Ocena zadania - Bohdan Khorvat

Polecenie

Stwórz implementację klasy Money. Klasa powinna przechowywać informację na temat ilości i waluty. Klasa powinna mieć funkcje, które pozwolą na: dodawania, odejmowanie, mnożenie i dzielenie. Najlepiej jeżeli klasa powstanie zgodnie ze wzorcem projektowym ValueObject. Zwróć uwagę na to, żeby zostały obsłużone potencjalne błędy, które mogą pojawić się podczas wykorzystania tego obiektu w praktyce. Napisz testy, które potwierdzą poprawne działanie implementacji.

Ocena

Na plus

1. Dzięki użyciu bcmath klasa nie jest podatna na błędy zaokrągleń, które zdarzają się podczas operacji na liczbach zmiennoprzecinkowych.
2. Dodawanie, odejmowanie, dzielenie i mnożenie działa i zwraca poprawne wyniki.
3. Kod jest w większości zgodny z PSR-12. Wyjątkiem są np. wcięcia.
4. Informacja o wersji PHP kompatybilnej z projektem w composer.json.
5. Użycie PHP >=8.
6. Metody mają zdefiniowane typy konsumowanych parametrów i zwracanych danych.
7. Repozytorium nie zawiera plików i katalogów, których nie powinno, jak np. vendor lub .idea.
8. Testy zostały zaimplementowane przy użyciu PHPUnit.

Na minus

1. Poniższy kod zwraca wyjątek, którego komunikat jest nieprawdą - “Value: 0.01000000000000is not positive number”.
```
$money = new \App\Money('35', 'USD');
$moneyPln = new \App\Money('34.99', 'usd');
$res = $money->subtract($moneyPln);
```
2. Wzorzec Value Object nie jest zaimplementowany poprawnie. Nie można porównać dwóch obiektów klasy Money bez wyciągania ich wewnętrznych wartości. Klasa powinna mieć metodę, która pozwalałaby na porównanie np takie: **$equals = $money1->equals($money2)**;
3. Brak deklaracji silnych typów declare(strict_types=1), co nieco zwiększa ryzyko powstania subtelnych błędów programistycznych.
4. Jeżeli wynik operacji subtract jest równy 0, metoda subtract zwraca int 0, zamiast obiektu klasy Money. Warto dbać o spójność typów zwracanych przez metodę. Użycie kodu **$amount = $money->subtract($moneyToSubtract)->getAmount();** może spowodować **Call to a member function getAmount() on int.**

5. W niektórych miejscach można było użyć bardziej ekspresywnego kodu zamiast komentarzy. Np. zamiast okomentowanego private const SCALE = 14 zmienić nazwę stałej na BC_MATH_SCALE.
6. Metoda `**checkValueForPositiveNumber**` wyrzuca Exception, gdy sprawdzana przez nią wartość jest ujemna. Nazwa metody nie do końca oddaje jej funkcjonalność. Dla czytelności można byłoby użyć np. throwExceptionIfNumberIsNegative.