Тестовое задание для соискателя на должность PHP-разработчик

Задание «Мультивалютный банковский счет»

Разработать класс (пакет классов), реализующих функционал банковского мультивалютного счета.

Требования к реализации:

1. Мультивалютный счет обслуживает сбережения в нескольких валютах, которые можно подключать и отключать от счета динамически. Пример валют: российский рубль (`RUB`), доллар США (`USD`), евро (`EUR`).
2. Одна из валют счета является основной (для разных счетов может быть разной).
3. Мультивалютный счет поддерживает операции пополнения/списания в конкретной валюте.
4. При попытке списания в конкретной валюте, нельзя списать больше количества этой валюты на счете. Например, на счете `1000 RUB` и `10 USD`, при попытке списать `11 USD`, нельзя списать `10 USD` из долларовой части счета, а оставшуюся с рублевой.
5. Курс валют со временем может изменяться. Базовый курс валют: `EUR/RUB = 80`, `USD/RUB = 70`, `EUR/USD = 1`
6. Клиент должен иметь возможность изменить основную валюту счета.

Требования к коду:

1. Использование ООП, и дробление на функциональные классы.
2. Использование типизации (можно использовать все возможности PHP 8.1).
3. Основная бизнес-логика должна быть покрыта тестами.
4. Соответствие кода PHP Standards Recommendations.

Тестовый сценарий:

1. Клиент открывает мультивалютный счет, включающий сбережения в 3-х валютах с основной валютой российский рубль, и пополняет его следующими суммами: `1000 RUB`, `50 EUR`, `40 USD`.

```
Счет = Банк->ОткрытьНовыйСчет()
Счет->ДобавитьВалюту(RUB)
Счет->ДобавитьВалюту(EUR)
Счет->ДобавитьВалюту(USD)
Счет->УстановитьОсновнуюВалюту(RUB)
Счет->СписокПоддеживаемыхВалют() // [RUB, EUR, USD]
Счет->ПополнитьБаланс(RUB(1000))
Счет->ПополнитьБаланс(EUR(50))
Счет->ПополнитьБаланс(USD(50))
```

2. Клиент хочет увидеть суммарный баланс счета в основной валюте, либо в валюте на выбор.

```
Счет->ПолучитьБаланс() => xxxxx RUB
Счет->ПолучитьБаланс(USD) => xxxxx USD
Счет->ПолучитьБаланс(EUR) => xxxxx EUR
```

3. Клиент совершает операции пополнения/списания со счета.

```
Счет->ПополнитьБаланс(RUB(1000))
Счет->ПополнитьБаланс(EUR(50))
Счет->СписатьСБаланса(USD(10))
```

4. Банк меняет курс валюты для `EUR` и `USD` по отношению к рублю на `150` и `100` соответственно

```
EUR->УстановитьКурсОбменаВалюты(RUR, 150)
USD->УстановитьКурсОбменаВалюты(RUR, 100)
```

5. Клиент хочет увидеть суммарный баланс счета в рублях, после изменения курса валют.

```
Счет->ПолучитьБаланс() => xxxxx RUB
```

6. После этого клиент решает изменить основную валюту счета на EUR, и запрашивает текущий баланс

```
Счет->УстановитьОсновнуюВалюту(EUR)
Счет->ПолучитьБаланс() => xxx EUR
```

7. Чтобы избежать дальнего ослабления рубля клиент решает сконвертировать рублевую часть счета в EUR, и запрашивает баланс

```
ДенежныеСредства = Счет->СписатьСБаланса(RUB(1000))
Счет->ПополнитьБаланс(EUR(ДенежныеСредства))
Счет->ПолучитьБаланс() => xxx EUR
```

8. Банк меняет курс валюты для EUR к RUB на 120

```
EUR->УстановитьКурсОбменаВалюты(RUR, 120)
```

9. После изменения курса клиент проверяет, что баланс его счета не изменился

```
Счет->ПолучитьБаланс() => xxx EUR
```

10. Банк решает, что не может больше поддерживать обслуживание следующих валют `EUR` и `USD`. Согласовывает с клиентом изменение основной валюты счета на `RUB`, с конвертацией балансов неподдерживаемых валют.

```
Счет->УстановитьОсновнуюВалюту(RUB)
Счет->ОтключитьВалюту(EUR)
Счет->ОтключитьВалюту(USD)
Счет->СписокПоддеживаемыхВалют() // [RUB]
Счет->ПолучитьБаланс() => xxxxx RUB
```

Приемка

Для приемки выложить код в одном из публичном git-репозитории (GitHub, GitLab) и предоставить к нему доступ. В процессе выполнения задания можно задавать уточняющие вопросы.
Наличие покрытия тестами будет плюсом.

Оценка результата

При оценке будет учитываться как сам код, так и структура полученного проекта, использование типизации, процент покрытия кода, история коммитов.