CSE 231 Problem Set 03

# Problem 31.1: Level of Adaptability

Consider the following class diagram. The base class has no member variables and two pure virtual functions. The derived classes honor the base class’ interface but provide all their own functionality. The convert() method allows one currency type to convert to another.



Characterize the level of adaptability based on the following scenario. Provide a convincing rationale as to why it is what you say it is.

Augment the Money class to include an English pound.

The level of adaptability for the augmenting the Money class to include the English pound is a ENABLING level of adaptability. This is due to the fact that the user may only need to add in a member variable within the EURO class which will store and include a value for the English Pound. Another option would be to continue to store all values as cents, then add a function that will display any cents value as Pounds.

# Problem 31.2: Level of Adaptability

Consider the following class diagram. The initialize() function sets the sign (used in the display() method) and conversion value (used in the convert() method) in the base class.



Characterize the level of adaptability based on the following scenario:

Augment the Money class to include an English pound, which behaves much like the American dollar and the Euro.

The level of adaptability for the augmentation of include the English pound, which will behave as the American dollar would be PROHIBITIVE in regards to the current layout. the user will have to add in a way to not only display the value the selected American dollar format, but flowing and converting the stored currency in and between different formats could result with a loss of data.

# Problem 31.3: Level of Adaptability

Consider the following class diagram:



Characterize the level of adaptability based on the following scenario:

Create a class to represent an English pound

The level of adaptability for the addition of including the English pound would be considered STRIGHTFORWARD as a additional child class of English Pound would be add ensuring that it include similar functions like are contained within the other classes represented.

# Problem 31.4: Design a Class

Create a class diagram matching the following scenario:

A personal finance application can display several types of reports: a budget report, a histogram showing account balances over time, and a cash flow graph depicting income vs outgo.

|  |
| --- |
| Report |
|  |
| *+display()*  *+format()* |



|  |
| --- |
| Cash |
| - income: double  -outgo: double |
| *+display()*  *+format()*  +returnBudget():double |



|  |
| --- |
| Histogram |
| -time: Time  -balance: double |
| *+display()*  *+format()*  +returnBudget():double |

|  |
| --- |
| Budget |
| -balance: double |
| *+display()*  *+format()*  +returnBudget():double |

Characterize the level of adaptability for this or for your class diagram.

The level of adaptability is Straightforward, there is much that can be done by adding in additional child class with any specific features per class

After your initial characterization of adaptability, make the following change:

|  |
| --- |
| Report |
|  |
| *+display()*  *+format()* |



|  |
| --- |
| Cash |
| - income: double  -outgo: double |
| *+display()*  *+format()*  +returnBudget():double |



|  |
| --- |
| Histogram |
| -time: Time  -balance: double |
| *+display()*  *+format()*  +returnBudget():double |

|  |
| --- |
| Budget |
| -balance: double |
| *+display()*  *+format()*  +returnBudget():double |



|  |
| --- |
| PieChart |
| -time:Time |
| *+display()*  *+format()*  +overview() |



Create a new report type: a pie chart. This will display the percentage of spending associated with each budget category.

Describe in English what you would need to do to accommodate this change:

The Pie Chart will need information from each of the different class, meaning they will need information from all of them, but it can also inherit the display and format from the general Reports class.

After making this change, characterize again the level of adaptability. Was your predicted and realized levels of adaptability comparable?

Yes, I only needed to add an additional Child class that also required information from the other classes.

oblem 31.5: Design a Class

Create a class diagram matching the following scenario:

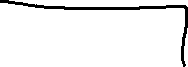
A personal finance system can have three types of users: an administrator who has complete access to the entire system, an auditor who has read-only access to the entire system, and a standard user who can only read/write those accounts specifically assigned to her.

|  |
| --- |
| Read |
|  |
| *+Read()* |

|  |
| --- |
| Security |
|  |
| *+access()* |

|  |
| --- |
| Write |
|  |
| *+Write()* |

|  |
| --- |
| auditor |
|  |
| *+readData()* |



|  |
| --- |
| standard |
|  |
| *+readData()*  *+WriteData()*  *+access()* |

|  |
| --- |
| administrator |
|  |
| *+readData()*  *+WriteData()* |

Characterize the level of adaptability for this or for your class diagram.

The level of adaptability is Straightforward, the user should only need to oldy add additional actions through adding or removing inheritance or changing and adding child classes

After your initial characterization of adaptability, make the following change:

Add a new type of user: restricted. This user can have read-only access to only her specifically assigned accounts and no other.

Describe in English what you would need to do to accommodate this change:

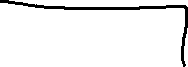
based off the current construction, I should only need to create a new class and inherit from the appropriate classes depended on the requirements for the restricted class

|  |
| --- |
| Read |
|  |
| *+Read()* |

|  |
| --- |
| Security |
|  |
| *+access()* |

|  |
| --- |
| Write |
|  |
| *+Write()* |

|  |
| --- |
| auditor |
|  |
| *+readData()* |



|  |
| --- |
| standard |
|  |
| *+readData()*  *+WriteData()*  *+access()* |

|  |
| --- |
| administrator |
|  |
| *+readData()*  *+WriteData()* |

|  |
| --- |
| restricted |
|  |
| *+readData()*  *+access()* |

After making this change, characterize again the level of adaptability. Was your predicted and realized levels of adaptability comparable?

Yes, all that was needed to do was to create a new child class called Restricted, and it inherited only that classes that it needed for its functionality.

