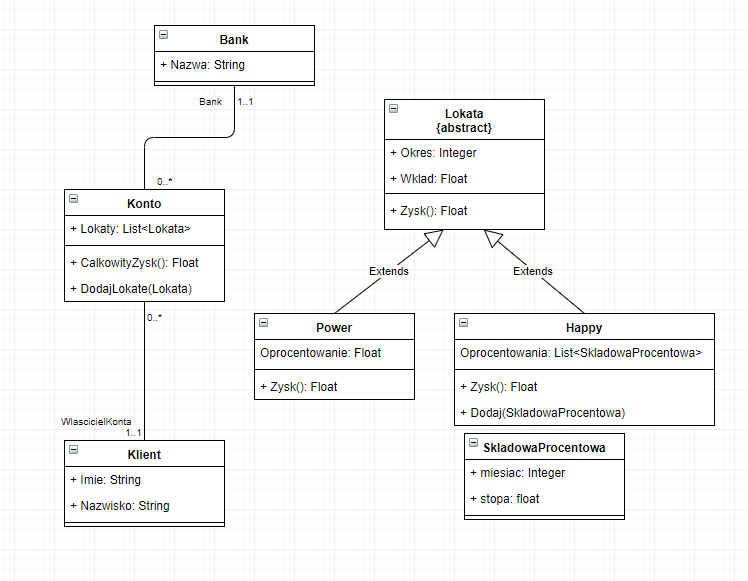
# Analiza problemu

Obiektowa baza danych rozwiązuje problem dotyczący zakładania lokat przez klienta oddziału bankowego. Klient posiada pieniądze (wkład własny), które chce ulokować na lokatach w banku. Struktura obiektowej bazy danych odpowiada strukturze bankowości i symuluje przepływ pieniędzy klienta przez poszczególne okresy kapitalizacji. Klient korzystając z bazy może sprawdzić co stanie się z jego pieniędzmi zanim faktycznie zainwestuje. Baza przechowuje dane o kliencie, wkładzie własnym, lokatach, które są możliwe do wyboru oraz umożliwia wyliczanie zysku z poszczególnych lokat zgodnie z podanymi parametrami.

# Model danych

Model danych został zaprojektowany przy użyciu aplikacji internetowej **www.draw.io:**



## Klasy

Model zawiera 7 klas:

1. Bank – klasa dla banku zawiera dane banku, w tym przypadku jego nazwę.
2. Konto – klasa dla konta klienta. Zawiera listę lokat znajdujących się na koncie.
3. Klient – klasa dla klienta. Zawiera dane klienta takie jak: imię i nazwisko.
4. Lokata – klasa abstrakcyjna po której dziedziczą klasy Power i Happy. Zawiera dane lokaty takie jak okres i wkład klienta.
5. Power – klasa lokaty o stałym oprocentowaniu. Zawiera parametr Oprocentowanie i metodą Zysk obliczającą zysk na podstawie stałego oprocentowania.
6. Happy – klasa lokaty o zmiennym oprocentowaniu. Zysk liczony jest na podstawie wcześniej ustalonych składowych.
7. SkladowaProcentowa – składowa służąca do wyliczania zysku w lokacie Happy.

## Metody

1. CalkowityZysk() – metoda liczy całkowity potencjalny zysk klienta na podstawie lokat znajdujących się na koncie.
2. DodajLokate() – dodaję nową lokatę do konta.
3. Zysk() – Metoda obliczająca zysk z lokaty.
4. Dodaj() – Metoda dodająca nową składową procentową do lokaty Happy.

## Dziedziczenie

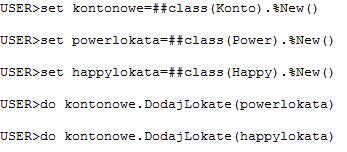
Klasy Power i Happy dziedziczą po klasie Lokata, która jest abstrakcyjna.

## Związki

Występują pomiędzy Bankiem a Kontem oraz pomiędzy Klientem a Kontem. Jedno konto może mieć jeden Bank, a Bank może mieć wiele kont. Podobnie jest w przypadku związku klienta z kontem.

## Klasy abstrakcyjne i polimorfizm

Klasą abstrakcyjną jest klasa Lokata, która może być albo lokatą Power albo lokatą Happy. Polimorfizm zachodzi w metodzie klasy Lokata Zysk (). Oba rodzaje lokat posiadają inną implementację tych metod co odróżnia je od siebie względem potencjalnych zysków klienta.



## Typy danych

Nowo zdefiniowanym typem danych jest klasa SkladowaProcentowa, która odpowiada za parametry do wyliczenia zysku w okresie kapitalizacji.

# Środowisko Cache

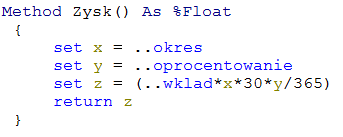
Projekt znajduje się w repozytorium pod adresem: <https://github.com/inimisaru/OBD-Wat-2018>

## Wypełnienie bazy danymi testowymi

Wypełnianie bazy danymi testowymi odbywa się przy użyciu skryptu znajdującego się w repozytorium.

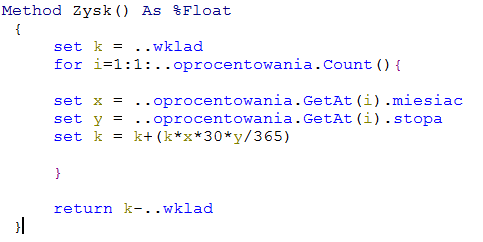
## Wykonanie metod

**Power.Zysk()**



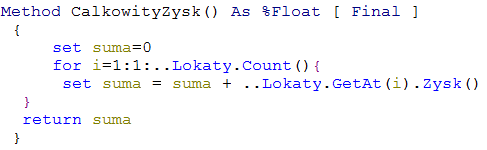


**Happy.Zysk()**





**Konto.CalkowityZysk()**





# Podsumowanie

## Napotkane problemy i ich rozwiązania

Z pewnością dużo problemów sprawia samo środowisko, pomimo zgodności kodu z dokumentacją występują błędy w kompilacji lub podczas zarządzania bazą z poziomu terminalu. Może to być wina maszyn i zainstalowanego na nich oprogramowania, być może problem by zniknął, gdyby zainstalować środowisko Cache na własnych maszynach.

## Pomysły/idee

Niektóre rzeczy nie są zbyt przejrzyste, szczególnie terminal, z którego poziomu zarządzanie bazą danych jest niezbyt przyjemnie włączając w to jego złośliwość.

## Ocena środowiska

Środowisko nie jest najbardziej przyjaznym środowiskiem dla użytkownika. Zdecydowanie należało by poprawić zarówno Studio oraz Terminal.

|  |  |
| --- | --- |
| Tabela Jak środowisko realizuje zagadnienia związane z: | Odpowiedź |
| Obiektowością - Metodami | Środowisko daje możliwość użytkownikowi implementować oraz wykorzystywać metody w trakcie działania bazy. Do implementacji metod i klas, w których te metody się zawierają wykorzystywany jest język ODL. Dokumentacja o metodach mówi, że w ODL programista ma możliwość zdefiniowania typu zwracanej wartości, nazwy czy listy parametrów, a następnie zaprogramowania ciała metody, zgodnie z przyjętym standardem. W większości przypadków metody są procedurami, nie zwracającymi konkretnych wartości, lecz wykonującymi określone operacje w bazie. Wykorzystanie metod odbywa się w oparciu o syntaktykę i mechanizmy obiektowe. Odwołanie do metod w klasach odbywa się zasadniczo jak w językach obiektowych czyli poprzez zasięg. Metody można wykorzystywać przy zapytaniach zgodnie z językiem OQL |
| Obiektowością – Dziedziczeniem, typami abstrakcyjnymi | Środowisko umożliwia zgodnie z paradygmatami obiektowości dziedziczenie i używanie typów abstrakcyjnych do zaprojektowania bazy danych. Zgodnie z językiem ODL, klasa A rozszerza klasę B kiedy w definicji klasy zawrzemy definicję „A extends B”. Typy abstrakcyjne, zgodnie z syntaktyką ODL definiowane są przez słowo „abstract” między modyfikatorami dostępu czyli public, private, protected oraz typami danych void, int, class. |
| Obiektowością – Związkami między klasami (asocjacja, kompozycja) | Dokumentacja środowiska mówi, że możliwa jest asocjacja pomiędzy klasami w projekcie jednak nie zostało to wykorzystane w modelu bazy, a co za tym idzie w implementacji bazy. |
| Obiektowością – Typy danych – proste, złożone | Środowisko udostępnia zarówno proste, jak i złożone typy danych. Umożliwia również implementację własnych typów danych, zależnie od potrzeb. Własny typ danych został wykorzystany w implementacji listy obiektów. |
| Obiektowością - Polimorfizmem | W projekcie wykorzystano polimorfizm do przeciążenia metod stąd wiadomo, że środowisko umożliwia stosowanie mechanizmów polimorfizmu. |
| Obiektowością – Tożsamością danych | Przypisanie obiektom unikalnych identyfikatorów zapewnia ustalenie tożsamości danych (OID) zgodnie z paradygmatami obiektowości baz. |
| Obiektowością – Enkapsulacją | Enkapsulacja w środowisku została uzyskana dzięki modyfikatorom dostępu. |
| Obiektowością – Trwałością danych | W środowisku można wyróżnić obiekty o różnych trwałościach. Dokumentacja mówi, że nieużywany trwały obiekt nie jest usuwany z bazy danych. |
| Administracja – Zarządzanie środowiskiem | Na środowisko składa się kilka pomniejszych narzędzi, które umożliwiają realizację zadań w bazie. Studio odpowiada za zarządzanie projektem bazy oraz implementację. W narzędziu Terminal możemy zarządzać bazą z poziomu konsoli. Właśnie w terminalu wykonywane są polecenia z CRUD. |
| Interfejs – Czy narzędzie posiada API? Dla jakich języków? | W środowisku domyślne API korzytsa z Cache ObjectScript, MultiValueBasic, Basic, Java, JavaScript i Transact-SQL. |
| Środowisko – Czy narzędzie zawiera w sobie środowisko programistyczne? | Środowisko programistyczne, które jest wykorzystywane to Atelier. |
| Skalowalność – Czy narzędzie umożliwia horyzontalne skalowanie środowiska? (rozproszone przetwarzanie, magazynowanie, replikacja) | Zgodnie z dokumentacją można przetwarzać jako działanie rozproszone. Korzysta się wtedy z protokołu ECP. Środowisko jest skalowalne i podczas tego zabiegu nie tracimy funkcjonalności. |
| Multi-model – Czy narzędzie zapewnia inne rodzaje bazy danych? | Oprócz obiektowej bazy danych można implementować wszystkie relacyjne bazy danych. |