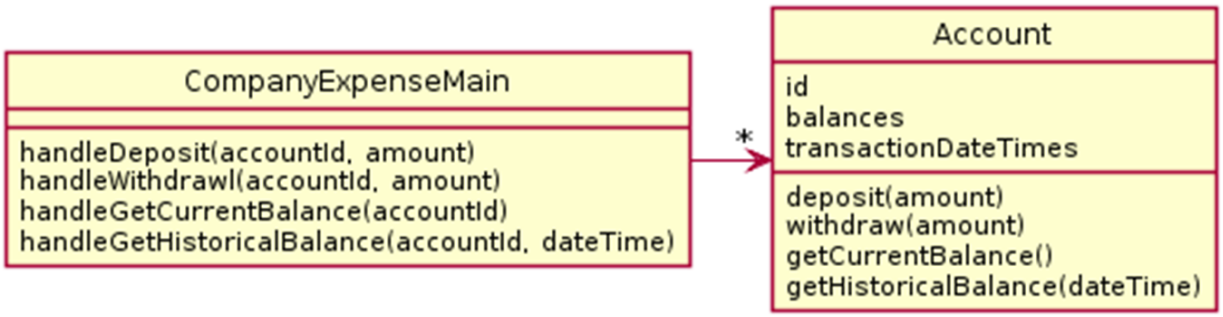
Design A:

1. Main has only one account, but the system needs to support many. How do we know that from this diagram?
2. Also, computing the data for historic balances is moderately hard.

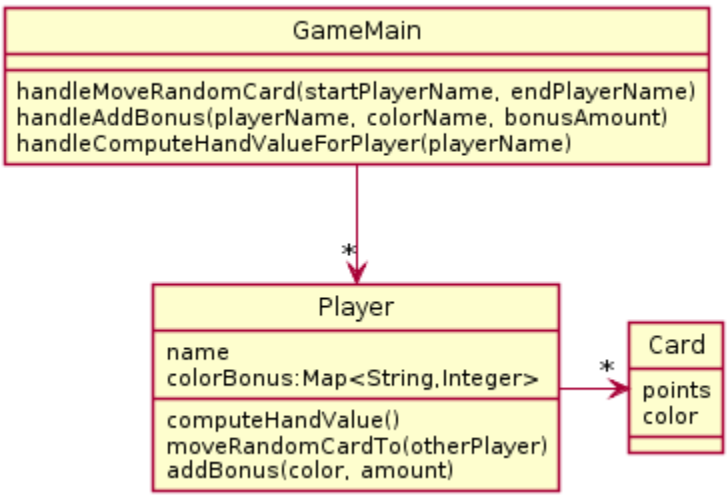
Design B:

This design does not function correctly.
Account does not have enough data to make getHistoricalBalance work.
oldBalances stores perhaps a list of balances? But the date of the transactions is not stored, so we can't look up the balance on a particular day/time.

Potential Solution:



Colored Cards (in-class exercise SETechniques)- Solution



Hour Tracker- Solution

Problems with A

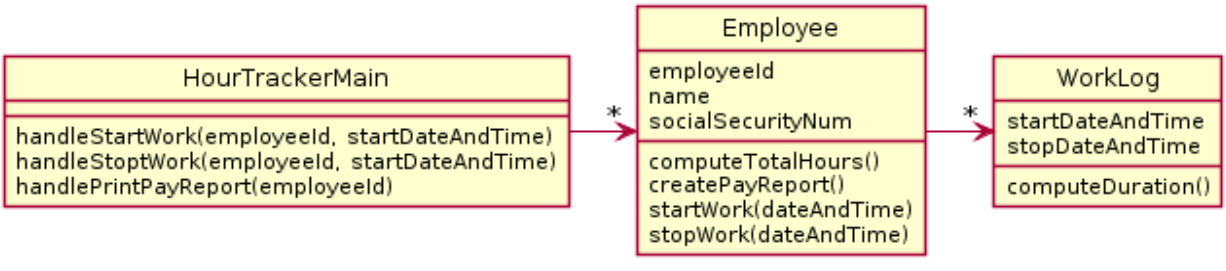
1(a or c). Design does not function correctly. A does not store name, social security #. Putting that in worklog would cause data duplication.
2a. Employee should become a class, it is a noun that holds data and needs to report information about itself

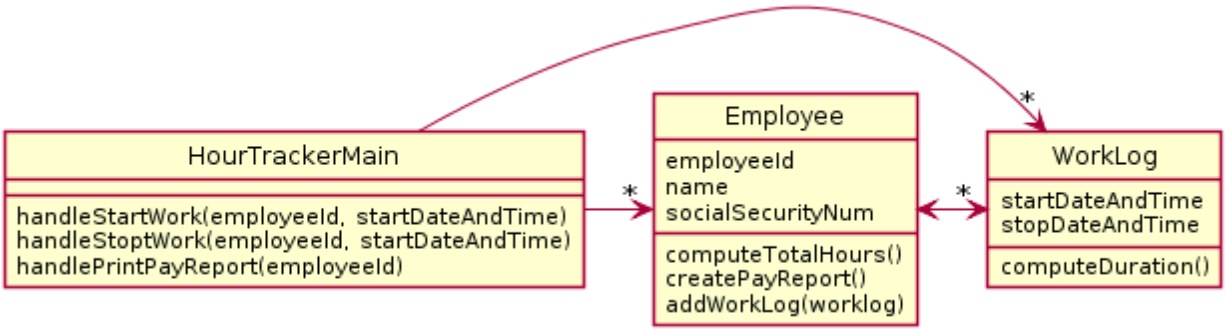
Problems with B

1 (a or c). Design does not function correctly. A worker might have many different start times and end times in a week - this data cannot be stored. Or if multiple employee objects are created for one employee, that's data duplication.

3a. Probably won't have this, since we hadn't discussed it yet, so don't penalize - this is just for our notes. Employee is too large and does too much (manages work log info and has to calculate it, when that data belongs to WorkLog)

Final Solution





Supercomputer- Solution

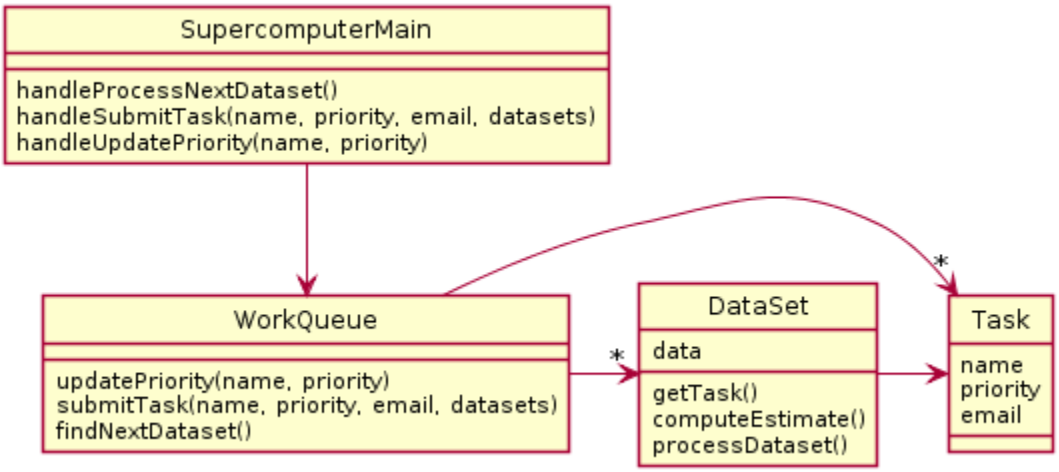
Problems With A

2c. This design has duplication of the taskName, priority, and email. As well as being bad in it's own right, this causes updating priority to require searching all datasets.

1a. No task

Problems With B

3a/b. WorkQueue is doing too much - DataSet should logically be responsible for computing estimates and processing.



I could live with Tasks containing datasets or even DataSet not existing at all and just being handled through task. Both have their pros and cons.

Libraries- Solution

- The method getTitle() in BookCopy cannot be implemented based on how the system is currently designed
- The fields title and authorName ought to be in BookCopy not BookRecord
- The field isBorrowed ought to be in BookCopy not BookRecord
- The field tagId ought to be in BookRecord not BookCopy

Sometimes, it might be nice to get a list of all the library branches in the city that own a particular book (byTitle). Let's add a method called "printAllLibrariesThatOwnBook(title:String)" to main. Assuming all the fields in this diagram are private, what additional methods will need to be added to the system in order to accomplish this feature?

Right now, if we wanted to list all the books that a particular library branch owns, it is a fairly intensive thing to compute. Modify the diagram above so that this feature is easy to compute.

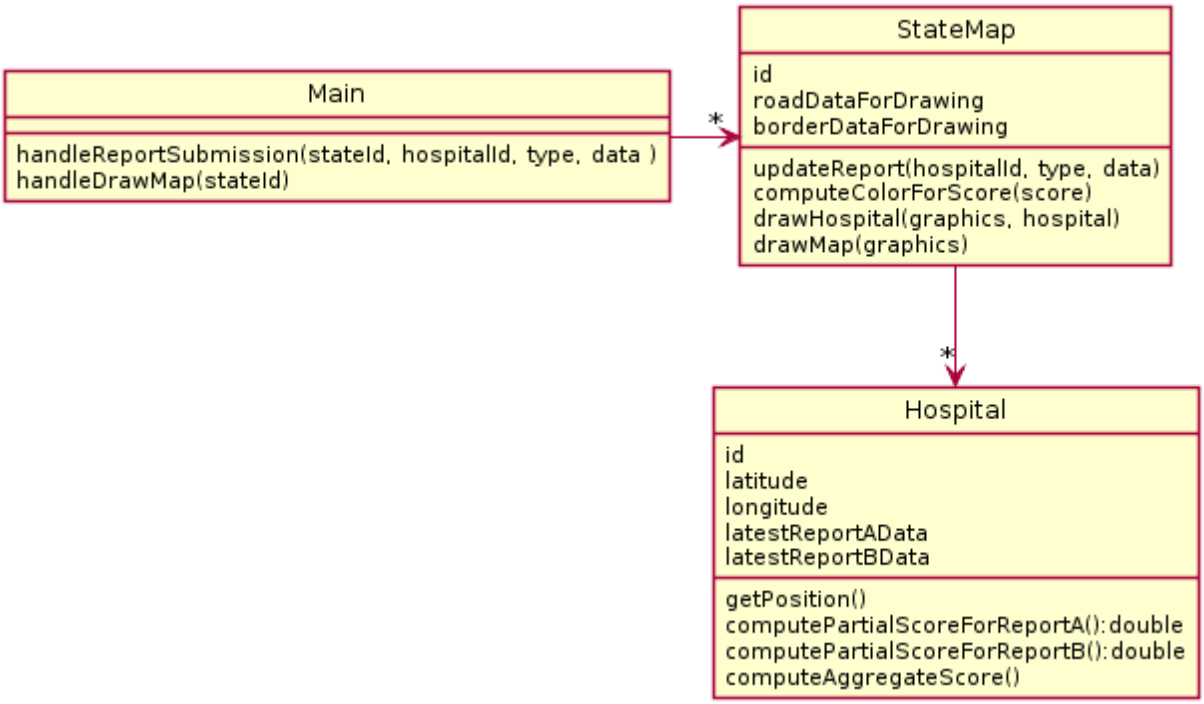
State Hospitals- Solution

Problems with A

State is doing 2 different things - drawing and dealing with combining reports into the aggregate score calculations. I'm not quite sure I'd allow "State is doing too much" for full credit. Thoughts?

Problems with B
Ask methods on hospital meaning it is not in control of its own data

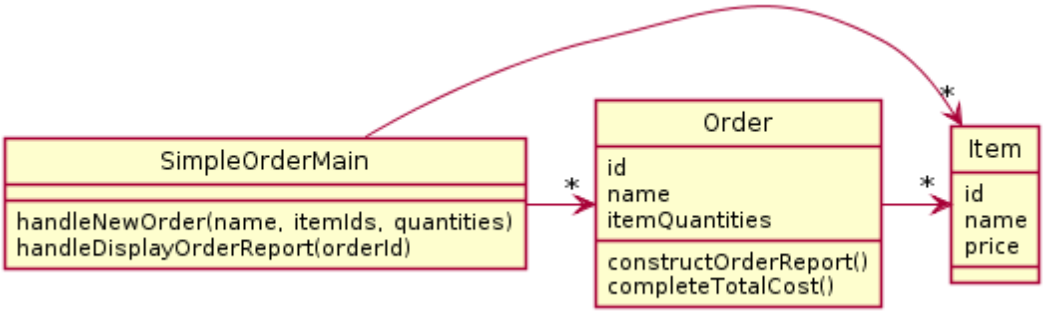
My solution



I think I'd be willing to accept keeping the StateData/StateMap distinction or have reports in their own classes like in A. But I think this is sufficient for what we need to do.

SimpleOrder- Solution

A. There is duplication of item names and prices every time an item is ordered.
B. The system cannot represent the quantity ordered



Notflix Billing- Solution

a) Solution A violates principle **1a**
b) because the Main program NotflixBillingSite only has a single customer and a single video. The site needs to track many customers and many videos.

c) Solution B violates principles **1c, 2a, 1b (one could also argue 1a probably)**
d) **1c** because the video data is duplicated by every customer that watches the same video; **2a** because Video is a noun that is not even represented; **1b** because if there is a video in the system that no customer has watched (like a very low-resolution version of some unpopular video), that information is lost; **1a** could be argued because the same reason as 1b is violated, we can't store a video unless a customer has watched it.
e) Design B