Domian Logic Patterns Practice

1 Background

Building on the chemical example on the last homework. The inheritance hierarchy has two changes:

- In Chemical, inhabits is replaced by a real number that is the number of moles currently in stock
- Metal has one additional field that is the number of moles of acid required to dissolve one mole of the metal.

In addition to the user interactions that assignment required, this assignment adds these (yes, you will need to have some data existing in your db since we are not including all of the CRUD operations):

- Add a new element (name, atomic number and atomic mass) with an option to specify that it is a metal (optional selection of an acid that dissolves it)
- Modify the amount of a chemical in our inventory
- Modify the atomic number of an element
- Modify or add which acid dissolved an existing metal
- Add, modify and delete operations for the relationship between a compound and the list of elements that it is made of
- Generate a report which chemicals are currently low in our inventory (asks for a text file where the output should go.)

The system must enforce these rules:

- There can be no duplicates in the list of elements a compound is made of
- The atomic number of an element cannot be more than its atomic mass
- The name of an element must be a single word, but the name of a compound, base, or acid can contain multiple words
- A chemical is considered low in our inventory as follows:
  - We are low on an element if we could not completely replace our current inventory of compounds
  - We are low on an acid if we could not completely dissolve the metals in our inventory

- We are low on an element if we have less than 20 moles
- We are low on a base if we have less than 40 moles

You can assume that the presentation layer only has to know if an operation was successful or failed. In other words, there is no existing display that has to be updated with new information.

2 Transaction Scripts

Suppose we had one script for each of the required operations.

1. List the scripts
2. Identify any functionality that is shared between the scripts
3. Suppose you were working in an object-oriented language. Is there a way to rationally group the scripts? If so, how would you group them. If not, give an example that shows why not.

3 Domain Model

4. Assuming we are going to use the Command Pattern, list the command classes and their attributes.
5. Update the inheritance hierarchy to include the methods that each class will require

4 Table Module

This example doesn't have much need to do things a table at a time, so, unless you were working in a framework that prescribed a table module, you probably wouldn't use one. Instead, answer these questions about the Table Module pattern.

6. Relating to the revenue recognition example for this pattern:
   (a) Who would create the instance of Contract?
   (b) Explain the method declare with "public DataRow this [long key]" on p 130 by answering these questions
      i. What is the "this"?
      ii. Why does it have square brackets instead of parentheses?
   (c) Explain what allocate is doing including the magic it is doing with remainders
7. Why does table module not give you the ability to do OO things like:
   (a) polymorphism
   (b) instance-to-instance relationships