Coins Project

Sprint 3

Sprint Goal

The goal of the sprint is to refactor the metallurgy to use the Strategy design pattern. And, as always, our code meets our corporate requirements: checkstyle, unit testing.

Sprint tasks

1. Create Metallurgy hierarchy (interface and concrete classes).
2. Test the concrete Metallurgy implementations of the interface. (The interface does not need a test because it has no executable code.)
3. Refactor the Coin class to have a Metallurgy delegate and add the smelt() method that uses the delegate. Refactor the Coin constructor to set the delegate with the concrete Metallurgy object passed by the coin subclass and then call its own smelt method to set the metallurgy String field.
4. Refactor the concrete coin classes (e.g., Penny) to no longer pass a metallurgy string but to instead pass a Metallurgy object.
5. Write a Demo class that adds to its demonstration of our code. We should be able to create a Penny with a different metallurgy now.

Domain details

The Treasury Department undersecretary is very happy with your progress so far! They are now thinking that one area of concern for them is coin metallurgy. They want to explore various metallurgy options to increase coin longevity and durability while possibly reducing manufacturing costs.

United States coins specifications are indicated in table below.

Familiar name	value	frontMotto	backMotto	frontLabel	backLabel	frontImag
Penny	0.01	"IN GOD WE TRUST"	"E PLURIBUS UNUM"	"LIBERTY"	"UNITED STATES OF AMERICA"	"A_Lincoln"

Nickel	0.05	"IN GOD WE TRUST"	"E PLURIBUS UNUM"	"LIBERTY"	"UNITED STATES OF AMERICA"	"T_Jefferso
Dime	0.10	"IN GOD WE TRUST"	"E PLURIBUS UNUM"	"LIBERTY"	"UNITED STATES OF AMERICA"	"F_Roosevel
Quarter	0.25	"IN GOD WE TRUST"	"E PLURIBUS UNUM"	"LIBERTY"	"UNITED STATES OF AMERICA"	"G_Washing
HalfDollar	0.50	"IN GOD WE TRUST"	"E PLURIBUS UNUM"	"LIBERTY"	"UNITED STATES OF AMERICA"	"J_Kennedy'
Dollar	1.00	"IN GOD WE TRUST"	"E PLURIBUS UNUM"	"LIBERTY"	"UNITED STATES OF AMERICA"	"S_Anthony

Implementation details

Right now, our various concrete coin classes (e.g., Quarter) are passing a constant string that describes the metallurgy of its coin. We need this to be able to change. So we will employ the Strategy Design Pattern.

We will move the current metallurgy (the constant strings) into their own encapsulations. All of these various metallurgy encapsulations are related to each other by a common interface that we will call Metallurgy. This interface will require a String smelt() method that returns the string constant.

The Coin abstract class currently has a String metallurgy field. We will add a new composition with a Metallurgy delegate named smelter. The Coin class will have a new method `void smelt()` that it will call to set the metallurgy field. This smelt() method will relay the request to the smelter Metallurgy delegate.