Inżynieria oprogramowania

Refaktoryzacja do wzorców projektowych

1. Proszę otworzyć w NetBeans projekt:
   https://github.com/mjochab/kredyty.git

2. Czy konstruktory w klasie Kredyty są poprawnie wywołane?

3. Jak odróżnić od siebie konstruktory?

4. Jednym z rozwiązań może być refaktoryzajca poprzez wprowadzenie metod tworzących egzemplarze (ang. CreationMethods):
   a. wprowadzając odpowiednie metody tworzące zyskamy możliwość ich dowolnego nazwania (w przypadku konstruktorów w Javie nie jest to możliwe),

   b. nowe metody najwygodniej będzie użyć, jeżeli będą publiczne i statyczne,

   c. w celu dokonania refaktoryzacji proszę:
   - w metodzie main() jako wzór nowej metody wybrać linię tworzącą obiekt `kredycik` i uruchomić z menu kontekstowego `Refactor/Introduce/Method`,
   - nową metodę można nazwać `createKredyt`,
   - za pomocą `Refactor/Move` należy przenieść metodę do klasy zawierającej konstruktory,
   - sprawdzić czy program i testy się nadal uruchamiają,
   - klikając w nazwę pierwszego konstruktora i wybierając `Find usages` proszę znaleźć miejsca, w których należy podmienić jego wywołanie nową metodą (powinny to być 3 miejsca),
   - skoro konstruktor nie będzie już używany z zewnątrz zmienić jego atrybut na `private`,
   - powtórnie sprawdzić czy program i testy nadal się uruchamiają,

- analogicznie stworzyć nowe metody tworzące i zastąpić nimi pozostałe konstruktory - proszę pamiętać o wybraniu odpowiednich nazw dla metod tak, aby łatwo było później wybrać odpowiednią,

5. Dalszym etapem porządkowania kodu mogłoby być skorzystanie z polimorfizmu oraz fabryki:

   a. Klasę `Kredyt` należałoby podzielić na dwie:
   - `FabrykaKredytów` - zawierającą napisane wcześniej metody kreacyjne oraz statyczną i prywatną metodę, która będzie wywoływana zamiast dawnego konstruktora przez te metody (w niej należy wywoływać odpowiednie konstruktory podklas).
   - abstrakcyjną klasę `Kredyt` zawierającą pola dawnej klasy, konstruktor pozbawiony parametru typ oraz metodę abstrakcyjną `liczOprocentowanie`,

   b. dla każdego z typu kredytów należałoby stworzyć podklasę klasy `Kredyt` (np. `KredytChwilowka`) zawierającą:
   - konstruktor ustalający typ oraz wywołujący konstruktor klasy bazowej,
   - właściwą implementację metody `liczOprocentowanie`,

   c. w testach oraz w `main` należy zmienić wywołania `Kredyt` na `FabrykaKredytow`,