CSSE220: Example Design Problems
Relating to Principles 1 and 2

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

All of these problems relate to Design Principle 1 and 2

This document includes example problems and space for you to write in solutions. In every case, the instructions are:

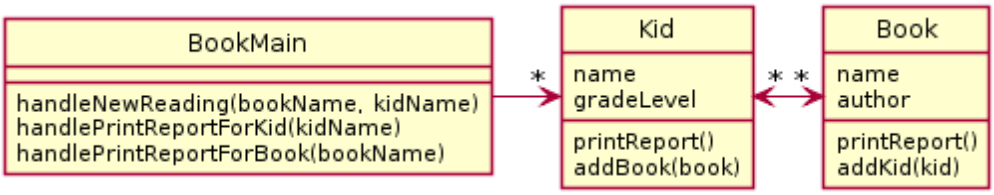
1. Identify the problems with Solution A & B using your design principles
2. Design a new solution that solves all problems

Table of Contents

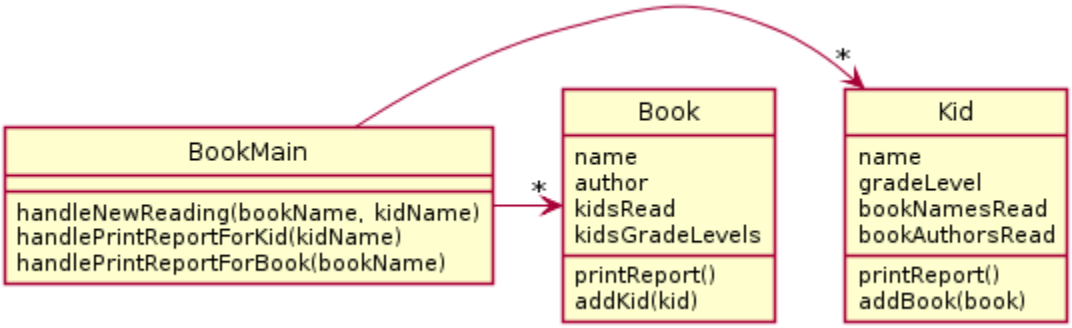
Books (in-class exercise SETechniques)

A website tracks books and the kids that read them. For each book the system stores the name and author. For each kid the system stores name and grade level. The teacher enters when a kid reads a particular book. It should be possible to print a report on a book that includes all kids who have read a particular book (with their grade level). It should be possible to print a report on a kid that includes the books (with authors) a particular kid has read.

Solution A



Solution B



1) Describe the problems with Design A and B

2) Design your own solution

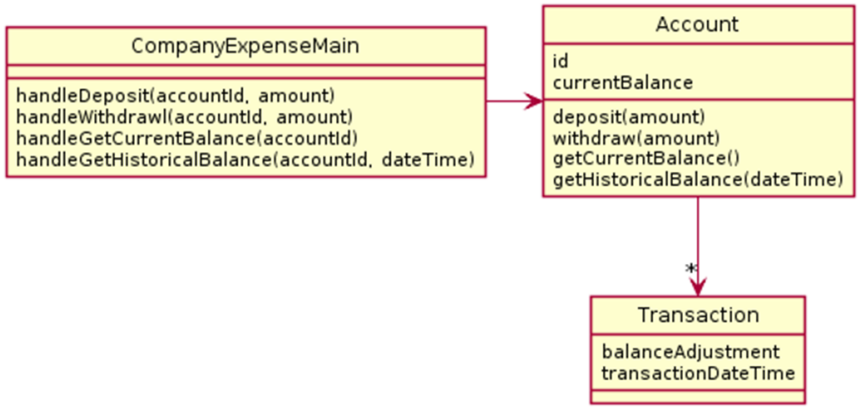
Company Accounts (in-class exercise SETechniques)

A particular company keeps a variety of different accounts for its projects. Each account has an account number and a balance. When a deposit or withdrawal occurs, the transaction occurs immediately, and the current balance should be updated. The system should support getting the current balance.
The system should also support getting the balance as it existed at any date/time in the past. Note the input historical date/time may not correspond to a particular transaction time - e.g., if the system had a balance of $1 at 1 p.m. and then was changed to $2 at 3 p.m., a request for the balance at 2 p.m. should return $1.

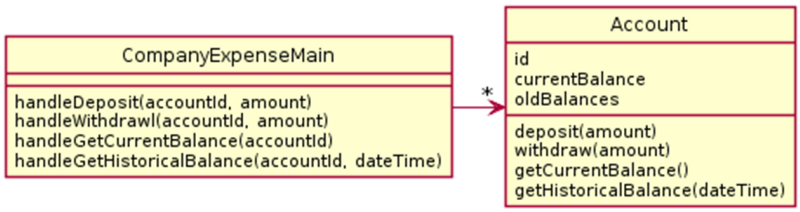
To Do #1 Identify all the *primary nouns*
- •A *primary noun* is a noun in the problem that has *attributes* *(other nouns*)
- •Nouns that designate *actors* of the system (i.e. The *user* can click…) can be **excluded**

To Do #2 Write down the *attributes* (*other nouns)* associated with the *primary nouns*
To Do #3 Identify all the *verbs*
To Do #4 Identify which *primary nouns* are worked on by the *verbs*
To Do #5 Design a system using UML to handle this problem

Design A



Design B

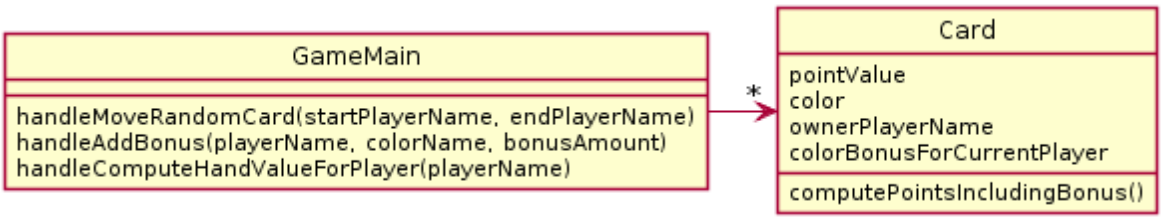


These designs do not function correctly. Why?

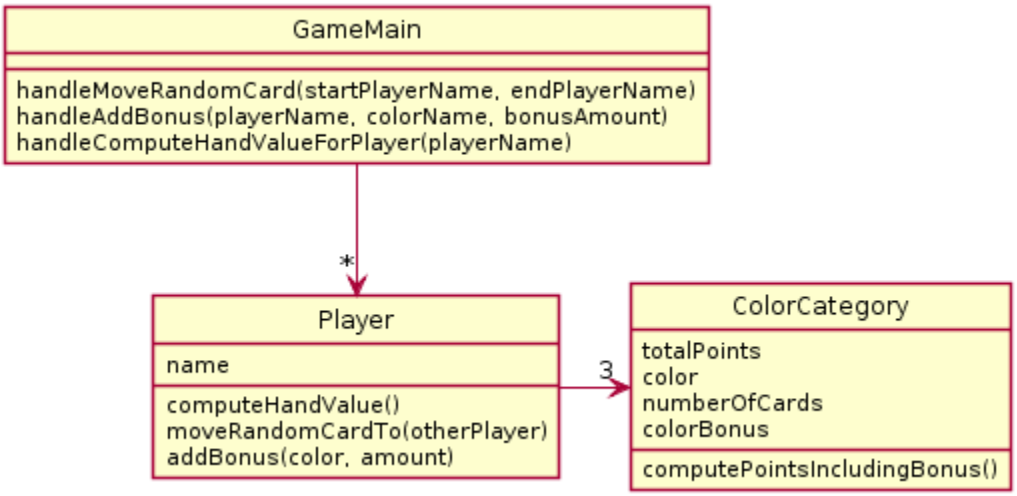
Colored Cards (in-class exercise SETechniques)

In a particular card game, players have hands of cards. Each card is worth some points and also has a color (red, blue, green). During play, players accrue bonuses that mean cards of a particular color are worth bonus points. During play, sometimes a random card is selected from one player's hand and moved to another player's hand. At the end of game, it is necessary to compute the total points for each player's hand. Here are 2 possible designs:

Solution A



Solution B



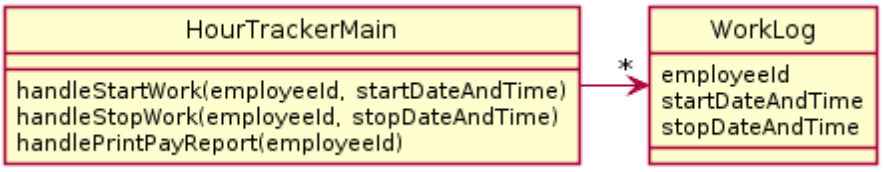
1) Describe the problems with Design A and B

2) Design your own solution

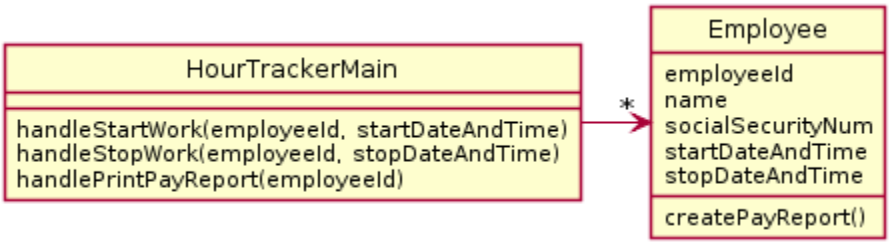
Hour Tracker

A system tracks employee hours at a particular company. Every time any employee starts work and stops work, the system must log it so the employee can be paid correctly and so management knows who was working when. The system must also print out a weekly pay report for each employee which includes total hours, the employee's name, social security number, and employee id.

Solution A



Solution B



Each of these designs also has a problem with functionality. One of them also violates another design principle that we discussed in class.

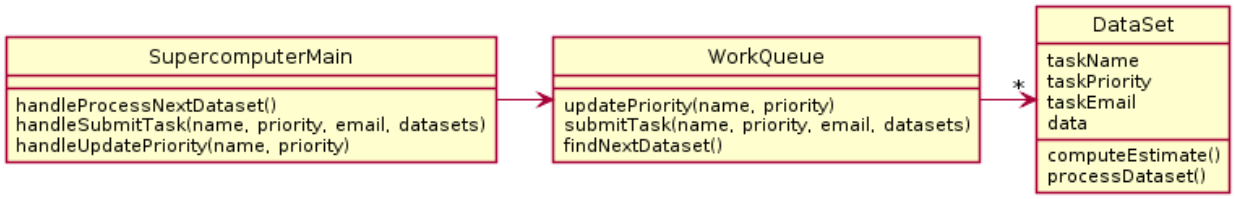
1) Describe the problems with Design A and B

2) Design your own solution

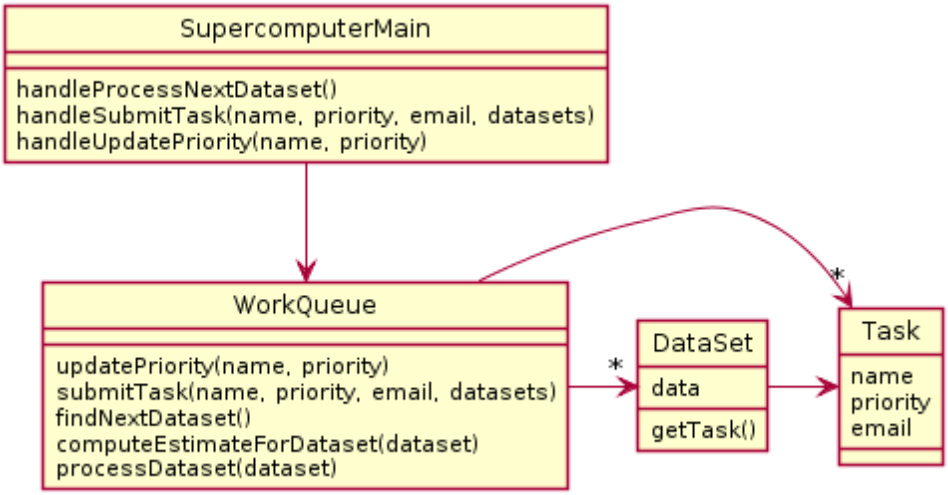
Supercomputer

The astronomy department maintains a supercomputer that everyone wants to use. Astronomers submit tasks to the supercomputer that consist of a series of datasets that must be processed independently. Each task includes a name, priority, and an email address where the results should be sent. Each dataset just has data. Given a dataset, it is possible to compute an estimate of how long it will take to run. The department agrees that the supercomputer should process datasets in priority order, and when priority is equal the supercomputer should select the dataset with the smallest estimate runtime. However, it must be possible to change a task's priority after it has been submitted.

Solution A



Solution B

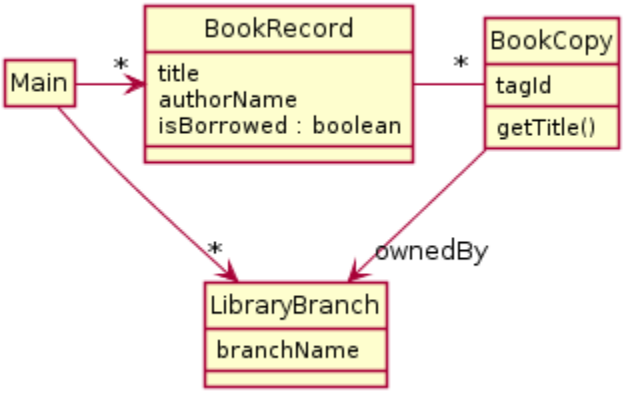


1) Describe the problems with Design A and B

2) Design your own solution

Libraries

In a specific city, there are multiple libraries all of whom use the same system to manage their book records. But it is often the case that the different branches will have copies of the same book. In these cases, the system manages two objects - a BookRecord which stores book data like title and author, and BookCopy which represents a physical book that's available for borrowing. Each book copy is tagged with a unique tag id number which is used to determine which branch the book ought to be shipped to when it is returned.



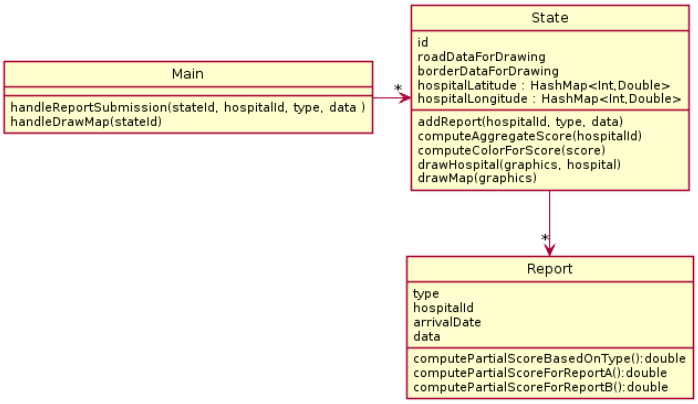
1) Describe the problems with Design A

2) Design your own solution

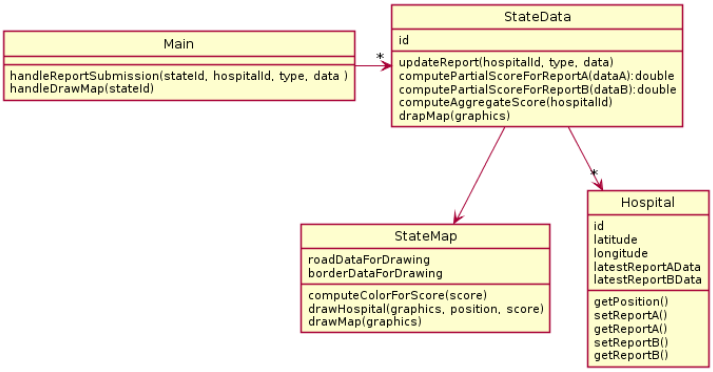
State Hospitals

A government agency tracks hospital data to make a monthly report. The highlight of the report is a set of colorful maps that show every hospital in a particular state with colors indicating the hospital's overall status. To make this report, hospitals submit two different kinds of reports (reportA, reportB) that come in at different times. Each of these reports is analyzed differently and produces a partial score. The final map color is produced by combining these three scores into an aggregate score, using the most up-to-date version of each report available at that time. Color on the map is positioned according the the hospital's specific latitude and longitude which is stored in the system and does not change. The drawn map also includes the states borders and major roads.

Solution A



Solution B



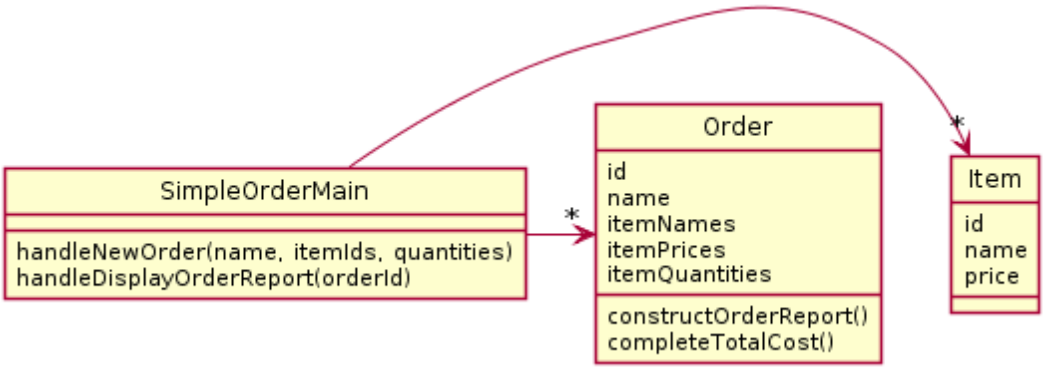
1) Describe the problems with Design A and B

2) Design your own solution

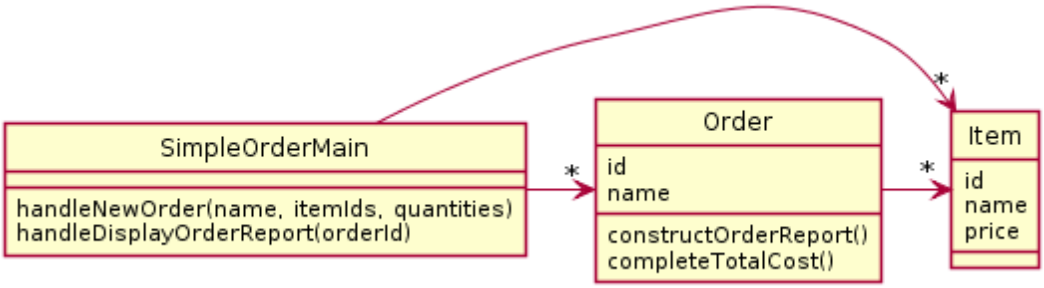
SimpleOrder

A particular company tracks orders for items. When an order is placed it consists of a customer name, a list of item ids, and a list of quantities. For example and order "Steve", [30012,30044], [1,2] would mean Steve has ordered one of item 30012 and 2 of item 30044. The ordering system keeps track of all the items that can be ordered - each item has id, name, and (per item) price. When an order is placed the system gives order id that can be used to identify the order later. Using that order id, the user can at any time display a report containing the names and quantities of all items ordered, the price of each item, and the total price.

Solution A



Solution B



1) Describe the problems with Design A and B

2) Design your own solution

Notflix Billing

There is a new company trying to enter the world of online video streaming. They have a different philosophy on how customers should be billed though. This new company, called *Notflix*, believes that customers should **not** have to pay the *full* monthly fee if they haven't watched many videos that month. It's not a rental service per se because there is a "cap" on the cost. The content is a lot like Netflix or Hulu, etc., but what you pay at the end of the month depends on how many videos you have streamed UP TO a certain number. If a customer watches more than a certain number X of videos (this number can fluctuate month-to-month due to promotions, customer appreciation, etc., and is provided externally and thus you do not need to worry about this value in the designs below), the customer is simply charged the usual flat monthly rate similar to the other companies; however, if a customer watches less than X in a particular month, the customer will pay less than the flat monthly rate -- The bill is calculated by multiplying the number of videos (that the customer has watched that month) by a flat fee for each video (regardless of video type, length and/or resolution, all videos have the same flat fee).

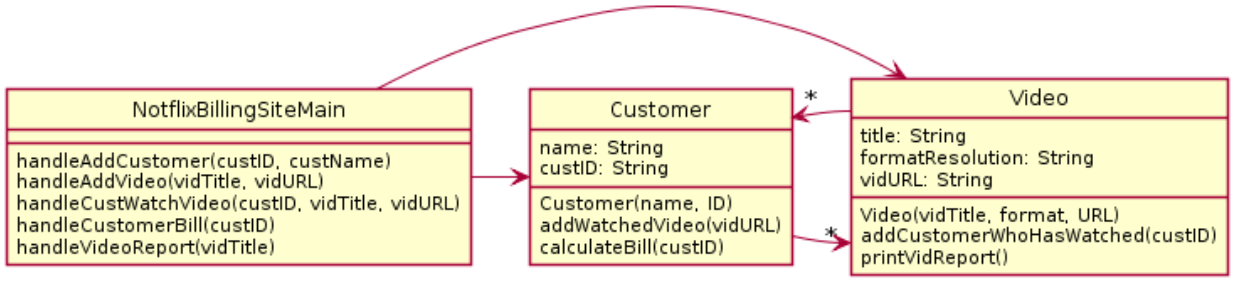
All videos have a title, a format/resolution and then a URL of each (title, format) pair of videos. One movie or episode can have multiple formats/resolutions, so the pair (title, format) uniquely identifies a video (note that all episodes have unique names from one another across all the different shows available); however, a video URL can also uniquely identify a specific title at a specific format/resolution. Note, a customer can watch a video in any format/resolution, but the price is constant. "You shouldn't have to pay for higher resolution video anymore," says Notflix.

For customers, Notflix needs to track their name and a unique ID. You can assume that a different part of the site (not shown below) handles the actual billing for a customer, i.e., a different part of the program tracks their billing address, credit card information, etc., and then handles the transactions. For this problem, you are only tracking the necessary information for *calculating* a customer's bill.

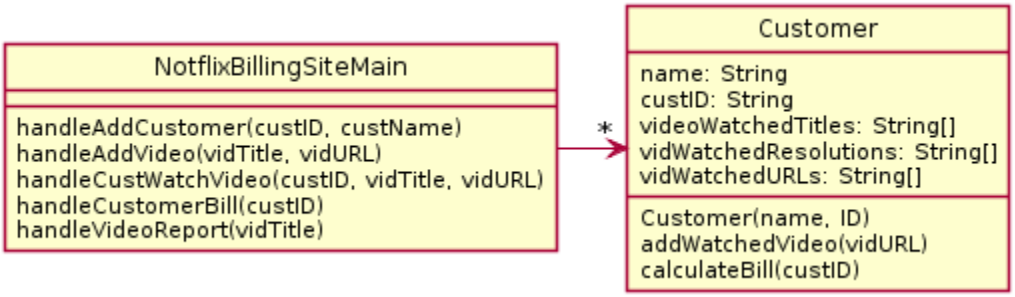
At the end of every month, Notflix billing needs to calculate a bill for each customer based on the number of videos that they watched (charging a flat fee if more than X videos were watched or, if less, calculating the charge from multiplying the flat fee for each video and the number watched by that customer). It is also important to be able to run reports on all the videos in the system to see how many customers have watched that video (in any format/resolution) so that they know which videos are more popular than others.

The following 2 solutions are both problematic. You will be asked to identify the specific OO Design Principle violated and then you will be asked to give a brief explanation of your answer.

Solution A



Solution B



a) Identify the number/letter combination of principle(s) violated (i.e. 1.b, 2.a) by SolutionA:

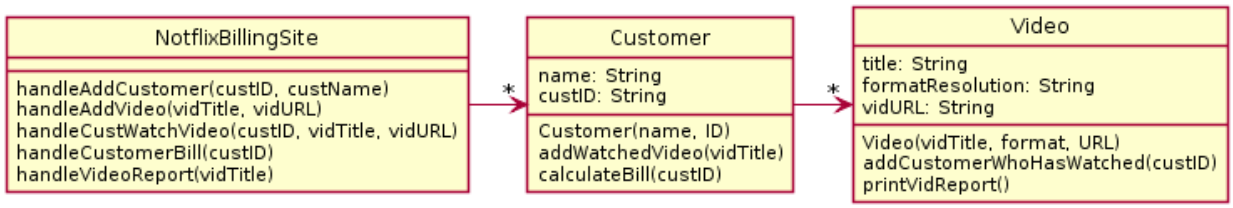
b) Explain why that/those principle(s) is/are violated (what is wrong with SolutionA?).

c) Identify the number/letter combination of principle(s) violated (i.e. 1.b, 2.a) by SolutionB:

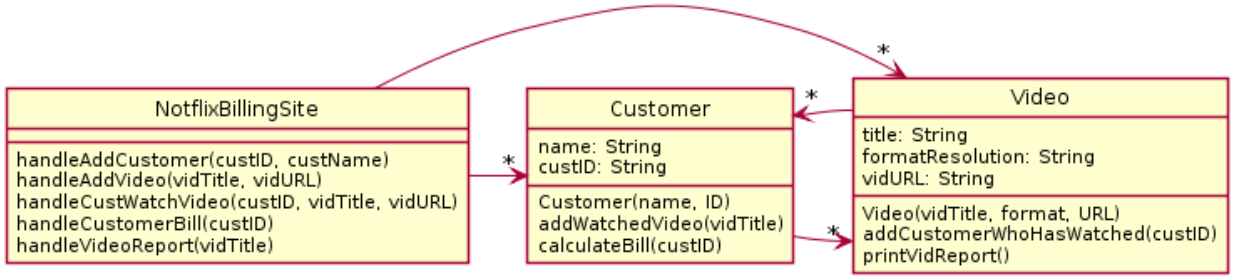
d) Explain why that/those principle(s) is/are violated (what is wrong with SolutionA?).

e) Below, you are given several designs that should "fix" the problem(s) with Solution A and/or Solution B. Select the option that represents the "best" design (i.e., a design that doesn't violate any of the OO Design Principles that we have discussed so far).

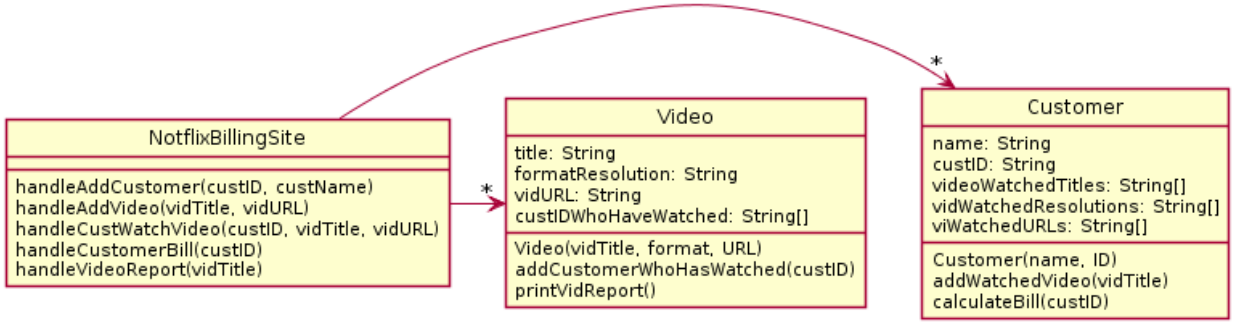
Design A



Design B



Design C



Design D