Java Lab 12: Memo Mania Fall 2022

This lab will re-use your solution to the Lab 11 project. Create a new project, Lab12, and copy your code into it, using the same package structure. When the problem says to change or replace some existing code, you should comment out the old line and add the new line, just in case you have to revert.

1. Add a static factory method in NoteCollection: **Note createNote(type, name, body, from, to)**, where type is one of the concrete note types. Change every constructor in the Note hierarchy to package private. Use createNote( ) in main( ) in the places you previously new'd up notes. Make sure it all works.

2. In **Note**, add a getter for the **body** field and add **abstract** getters for the **to** and **from** fields (this is not a great solution, but go with it). Then add regular getters from the **to** and **from** fields in **Memo** and a getter for today in TimedMemo. Create a class called **NoteDisplayer** that contains the following static methods:

void displayNote(Note note) – simply prints note's toString.

void displayNoteFancy(Note note) – display the Note as shown in the figures using the getters above instead of toString. Think about how to handle the today field for TimedMemo and PoliteTimedMemo.

```
Enter your choice: 1
************************
* Number: 1            *
* Name  : Memo         *
* From  : Barrett      *
* To    : Jones        *
************************
*
* Body:
This is a regular memo
```

```
************************
* Number: 2            *
* Name  : Timed        *
* From  : Barrett      *
* To    : Smith        *
************************
* Date: 2020-10-12
*
* Body:
This is a TimedMemo
```

```
************************
* Number: 3            *
* Name  : Polite       *
* From  : Barrett      *
* To    : Adams        *
************************
* Date: null
*
* Body:
This is a PoliteTimedMemo
```

void displayErrorMessage(String errorMessage) – display an error message.

In main, find the places that notes are displayed and replace them by displayNote, except for the "display all notes" option – use displayFancyNote for that. Find all the error message println's and replace them with displayErrorMessage.

What pattern is being used, and what are its parts?

3. This problem is practice with the Builder method, but just for the **PoliteTimedMemo** class. Following the pattern outlined in the notes:

- create a **public static Builder** class *inside* the **PoliteTimedMemo** class;

- since all four parts will be required, Builder needs only a default constructor;

- create four methods in Builder, one each for the four fields name, body, from, to; the return type for each should be Builder, and each should return this;

- create a **build( )** method in Builder that returns a new PoliteTimedMemo with this as the parameter using the next method;

- add a private PoliteTimedMemo constructor taking a Builder as a parameter; it should set this's fields base on the parameters fields. In addition, increment noteCount and set noteNumber to it.

Now test the Builder: in NoteCollection's factory method, replace the call to new PoliteTimedMemo to one using the PoliteTimedMemo.Builder, calling the four methods (in any order – you should test it with different orderings to check that it works) and build( ).

Was it worth it? Seems like a lot of trouble!