Coins Project

Sprint 2

Sprint Goal

The goal of the sprint is to address a change request. Our client wishes to have separate and distinguishable coin objects for the various coin types (e.g., quarter, dime). And, as always, our code meets our corporate requirements: checkstyle, unit testing.

Sprint tasks

1. Refactor Coin as an abstract class
2. Create two-level coin hierarchy with Coin at top and Dollar, HalfDollar, Quarter, Dime, Nickel, Penny classes extending Coin. These classes should pass checkstyle and pass tests.
3. Update CoinTest to support changes in Coin class.
4. Write JUnit test classes for the concrete coin classes (e.g., Dime, Penny).
5. Write a Demo class that demonstrates our code (Coin class) satisfies customer requirements.

Domain details

Client wants separate and distinguishable coin objects instead of a singular object that has varying field values. In essence, the software should reflect, or model, the physical reality. The physical reality is that quarters are different kinds of objects from pennies even though they have some commonalities.

Client still wants to be able to create multiple coin objects of various value denominations. Most attributes are "built in" to the coin denomination; for example, images and mottos for coins are fixed. Consequently, constructor doesn't need many parameters and there should not be any way to change these attributes once they are set. (Coins don't change after construction!) **NEW DESIGN:** The Quarter constructor knows all the attribute values and needs a superclass constructor that accepts all these.

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

Our test classes and demo classes do not have to meet checkstyle or testing requirements. Only "production code" classes must meet those requirements.

We will separate the "main" code from the "test" code from the "demo" code with subfolders inside the src/ folder. For example, we will have `src/main/` and `src/demo/` and `src/test/` folders.

See the docs/Coin-V1-Class.png image for a class diagram depicting the system.

Each production class should have a corresponding test class. Each production class method should have at least one corresponding test class test method. Exceptions are the common "groups" of methods (constructors, getters, setters).

The Quarter test class needs a method to test it's constructor and that's all. It's up to the Coin test class to test its constructors and getter methods. How can a test class for an abstract class create an instance to test against? It must instead create a "mock" subclass to instantiate. So something like:

```
public class CoinTest {
   private Coin testCoin = new MockCoin();
   ...
   @Test
   void public testGetters() {
      assertEquals(24, testCoin.getValue());
      ...
    }
}
class MockCoin extends Coin {
   public MockCoin() {
      super(24, "Mock", "frontMotto", 1788, "frontImage", "backImage",
"backMotto", "frontLabel", "backLabel", "twenty-four cents", false,
"metallurgy");
   }
}
```