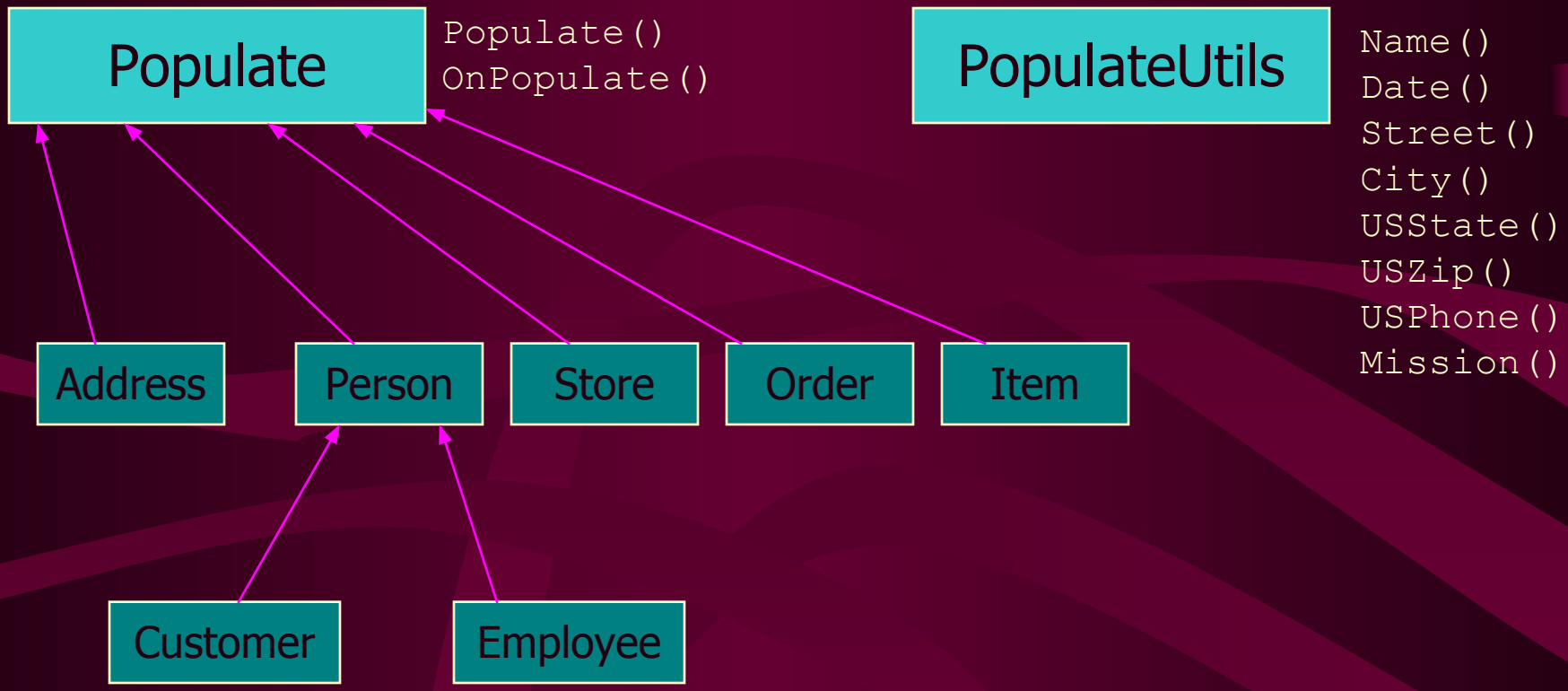
# Модуль: Методы

InterSystems
CACHE

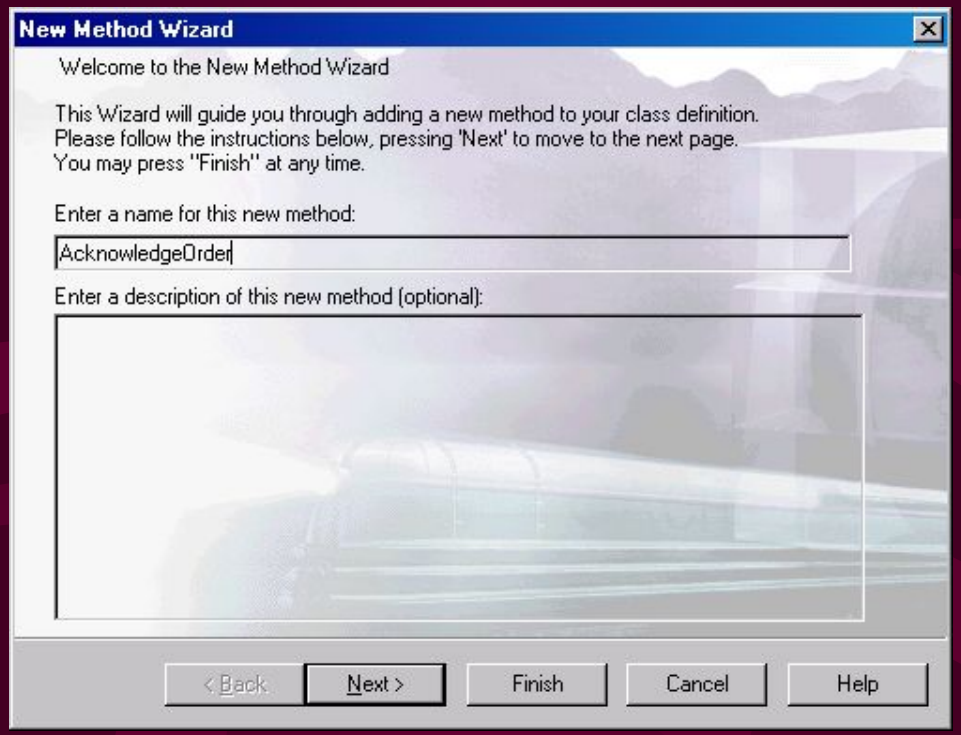
# Иерархия: Классы Population



# Методы

- Caché предоставляет набор методов для пользовательских классов.
  - Методы наследуются из системных классов.
- Создавайте собственные методы для описания бизнес логики приложения.
- Сигнатура метода определяет
  - имя,
  - Возвращаемое значение,
  - Спецификацию формальных аргументов,
  - Характеристики и
  - Код метода.

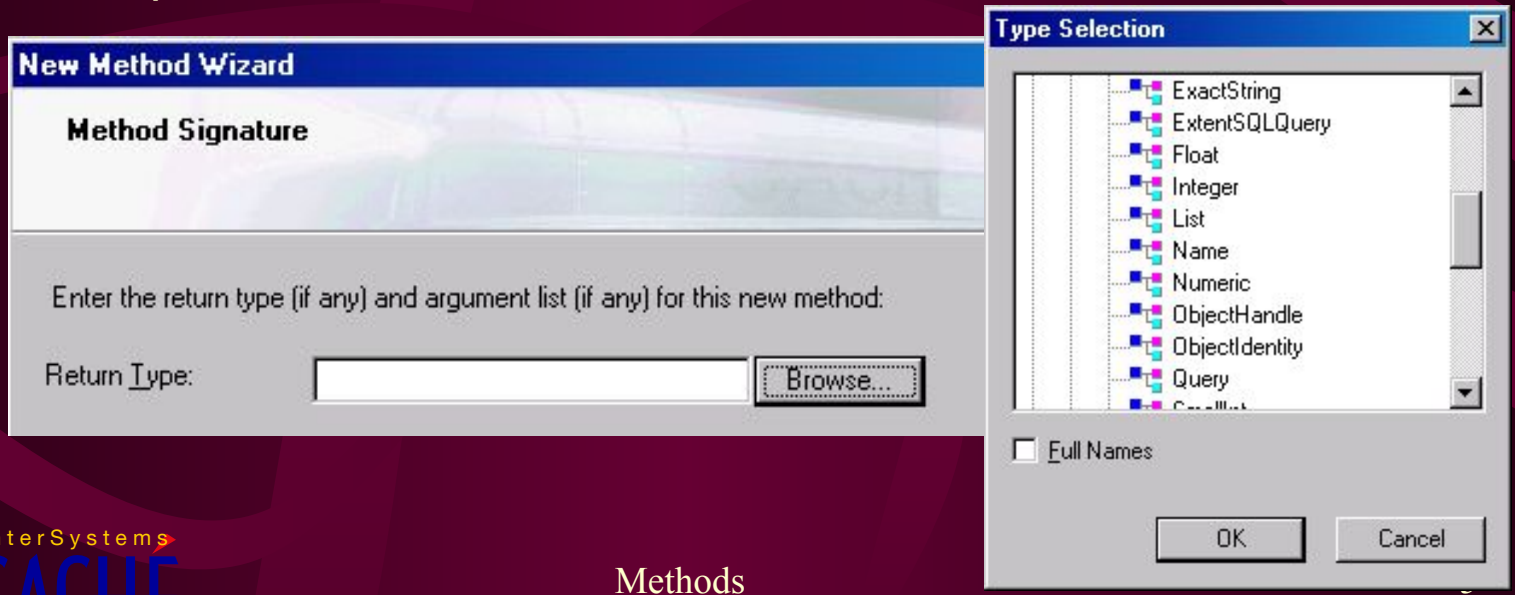
# Новый метод




- Используйте мастер для создания нового метода.

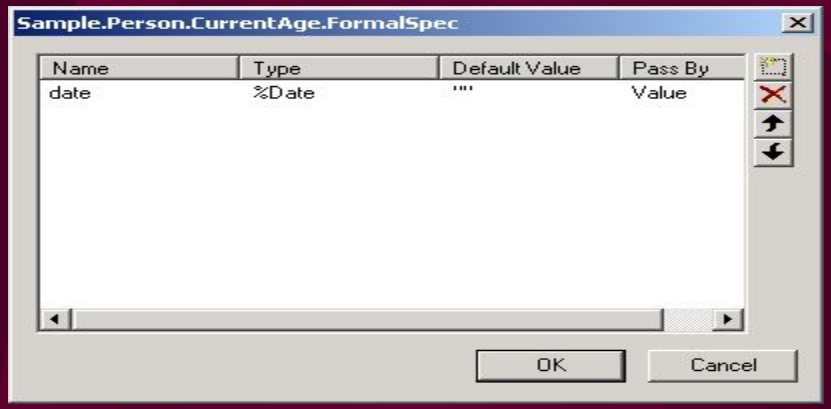
# Возвращаемое значение

- Сигнатура каждого метода определяет тип возвращаемого значения.
- Метод может возвращать значение любого определенного типа данных.
- Каждый метод должен возвращать значение:
  - Это или результат выполнения кода метода или
  - Статус, определяющий успешное или неудачное завершение метода.



# Аргументы

- Сигнатура метода определяет:
  - Список аргументов,
  - Тип данных каждого аргумента и
  - Способ передачи аргумента (значение или ссылка).
- Метод оперирует переменными, определенными в качестве аргументов.
- Любое количество аргументов может быть определено для метода.
- По умолчанию аргументы имеют тип %String.



# Ссылка vs. Значение

- Укажите способ передачи каждого аргумента, для определения синтаксиса вызова метода.
- В отличие от других языков, пользователь определяет способ передачи аргумента, а не код метода.
- Поставьте точку перед аргументом для передачи аргумента по ссылке.
  - Передача по значению: `do obj.Method(a,b)`
  - Передача по ссылке: `do obj.Method(.a,.b)`

# Пример

- Пример:

```
add(a,b)
   set a = a + b
   quit a
```

- При вызове метода таким образом (первый аргумент передается по ссылке) и `sum` и `x` будут содержать сумму `x+y`. При модификации `a` в коде метода, модифицируется и переменная `x` :

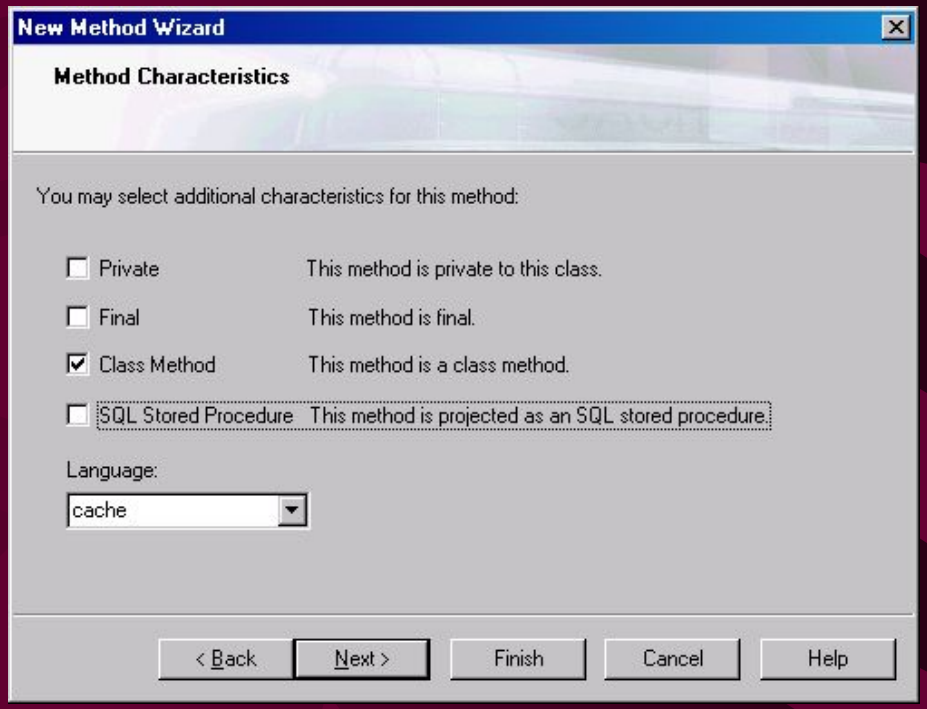
```
set sum = obj.add(.x,y)
```

- Если вызвать метод следующим образом (передается значение первого аргумента), только `sum` будет содержать сумму `x+ y`; `x` останется неизменной:

```
set sum = obj.add(x,y)
```

# Характеристики

- Метод типа Private может быть вызван только из методов того же класса.
- Метод типа Final не может быть переопределен в классах-наследниках.
- Метод, который не является методом класса – это метод объекта.



# Метод класса и метод объекта

- Метод объекта может быть применен только для конкретного открытого объекта.

```
do cust.%Save()  ; записать ЭТОТ объект customer!
```

- Метод класса применим к классу в целом, а не к конкретному объекту. Используйте директиву ##class:

```
set cu = ##class(User.Customer).%OpenId(id)
set mi = ##class(User.MenuItem).%New()
```

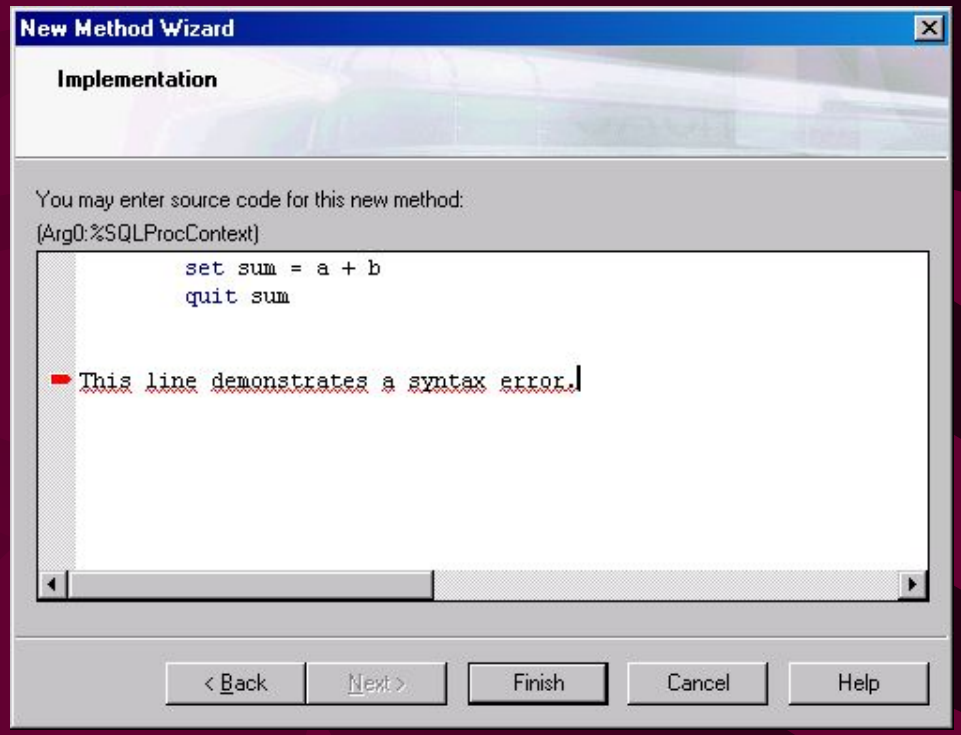
- Метод класса (обычно написанный на ObjectScript) при необходимости, проецируется как хранимая процедура SQL.
  - Выполнить хранимую процедуру можно из любого SQL-клиента, например, MS Access.

# Код метода

- Код метода содержит одну или большее количество строк Caché ObjectScript или Caché Basic.
- Код метода может также включать выражения SQL и таги HTML.

# Код метода

- Каждая строка кода метода должна начинаться со знака табуляции.
- Для указания значения, возвращаемого методом, используйте команду Quit с аргументом.
- Ошибки синтаксиса помечаются красным цветом.
- Для метода-выражения, выражение не должно начинаться со знака табуляции.



# Относительный точечный синтаксис

- Синтаксис “..” используется для вызова метода или получения значения свойства того же класса.
- Например, в классе Order:
  - ..Time Свойство Time текущего объекта.
  - ..Print() Метод Print() объекта класса Order.
  - ..#XYZ Параметр XYZ класса Order.
  - ..Customer.Name Свойство Name объекта, на который ссылается свойство Customer открытого объекта.
  - $this Ссылка на себя

# Обработка исключений

- Используйте механизм TRY-CATCH

```
TRY {
    protected statements
} CATCH [ErrorHandle] {
    error statements
}
```

- При возникновении ошибки в коде, заключенном в TRY обработка сразу переходит на соответствующий блок CATCH.
- `ErrorHandle` – «исключение», объект класса наследника `%Exception.AbstractException`
- Используйте THROW, если необходимо выбросить исключение

# Обработка исключений

- Пример:

```
ClassMethod div(num As %Float, div As %Float) As %Float
{
 TRY {
  SET ans=num/div
 } CATCH errobj {
  IF errobj.Name="<DIVIDE>" { SET ans=0 }
  ELSE { THROW }
 }
 QUIT ans
}
```

# Обработка исключений

- Создание исключений
  - Создайте новый класс наследник **%Exception.AbstractException**
  - Если метод возвращает статус можно использовать метод **CreateFromStatus** класса **%Exception.StatusException**

```
s st = pers.%Save()
if (st'=1) {
   THROW ##class(%Exception.StatusException).
   CreateFromStatus(st)
}
```

# Генератор методов

- Генератор методов вызывается во время компиляции класса для динамической генерации методов, в зависимости от особенностей класса.
  - Такие методы, сами по себе, не содержат предопределенный исполнимый код.
- Генератор методов создает эффективный, специализированный код для методов, наследуемых из класса-предка.

# Пример генератора методов

- Метод %Save() - генерируемый метод, наследуется из класса Persistent
- %Save() принимает разные формы в разных классах – пример полиморфизма.

- При сохранении объекта, Caché должна знать какие данные содержит объект, какие критерии корректности применимы к данным и где хранить данные.
- Во время компиляции класса, генератор кода создает соответствующий метод %Save().

# ObjectScript: Форматирование

- Перевод строки.

```
USER>write cust.Name, !, cust.Address.City
Doe,John
Boston
```

- Для создания столбца используйте знак вопроса с указанием размера столбца в символах.

```
USER>write cust.Name, ?20, cust.Address.City
Doe,John              Boston
```

- Вывод символов в двойных кавычках.

```
USER>write cust.Address.City, ", ", cust.Address.State
Boston, MA
```

- Для конкатенации используйте символ подчеркивания.

```
USER>set line3 = cust.Address.City_", "_cust.Address.State
USER>write line3
Boston, MA
```

# ObjectScript: Цикл For

- Цикл For имеет следующий синтаксис:

```
For <var>=<начало>:<шаг>:<конец> { <код> }
```

- В качестве аргументов можно использовать числа, переменные или методы.
- Третий аргумент можно исключить, если в коде цикла определена команда quit
  - Команда quit завершает работу цикла.
  - Note: Цикл без указания третьего параметра и команды quit – это бесконечный цикл.
- Код цикла заключается в фигурные скобки.

# ObjectScript: Условные переходы

- Команда If позволяет контролировать последовательность выполнения фрагментов программы

```
if <условие> {код}
elseif <условие> {код}
else {код}
```

- Сохраните объект.  Если возвращаемый статус не равен 1, то выйти и вернуть статус.

```
set st = cust.%Save()
if (st '= 1) {quit st}
```

- Использование постусловий команд COS

```
quit:(st '= 1) st
```

команда выполнится, если условие истинно