Konta bankowe

Wprowadzenie

Jak już wiemy, **klasy wewnętrzne** w Javie przydają się na przykład wtedy, gdy chcemy by niektóre klasy były dostępne tylko w obrębie jednej, więszkej. Najczęściej mówimy o 3 typach klas wewnętrznych: statyczne/dynamiczne, lokalne oraz anonimowe.

Treść zadania

Zależy nam na symulowaniu przyrostu gotówki na naszych kontach bankowych. Każde konto bankowe ma swój aktualny balans oraz w jakis sposób korzysta ze swojego "zliczacza" implementującego `ActionListener`, nazywajmy go `BalanceCounter`. Jego zadaniem jest co sekundę zwiększać stan konta o zadane oprocentowanie.
Aplikacja ma wyświetlić proste okno utworzone w Swingu z guzikiem umożliwiającym jej zatrzymanie. Aby umożliwić równoległe naliczanie odsetek, w aplikacji ma chodzić `Timer`, który co sekundę odpali swojego listenera `BalanceCounter`.
W związku z tym stwórz 3 klasy:

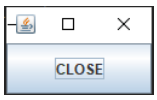
- `BankAccount1`, dla której `BalanceCounter` ma być **dynamiczną klasą wewnętrzną**, jej instancję stwórz własnoręcznie w mainie.
- `BankAccount2`, dla której `BalanceCounter` jest instancją **klasy lokalnej** wewnątrz metody `createBalanceCounter(double interestRate)`. Jej instancję otrzymaj w mainie przez użycie właśnie tej metody.
- `BankAccount3`, dla której `BalanceCounter` jest instancją **klasy anonimowej**, zwracaną przez metodę `createBalanceCounter()`. Jej instancję otrzymaj w mainie przez użycie właśnie tej metody.

Oczekiwany output dla początkowej wartości 1000 i oprocentowania 0.2 (liczenie aż do zatrzymania):

```
Changing account balance from 1000 to: 1200
Changing account balance from 1200 to: 1440
Changing account balance from 1440 to: 1728
Changing account balance from 1728 to: 2073
Changing account balance from 2073 to: 2487
Changing account balance from 2487 to: 2984
Changing account balance from 2984 to: 3580
```

Uwagi

1. Należy zadbać o podstawowe zasady programowania obiektowego - enkapsulację, gettery, settery, odpowiednią dostępność zmiennych... Te rzeczy nie są uwzględnione w treści zadania, ale są to dobre praktyki, które należy stosować.
2. Prezentację każdego z trzech zadań należy przeprowadzić oddzielnie. Podczas prezentacji można przyjąć, że każde konto zaczyna z balansem 1000, a każdy `BalanceCounter` będzie naliczał oprocentowanie 0.2. Oczywiście każda z tych wartości powinna być modyfikowalna.
3. Okno tworzone w Swingu powinno wyglądać podobnie do zaprezentowanego poniżej.



4. Wskazówka: jedynie `BankAccount3` przechowuje oprocentowanie jako atrybut, w `BankAccount2` oprocentowanie jest parametrem metody, a w przypadku pierwszym, oprocentowanie ma być trzymane jako atrybut klasy wewnętrznej.