Data Structures Programming Assignment 1: OOP Review

Last month, scientists at NASA discovered life on Mars, Neptune, and Saturn! Your job is to create a program that can convert currencies between the planets to aid in interplanetary transactions.

Implementation details

You will create 3 currency classes: **Mars.java**, **Neptune.java**, and **Saturn.java**. Each of these classes must derive from an abstract parent class **Currency.java** and implement the interface **Exchangeable.java**.

Currency.java

The currency class must contain the following data members:

- name of the currency (`String currencyName`)
  - MarsMoney
  - NeptuneNuggets
  - SaturnSilver
- total funds (`double totalFunds`)

Additionally, because the planets only trust Earth, dollars will serve as the intermediary currency.
The Currency class will contain the following 2 abstract methods:

```
    public abstract double toEarthDollars(double amount);
    public abstract double fromEarthDollars(double EarthDollars);
```

Use these methods to exchange money between planets.

Exchangeable.java

Exchange rates should be implemented as constants in the Exchangeable interface.
1.00 EarthDollar (ED) = 1.30 MarsMoney (MM) = 0.87 SaturnSilver (SS) = 2.00 Neptune Nuggets (NN)

Note that we may change the values of the exchange rates when testing your code. We will only edit the Exchangeable interface, so rates should be encapsulated in and accessed from Exchangeable.java.

Exchangeable must also include a method to exchange between the current currency and a target currency. When a planet calls `exchange()` with another planet and a specified amount, the methods `toEarthDollars()` and `fromEarthDollars()` should be used to convert the source currency into the target currency. The amount should be subtracted from the calling planet and added to the target planet. If a planet tries to exchange more funds than it currently has, an error should be printed and no transfer should occur.

Each planet has a different exchange fee for a transaction, which you must implement. Mars charges 10% of the total transaction amount, Neptune charges a flat rate of 5 NeptuneNuggets per transaction, and Saturn is free. The exchange fee should also be subtracted from the calling planet, but not added to the target planet.

The exchange method should have the following signature:

```
    public void exchange(Currency other, double amount);
```

The remainder of the implementation details are up to you. As always, use best practices, including the principle of least privilege, inheritance, and encapsulation as much as possible.

Submission details

Your code should also include a main method (you can put it in Currency.java or create a separate class) to test your functionality.

Please zip all source files and submit on Brightspace.

Sample run

Input

```
Currency mars = new Mars(100.00);
Currency neptune = new Neptune(100.00);
Currency saturn = new Saturn(100.00);

System.out.println("<-- Exchanges -->");

mars.exchange(saturn, 25.00);
neptune.exchange(saturn, 10.00);
saturn.exchange(mars, 122.00);
saturn.exchange(mars, 121.00);
```

Output

```
<-- Exchanges -->
Converting from MarsMoney to SaturnSilver and initiating transfer…
25.00 MarsMoney = 19.23 EarthDollars = 16.73 SaturnSilver
Mars exchange fee is 2.50 MarsMoney
Mars has a total of 72.50 MarsMoney
Saturn has a total of 116.73 SaturnSilver

Converting from NeptuneNuggets to SaturnSilver and initiating transfer…
10.00 NeptuneNuggets = 5.00 EarthDollars = 4.35 SaturnSilver
Neptune exchange fee is 5.00 NeptuneNuggets
Neptune has a total of 85.00 NeptuneNuggets
Saturn has a total of 121.08 SaturnSilver

Uh oh - Saturn only has an available balance of 121.08, which is less than
122.00!

Converting from SaturnSilver to MarsMoney and initiating transfer…
121.00 SaturnSilver = 139.08 EarthDollars = 180.80 MarsMoney
Saturn exchange fee is 0.00 SaturnSilver
Saturn has a total of 0.08 SaturnSilver
Mars has a total of 253.30 MarsMoney
```