WOJSKOWA AKADEMIA TECHNICZNA

im. Jarosława Dąbrowskiego

**Wydział Cybernetyki**



Przedmiot:

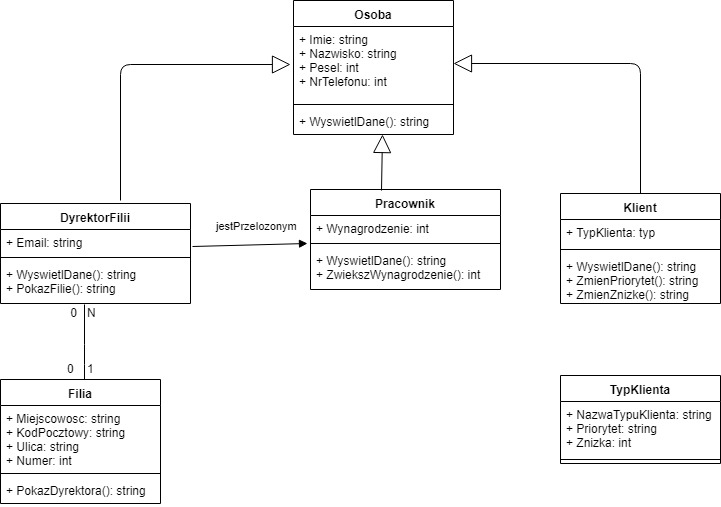
**Systemy Baz Danych**

**SPRAWOZDANIE**

**Projekt obiektowej bazy danych**

|  |  |
| --- | --- |
| Autor:  **sierż. pchor. inż. Michał Duda** | Prowadzący: |
| **sierż. pchor. inż. Mateusz Rzemieniewski** | **mgr inż. Maciej Szymczyk** |
| Grupa: **WCY18IH1S4** |  |

1. Model OBD



Klasy:

|  |  |
| --- | --- |
| Klasa | opis |
| Osoba | Jest to klasa abstrakcyjna zawierająca dane osobowe |
| Klient | Jest to kalsa dziedzicząca po klasie osoba, zawiera osoby będące klientami dla ubezpieczalni w sensie biznesowym |
| Pracownik | Jest to klasa dziedzicząca po klasie osoba, zawiera dane pracowników ubezpieczalni, nie będących dyrektorami. |
| DyrektorFilii | Jest to klasa dziedzicząca po klasie osoba zawierająca pracowników firmy będących dyrektorami. W jednej filii może być wielu dyrektorów (każdy od innej branży) |
| Filia | Jest to konkretny oddział firmy ubezpieczeniowej reprezentowany poprzez swoją lokalizację. |

Związki:

|  |  |
| --- | --- |
| związek | opis |
| jestPrzełożonym | DyrektorFilii jest przełożonym Pracownika |
| pracuje | DyrektorFilii pracuje (jest zatrudniony) w Filii |

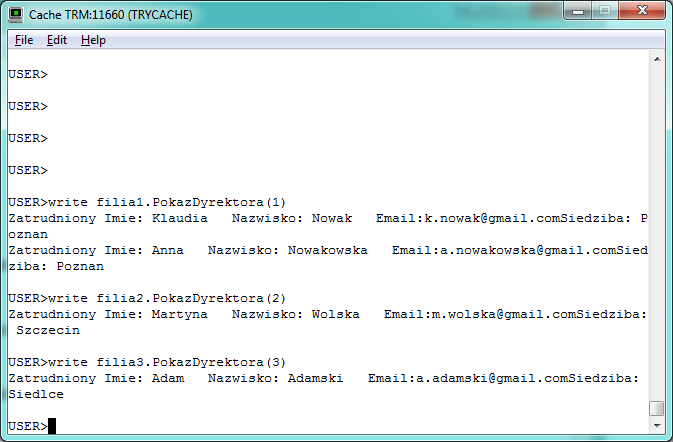
Metody:

|  |  |
| --- | --- |
| metoda | opis |
| WyswietlDane() | Wyswietla dane o Osobie, Pracowniku, DyrektorzeFilii lub o kliencie, wsytępuje tu polimorfizm |
| ZwiekszWynagrodzenie() | Metoda ta powoduje zwiększenie wynagrodzenia Pracownika o zadaną jako argument kwotę |
| PokazDyrektora | Metoda pokazuje kto jest dyrektorem wskazanej Filii |
| ZmienPriorytet() | Metoda ta pozwala zmieniać priorytet na wskazany, wskazanemu Klientowi |
| ZmienZnizke() | Metoda ta pozwala zmieniać znizke na zadeklarowaną w argumencie, wskazanemu Klientowi |

1. Prezentacja wykonanych metod

**Filia**

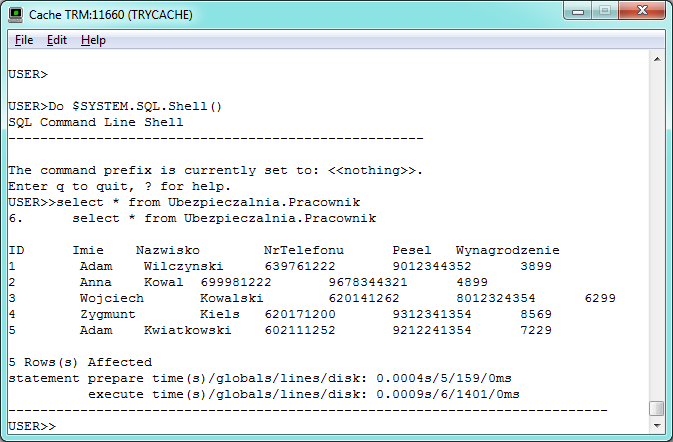
Metoda PokazDyrektora()



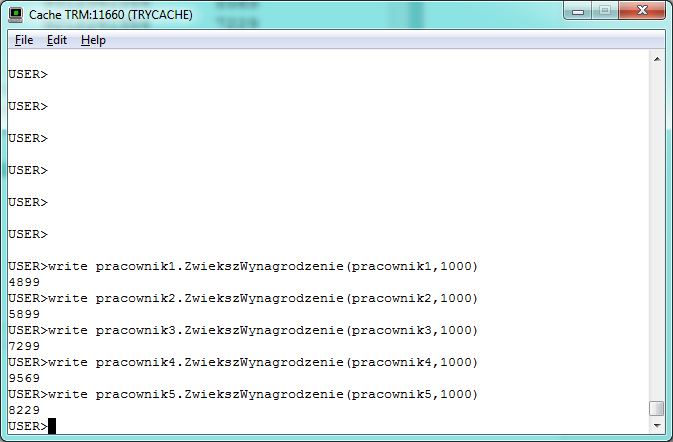
**PRACOWNIK**

Metoda ZwiekszWynagrodzenie()

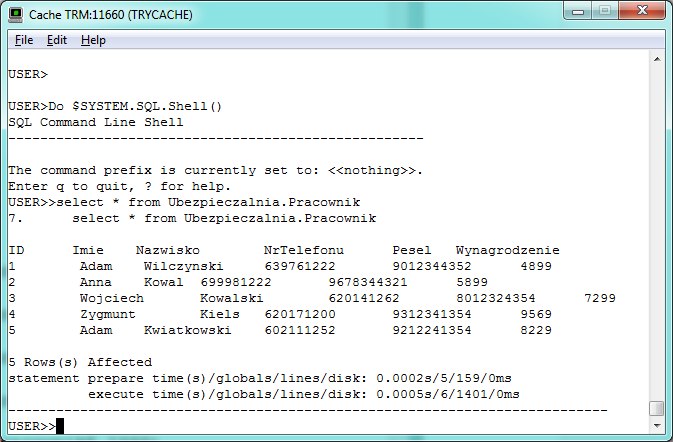
Baza przed użyciem metody:



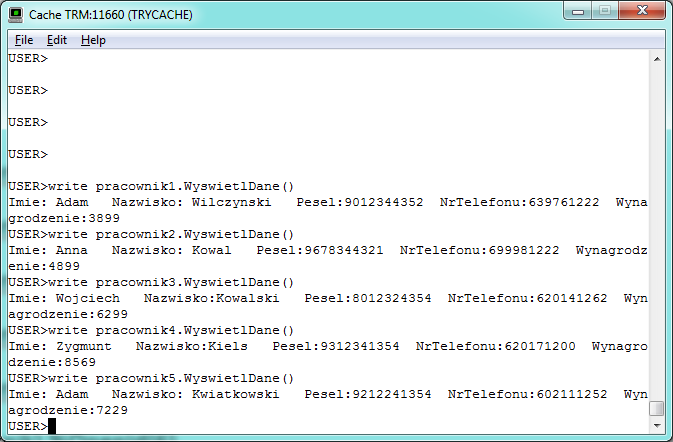
Użycie metody:



Sprawdzenie zmian w bazie:

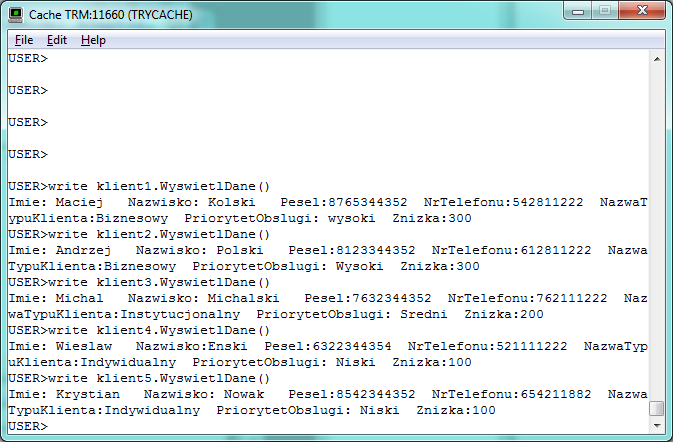


Metoda WyswietlDane()

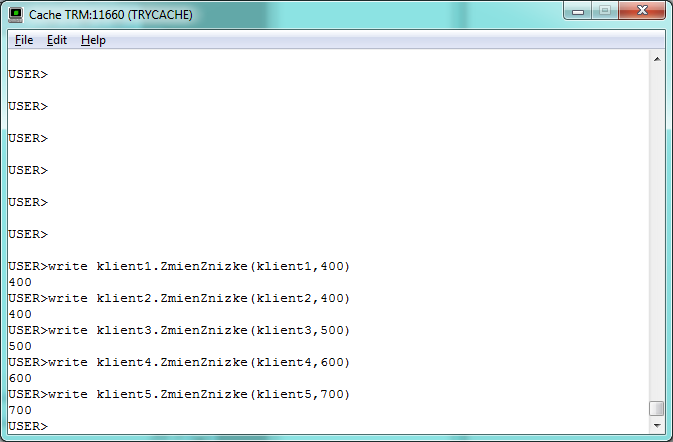


**KLIENT:**

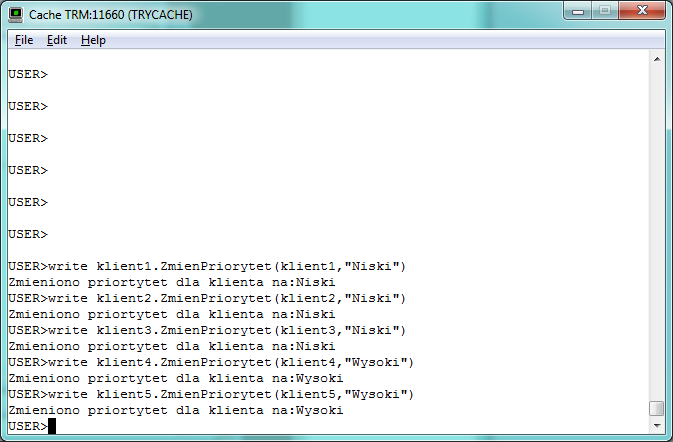
Metoda WyswietlDane()



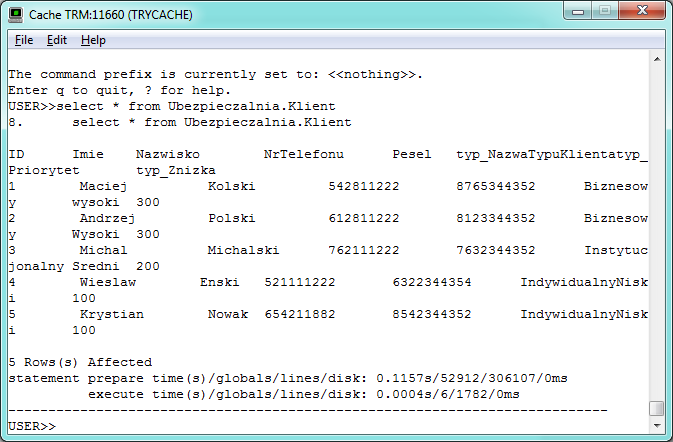
Metoda ZmienZnizke()



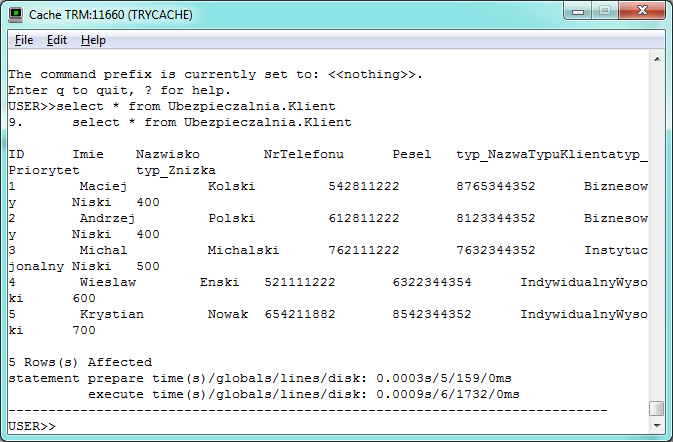
Metoda ZmienPriorytet()



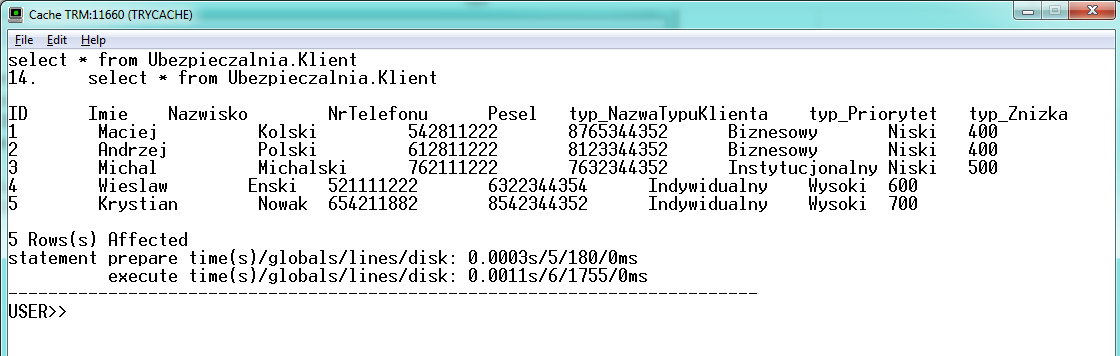
Baza przed zmiana:



Baza po zmianach:

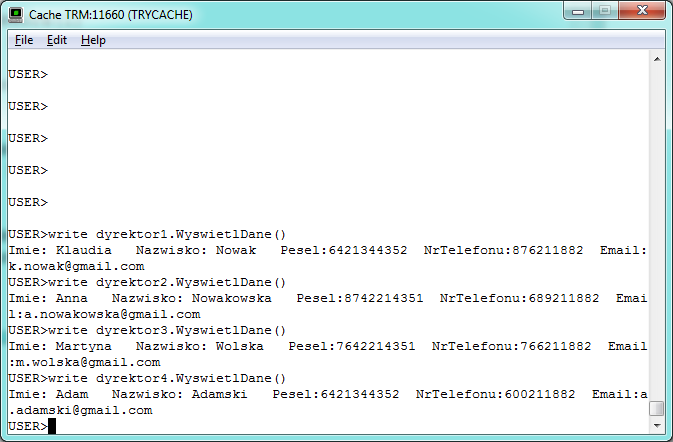


Lepiej widać:

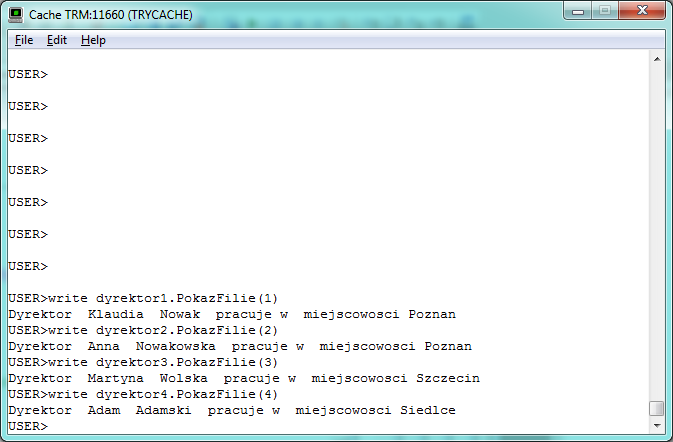


**DyrektorFilii**

Metoda WyswietlDane()



Metoda PokazFilie()



1. Kod utworzonej obiektowej bazy danych

**Filia:**

Class Ubezpieczalnia.Filia Extends (%Persistent, %Populate, %XML.Adaptor, %ZEN.DataModel.Adaptor, %RelationshipObject)  
{  
  
Property Miejscowosc As %String [ Required ];  
  
Property KodPocztowy As %String(MAXLEN = 6) [ Required ];  
  
Property Ulica As %String [ Required ];  
  
Property Numer As %Integer [ Required ];  
  
Relationship zatrudnia As Ubezpieczalnia.DyrektorFilii [ Cardinality = many, Inverse = pracuje ];  
  
Method PokazDyrektora(filiaID As %Integer)  
{  
 Set filia = ##class(Ubezpieczalnia.Filia).%OpenId(filiaID)  
  
 // wypisywanie wszystkich pracowników filii  
 Set key = ""  
  
 Do {  
    Set dyrektor = filia.zatrudnia.GetNext(.key)  
    If (dyrektor '= "") {  
        Write "Zatrudniony ","Imie:",dyrektor.Imie," Nazwisko:",dyrektor.Nazwisko, " Email:",dyrektor.Email, " Siedziba:",filia.Miejscowosc ,!  
    }  
 } While (key '= "")  
  Return ""  
}  
  
Storage Default  
{  
<Data name="FiliaDefaultData">  
<Value name="1">  
<Value>%%CLASSNAME</Value>  
</Value>  
<Value name="2">  
<Value>Miejscowosc</Value>  
</Value>  
<Value name="3">  
<Value>KodPocztowy</Value>  
</Value>  
<Value name="4">  
<Value>Ulica</Value>  
</Value>  
<Value name="5">  
<Value>Numer</Value>  
</Value>  
<Value name="6">  
<Value>zatrudnia</Value>  
</Value>  
<Value name="7">  
<Value>InverseReference</Value>  
</Value>  
<Value name="8">  
<Value>ElementType</Value>  
</Value>  
<Value name="9">  
<Value>Cardinality</Value>  
</Value>  
<Value name="10">  
<Value>InversePersistent</Value>  
</Value>  
<Value name="11">  
<Value>InverseProperty</Value>  
</Value>  
<Value name="12">  
<Value>Persistent</Value>  
</Value>  
</Data>  
<DataLocation>^Ubezpieczalnia.FiliaD</DataLocation>  
<DefaultData>FiliaDefaultData</DefaultData>  
<IdLocation>^Ubezpieczalnia.FiliaD</IdLocation>  
<IndexLocation>^Ubezpieczalnia.FiliaI</IndexLocation>  
<StreamLocation>^Ubezpieczalnia.FiliaS</StreamLocation>  
<Type>%Library.CacheStorage</Type>  
}  
  
}

**DyrektorFilii:**

Class Ubezpieczalnia.DyrektorFilii Extends (%Persistent, %RelationshipObject, %Populate, %XML.Adaptor, %ZEN.DataModel.Adaptor, Ubezpieczalnia.Osoba)  
{  
  
Property Email As %String [ Required ];  
  
Relationship pracuje As Ubezpieczalnia.Filia [ Cardinality = one, Inverse = zatrudnia ];  
  
Method WyswietlDane() As %String  
{  
  Return "Imie:"\_..Imie\_" Nazwisko:"\_..Nazwisko\_" Pesel:"\_..Pesel\_" NrTelefonu:"\_..NrTelefonu\_" Email:"\_..Email  
}  
  
Method PokazFilie(dyrektorID As %Integer)  
{  
 Set dyrektor = ##class(Ubezpieczalnia.DyrektorFilii).%OpenId(dyrektorID)  
  
 // wypisywanie filii, w ktorej pracuje dyrektor  
      
    Set filia = dyrektor.pracuje.Miejscowosc  
    If (filia '= "") {  
        Write "Dyrektor ",dyrektor.Imie, dyrektor.Nazwisko, " pracuje w "," miejscowosci", filia   
    }  
  Return ""  
}  
  
Index Dyrektor On pracuje;  
  
Storage Default  
{  
<Data name="DyrektorFiliiDefaultData">  
<Value name="1">  
<Value>%%CLASSNAME</Value>  
</Value>  
<Value name="2">  
<Value>Email</Value>  
</Value>  
<Value name="3">  
<Value>Filia</Value>  
</Value>  
<Value name="4">  
<Value>Pesel</Value>  
</Value>  
<Value name="5">  
<Value>NrTelefonu</Value>  
</Value>  
<Value name="6">  
<Value>Imie</Value>  
</Value>  
<Value name="7">  
<Value>Nazwisko</Value>  
</Value>  
<Value name="8">  
<Value>pracuje</Value>  
</Value>  
<Value name="9">  
<Value>InverseReference</Value>  
</Value>  
<Value name="10">  
<Value>ElementType</Value>  
</Value>  
<Value name="11">  
<Value>Cardinality</Value>  
</Value>  
<Value name="12">  
<Value>InversePersistent</Value>  
</Value>  
<Value name="13">  
<Value>InverseProperty</Value>  
</Value>  
<Value name="14">  
<Value>Persistent</Value>  
</Value>  
</Data>  
<DataLocation>^Ubezpieczalnia.DyrektorFiliiD</DataLocation>  
<DefaultData>DyrektorFiliiDefaultData</DefaultData>  
<IdLocation>^Ubezpieczalnia.DyrektorFiliiD</IdLocation>  
<IndexLocation>^Ubezpieczalnia.DyrektorFiliiI</IndexLocation>  
<StreamLocation>^Ubezpieczalnia.DyrektorFiliiS</StreamLocation>  
<Type>%Library.CacheStorage</Type>  
}  
  
}

**Klient:**

Class Ubezpieczalnia.Klient Extends (%Persistent, %Populate, %XML.Adaptor, %ZEN.DataModel.Adaptor, Ubezpieczalnia.Osoba)  
{  
  
Property typ As TypKlienta [ Required ];  
  
Method WyswietlDane() As %String  
{  
 Return "Imie:"\_..Imie\_" Nazwisko:"\_..Nazwisko\_" Pesel:"\_..Pesel\_" NrTelefonu:"\_..NrTelefonu\_" NazwaTypuKlienta:"\_..typ.NazwaTypuKlienta\_" PriorytetObslugi: "\_..typ.Priorytet\_" Znizka:"\_..typ.Znizka  
}  
  
Method ZmienPriorytet(klient As %String, priorytet As %String) As %String  
{  
 Set klient.typ.Priorytet = priorytet  
 //zapis do bazy  
 Set sc3 = klient.%Save()  
 if $$$ISERR(sc3)  
 {  
        do $system.Status.DisplayError(sc3)  
     }  
 Return "Zmieniono priortytet dla klienta na:"\_priorytet  
}  
  
Method ZmienZnizke(klient As %String, znizka As %Integer) As %Integer  
{  
  Set klient.typ.Znizka = znizka  
 //zapis do bazy  
 Set sc3 = klient.%Save()  
 if $$$ISERR(sc3)  
 {  
        do $system.Status.DisplayError(sc3)  
     }  
 Return znizka  
}  
  
Storage Default  
{  
<Data name="KlientDefaultData">  
<Value name="1">  
<Value>%%CLASSNAME</Value>  
</Value>  
<Value name="2">  
<Value>Imie</Value>  
</Value>  
<Value name="3">  
<Value>Nazwisko</Value>  
</Value>  
<Value name="4">  
<Value>Pesel</Value>  
</Value>  
<Value name="5">  
<Value>NrTelefonu</Value>  
</Value>  
<Value name="6">  
<Value>typ</Value>  
</Value>  
</Data>  
<DataLocation>^Ubezpieczalnia.KlientD</DataLocation>  
<DefaultData>KlientDefaultData</DefaultData>  
<IdLocation>^Ubezpieczalnia.KlientD</IdLocation>  
<IndexLocation>^Ubezpieczalnia.KlientI</IndexLocation>  
<StreamLocation>^Ubezpieczalnia.KlientS</StreamLocation>  
<Type>%Library.CacheStorage</Type>  
}  
  
}

**TypKlienta:**

Class Ubezpieczalnia.TypKlienta Extends (%SerialObject, %Populate, %XML.Adaptor, %ZEN.DataModel.Adaptor)  
{  
  
Property NazwaTypuKlienta As %String [ Required ];  
  
Property Priorytet As %String [ Required ];  
  
Property Znizka As %Integer [ Required ];  
  
Storage Default  
{  
<Data name="TypKlientaState">  
<Value name="1">  
<Value>NazwaTypuKlienta</Value>  
</Value>  
<Value name="2">  
<Value>Priorytet</Value>  
</Value>  
<Value name="3">  
<Value>Znizka</Value>  
</Value>  
</Data>  
<State>TypKlientaState</State>  
<StreamLocation>^Ubezpieczalnia.TypKlientaS</StreamLocation>  
<Type>%Library.CacheSerialState</Type>  
}  
  
}

**Osoba:**

Class Ubezpieczalnia.Osoba Extends (%Persistent, %Populate) [ Abstract ]  
{  
  
Property Imie As %String [ Required ];  
  
Property Nazwisko As %String [ Required ];  
  
Property Pesel As %Integer [ Required ];  
  
Property NrTelefonu As %Integer [ Required ];  
  
Method WyswietlDane() As %String  
{  
 Return "Imie:"\_..Imie\_" Nazwisko:"\_..Nazwisko\_" Pesel:"\_..Pesel\_" NrTelefonu:"\_..NrTelefonu  
}  
  
Storage Default  
{  
<Data name="OsobaDefaultData">  
<Value name="1">  
<Value>%%CLASSNAME</Value>  
</Value>  
<Value name="2">  
<Value>Imie</Value>  
</Value>  
<Value name="3">  
<Value>Nazwisko</Value>  
</Value>  
<Value name="4">  
<Value>Pesel</Value>  
</Value>  
<Value name="5">  
<Value>NrTelefonu</Value>  
</Value>  
</Data>  
<DataLocation>^Ubezpieczalnia.OsobaD</DataLocation>  
<DefaultData>OsobaDefaultData</DefaultData>  
<IdLocation>^Ubezpieczalnia.OsobaD</IdLocation>  
<IndexLocation>^Ubezpieczalnia.OsobaI</IndexLocation>  
<StreamLocation>^Ubezpieczalnia.OsobaS</StreamLocation>  
<Type>%Library.CacheStorage</Type>  
}  
  
}

**Pracownik:**

Class Ubezpieczalnia.Pracownik Extends (%Persistent, %Populate, %XML.Adaptor, %ZEN.DataModel.Adaptor, Ubezpieczalnia.Osoba)  
{  
  
Property Wynagrodzenie As %Integer [ Required ];  
  
Method WyswietlDane() As %String  
{  
  Return "Imie:"\_..Imie\_" Nazwisko:"\_..Nazwisko\_" Pesel:"\_..Pesel\_" NrTelefonu:"\_..NrTelefonu\_" Wynagrodzenie:"\_..Wynagrodzenie  
}  
  
Method ZwiekszWynagrodzenie(pracownik As %String, Arg1 As %Integer) As %Integer  
{  
 set ..Wynagrodzenie = ..Wynagrodzenie + Arg1   
 Set pracownik.Wynagrodzenie = ..Wynagrodzenie  
 //zapis do bazy  
 Set sc3 = pracownik.%Save()  
 if $$$ISERR(sc3)  
 {  
        do $system.Status.DisplayError(sc3)  
     }  
 set zarobek = ..Wynagrodzenie  
 Return zarobek  
}  
  
Storage Default  
{  
<Data name="PracownikDefaultData">  
<Value name="1">  
<Value>%%CLASSNAME</Value>  
</Value>  
<Value name="2">  
<Value>Imie</Value>  
</Value>  
<Value name="3">  
<Value>Nazwisko</Value>  
</Value>  
<Value name="4">  
<Value>Pesel</Value>  
</Value>  
<Value name="5">  
<Value>NrTelefonu</Value>  
</Value>  
<Value name="6">  
<Value>Wynagrodzenie</Value>  
</Value>  
</Data>  
<DataLocation>^Ubezpieczalnia.PracownikD</DataLocation>  
<DefaultData>PracownikDefaultData</DefaultData>  
<ExtentSize>14</ExtentSize>  
<IdLocation>^Ubezpieczalnia.PracownikD</IdLocation>  
<IndexLocation>^Ubezpieczalnia.PracownikI</IndexLocation>  
<Property name="%%CLASSNAME">  
<AverageFieldSize>1</AverageFieldSize>  
<Selectivity>100.0000%</Selectivity>  
</Property>  
<Property name="%%ID">  
<AverageFieldSize>3</AverageFieldSize>  
<Selectivity>1</Selectivity>  
</Property>  
<Property name="Imie">  
<AverageFieldSize>7.86</AverageFieldSize>  
<Selectivity>7.1429%</Selectivity>  
</Property>  
<Property name="Nazwisko">  
<AverageFieldSize>4.79</AverageFieldSize>  
<Selectivity>7.1429%</Selectivity>  
</Property>  
<Property name="NrTelefonu">  
<AverageFieldSize>1</AverageFieldSize>  
<Selectivity>14.2857%</Selectivity>  
</Property>  
<Property name="Pesel">  
<AverageFieldSize>1</AverageFieldSize>  
<Selectivity>14.2857%</Selectivity>  
</Property>  
<Property name="Wynagrodzenie">  
<AverageFieldSize>9</AverageFieldSize>  
<Selectivity>7.1429%</Selectivity>  
</Property>  
<SQLMap name="IDKEY">  
<BlockCount>-4</BlockCount>  
</SQLMap>  
<StreamLocation>^Ubezpieczalnia.PracownikS</StreamLocation>  
<Type>%Library.CacheStorage</Type>  
}  
  
}

Podsumowanie:

Napotkane problemy:

-mała ilość dostępnych przykładów, jak również uboga dokumentacja skutkowały koniecznością poświęcenia dużej ilości czasu na wypracowanie rozwiązań

-podczas deklarowania metod były problemy z używaniem języka SQL, w szczególności napotkaliśmy problem podczas próby wyświetlenia wartości, które były typu string. Dla typu int nie było takich problemów

-metoda, która ma na celu trwałą zmianę rekordów w bazie danych wymagała podawania jako argument nazwy atrybutu jaki ma być zmieniony, co wywołało problemy i kosztowało dużo czasu zanim znalezione zostało odpowiednie rozwiązanie.

Ocena środowiska:

Środowisko umożliwiła zbudowanie obiektowej bazy danych. Jest możliwość stosowania typowych dla obiektowości cech takich jak dziedziczenie czy polimorfizm. Można korzystać z własnych typów danych. Wytworzenie OBD w tym środowisku nastręcza jednak trudności, nie tyle w tworzeniu samej struktury bazy danych co w szczególności podczas deklarowania metod. Są pewne nieścisłości donoście składni SQL.

|  |  |
| --- | --- |
| Jak środowisko realizuje zagadnienia związane z: | Odpowiedź |
| Obiektowością - Metodami | Pozwala tworzyć metody |
| Obiektowością – Dziedziczeniem, typami abstrakcyjnymi | Baza danych miała cechy obiektowości takie jak dziedziczenie oraz klasy abstrakcyjne. |
| Obiektowością – Związkami między klasami (asocjacja, kompozycja) | Pozwala tworzyć takie związki |
| Obiektowością – Typy danych – proste, złożone | Można tworyzć własne typy dnaych, jak również wykorzystywać proste typy danych |
| Obiektowością - Polimorfizmem | występuje |
| Obiektowością – Tożsamością danych | tak |
| Obiektowością – Enkapsulacją | tak |
| Obiektowością – Trwałością danych | Dane są trwale zapisane w bazie |
| Administracja – Zarządzanie środowiskiem | Umiarkowanie przejrzyste |
| Interfejs – Czy narzędzie posiada API? Dla jakich języków? | .NET |
| Środowisko – Czy narzędzie zawiera w sobie środowisko programistyczne? | Można pisać zapytanie SQL, jednakże skłądnia SQL nie zawsze działała poprawnie, być może nie znaleźliśmy dobrego rozwiązania, jednak cel zakładany został zawsze osiągnięty, niejako „na około” |
| Skalowalność – Czy narzędzie umożliwia horyzontalne skalowanie środowiska? (rozproszone przetwarzanie, magazynowanie, replikacja) | Nie umożliwia |
| Multi-model – Czy narzędzie zapewnia inne rodzaje bazy danych? | Nie zapewnia |