CSE 231 Problem Set 05

# Problem 33.1: Identify Level of Redundancy

Identify the level of redundancy in the following class diagram:



This class Diagram contains a level of redundancy of CRITICAL as Siblings in the lower classes have duplicate attributes. These duplicate attributes could be the Symbol, DecsSeoarator and the ThousanddsSeperator.

If the design is not distinct, suggest a design which is:

The best way to resolve this problem of CRITICAL redundancy would be to move the Symbol, descSeparator, and the ThousandsSepaarator to the Money Class so that theses attributes can instead be inherited and used that way.

# Problem 33.2: Identify Level of Redundancy

Identify the level of redundancy in the following class diagram:



The level of redundancy for this class is MINOR. There a small instances redundancy in this diagram, mainly in regards to -transactions, number, and price.

If the design is not distinct, suggest a design which is:

move the attributes of price, transaction, and number to the Account class, and provided the lower classes the option to used those attributes

# Problem 33.3: Identify Level of Redundancy

Identify the level of redundancy in the following class diagram:



The Level Of redundancy for this class diagram is CRITICAL, this is base on the fact that each of the lower classes of a method titled UPDATE(), which is common between them.

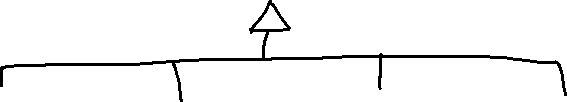
If the design is not distinct, suggest a design which is:

Making this class diagram Distinct is a relatively easy fix, the UPDATE() method which is contained and is common between the lower classes could be moved to the inherited class of GRAPH

# Problem 33.4: Design a Class Diagram

Design an inheritance hierarchy exhibiting distinct redundancy to satisfy the following problem definition:

A single transaction in a financial application corresponds to an event which influences the balance. There are many types of transactions. A deposit will put cash into a checking or saving account. A withdrawal will remove cash out of a checking or savings account. A purchase will add several shares at a given prices. A sell will remove several shares at a given price. All transactions have several dates: when the transaction was initiated, when it was completed, and when it was reconciled. Transactions also have a status: Pending, Cleared, Reconciled, Voided.



Transactions\_\_\_\_\_\_\_\_\_

#date: Date

#amount

#tranType:enum{ Pending, Cleared, Reconciled, Voided }

*+logDate()*

*+addShares(int)*

*+removeShares(int)*

*+transStatus (tranType)*

Deposit\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

-amount

-account:Acount

*+logDate()*

*+transStatus (tranType )*

*+addShares(int)*

+add(Account, int)

Withdrawal\_\_\_\_\_\_\_\_\_\_\_\_\_\_

-amount

-account:Acount

*+logDate()*

*+transStatus (tranType )*

*+addShares(int)*

+remove(Account, int)

Purchase\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

-amount

*+logDate()*

*+transStatus (tranType )*

*+addShares(int)*

Sell\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

-amount

*+logDate()*

*+transStatus (tranType )*

*+removeShares(int)*

# Problem 33.5: Design a Class Diagram

Design an inheritance hierarchy exhibiting distinct redundancy to satisfy the following problem definition:

There are several types of users on the system. There is the administrator, with unlimited access to all the system’s resources. There is an auditor, able to view everything on the system but make no changes. There are normal users who have complete access to their account but nothing else. In other words, there is a list of accounts to which they have access. Finally, there is a restricted user. This is like a normal user but has read-only access to a fixed set of accounts. Every user has a username, friendly name, and a password.



Standard\_\_

# password:Pwd

# username:Usn

# friendName: string

*+read()*

*+write()*

Administrator\_\_

# password:Pwd

# username:Usn

# friendName: string

*+read()*

*+write()*

Restricted\_\_

# password:Pwd

# username:Usn

# friendName: string

*+read()*

Auditor\_\_\_\_\_\_

# password:Pwd

# username:Usn

# friendName: string

*+Read()*

Secutity\_\_\_\_\_

# password:Pwd

# username:Usn

# friendName: string

*+accessLevel()*

Write\_\_\_\_\_\_\_\_\_\_

*+write())*

Read\_\_\_\_\_\_\_\_\_

*+read()*