СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью выполнения индивидуальных заданий лабораторного практикума 2 является:

- освоение средств объектно-ориентированного программирования - организации свойств классов, методов, событий, построения конструкторов класса, перегрузки конструкторов и методов класса.
- освоение элементарных приемов объектного анализа систем
- закрепление навыков работы в IDE
- освоение средств создания программных оболочек

В результате выполнения заданий Производственной практики студент должен

знать:

- средства объектного программирования конструкторов, свойств, методов и событий классов;
- средства объектного программирования механизма перегрузки конструкторов и методов;

уметь:

- использовать средства объектного программирования свойств, методов и событий классов;
- использовать средства объектного программирования механизма перегрузки конструкторов и методов;
- строить и обрабатывать меню приложения.

При выполнении индивидуальных заданий студенты получают первые навыки объектного программирования – использования свойств, методов и событий объектов.

В процессе выполнения заданий студенты осваивают методы отладки алгоритмов, приемы визуального проектирования приложений.

Осваивают технологии тестирования программных моделей систем.

Полученные навыки в дальнейшем могут быть использованы при построении программных моделей экономических систем.

1 УКАЗАНИЯ К ОФОРМЛЕНИЮ И ВЫПОЛНЕНИЮ РАБОТЫ

1.1 Порядок выполнения работы

Шаг 1
Подготовка шаблона файла отчета Report_LabPr_2_Ek_02_b_Sidorov.doc

Шаг 2
Построение оболочки программной модели

Шаг 3
Решение задач практикума

Шаг 4
Редактирование файла отчета

Шаг 5
Защита практикума

1.2 Задания Практикума 2

Практикум 2 является продолжением практикума 1 - помимо проектов решаемых задач, имеет проект приложения, содержащего меню, в котором подключаются исполнимые модули проектов задач, решенных в рамках практикумов 1 и 2. Таким образом, практикум 2 – это проект-оболочка плюс столько проектов, сколько задач в практикуме 2 и плюс столько исполнимых модулей проектов практикума 1, сколько задач в практикуме 1.

Кроме того, к практикуму 2 следует сделать пояснительную записку, которая имеет титульный лист, лист заданий и текстовую часть – решения задач, поданных по плану задач практикума 1.

Ниже приведен лист заданий практикума 2.

Лист заданий

1 Построить проект приложения Project_LabPr_2_Shell в среде Visual Basic .NET с меню вида:

```
File    LabPr_1     LabPr_2                      Help
  Exit    Digit       Properties                   Contents
          SySDigit      ReadWrite                  About
          Cepochki      ReadOnly
          Posled        WriteOnly
          Perestanovki  WORM
          Dichotom    Method
          Gold          Method
          Fibonachi     MethodOverloads
          VectorLabPr1              Consrtuctor
          MatrixLabPr1  ConsrtuctorOverloads
                      Inherits
                         Inherit
                      Events
                         EventDynamic
```

2 Построить пояснительную записку к проекту Report_LabPr_2_Ek_02_b_Sidorov.doc

3 Представить архивы Windows проектов Project_Shell, Project_ReadWrite и т.д. и пояснительной записки Report_LabPr_2_Ek_03_b_Sidorov.rar

4 Условия задач: получить у преподавателя.

Задание выдано10.02.2004г.

Студент гр. Эк 02 б - 3з Сидоров А.О.

2 НЕОБХОДИМЫЕ СВЕДЕНИЯ

2.1 Организация свойств классов

Свойства – это то, что определяет состояние объекта.

Свойства – это поля класса или процедуры свойств.

Открытое поле – это Public переменная класса. Свойства, построенные на основе Public переменных класса, не соответствуют принципу инкапсуляции.

Закрытое поле – это Private переменная класса.

Свойства, соответствующие принципу инкапсуляции строят на основе закрытых полей класса и процедур свойств.

Свойство для чтения и записи

Общий вид

Объявляется закрытая переменная класса

```
Private ИмяЗакрытойПеременной as ТипСвойства
```

Строится процедура свойства

```
Public Property ИмяСвойства(Параметры) as ТипСвойства
  get
    Return ИмяЗакрытойПеременной
  end get
  set (ByVal арг as ТипСвойства) as ТипСвойства
    ИмяЗакрытойПеременной = арг
  end set
end Property
```

Так организованное свойство позволяет себя устанавливать и читать.

Свойство только для чтения

Объявляется закрытая переменная класса

```
Private ИмяЗакрытойПеременной as ТипСвойства
```

Строится процедура свойства с ключевым словом ReadOnly

```
Public ReadOnly Property ИмяСвойства(Параметры) as ТипСвойства
  get
    Return ИмяЗакрытойПеременной
  end get
end Property
```

Свойство только для записи

```
Private ИмяЗакрытойПеременной as ТипСвойства
```

Строится процедура свойства с ключевым словом WriteOnly

```
Public WriteOnly Property ИмяСвойства(Параметры) as ТипСвойства
  set (ByVal арг as ТипСвойства) as ТипСвойства
    ИмяЗакрытойПеременной = арг
  end set
end Property
```

Свойство только для Одной записи и многократного чтения

Объявляется закрытая переменная класса

```
Private ИмяЗакрытойПеременной as ТипСвойства
```

Объявляется закрытая переменная класса

```
Private ИмяЗакрытойЛогическойПеременной as Boolean=False
```

Строится процедура свойства

```
Public Property ИмяСвойства(Параметры) as ТипСвойства
  get
    Return ИмяЗакрытойПеременной
  end get
  set (ByVal арг as ТипСвойства) as ТипСвойства
    if NOT ИмяЗакрытойЛогическойПеременной then
      ИмяЗакрытойПеременной = арг
      ИмяЗакрытойЛогическойПеременной =true
    else
      msgbox("Свойство уже установлено!")
    end if
  end set
end Property
```

2.1.1 Задача «ReadWrite»

Постановка задачи

Создать класс A с ReadWrite свойством Name String типа, инициализация которого выполняется значением NameA при создании экземпляра класса.

Сценарий тестирования

Создаем объект класса A, передав конструктору класса значение имени объекта NameA.

Отображаем значение свойства Name

Меняем свойство Name на новое значение

Отображаем значение свойства Name

Объектный анализ

Объектная модель

Пусть объектный анализ системы приводит к следующей объектной модели:

A

Модель потока событий

-

Объектное проектирование

Строим класс A

Спецификации классов

Класс A

Абстракция служащего

Конструктор

Инициирует свойство Name

Свойства

Name – имя объекта ReadWrite String типа

Методы

-

Идея алгоритма

-

Схема алгоритма

-

События

Исходный код

Класс A

```
Class A
  Private mName As String
  Public Sub New(ByVal arg As String)
      mName = arg
  End Sub
  Public Property Name() As String
    Get
      Return mName
    End Get
    Set(ByVal Value As String)
      mName = Value
    End Set
  End Property
End Class
```

Управляющий модуль

```
Sub Main()
  Dim oA As New A("Name A")
  MsgBox("Имя 1 -" & oA.Name)
  oA.Name = "Радаста"
  MsgBox("Имя 2 -" & oA.Name)
End Sub
```

Тестирование алгоритма

Тест 1

Если

Создаем объект класса A, передав конструктору класса значение имени объекта NameA.

Отображаем значение свойства Name

Меняем свойство Name на новое значение

Отображаем значение свойства Name

То ожидается ответ

-

Действительно

...

2.1.2 Задача «WriteOnceReadMany»

Постановка задачи

Создать класс A с ReadWrite свойством Name String типа и WriteOnceReadMany свойством Salary Single типа, инициализация которых выполняется значениями NameA и 1000 соответственно при создании экземпляра класса.

Сценарий тестирования

Создаем объект класса A, передав конструктору класса значение имени объекта NameA и величину зарплаты 1000.

Отображаем значение свойства Name и Salary

Меняем свойства Name и Salary на новые значения

Отображаем значение свойства Name и Salary

Меняем свойство Salary на новое значение

Отображаем значение свойства Salary

Объектный анализ

Объектная модель

Пусть объектный анализ системы приводит к следующей объектной модели:

A

Модель потока событий

-

Объектное проектирование

Строим класс A

Спецификации классов

Класс A

Абстракция служащего

Конструктор

Инициирует свойство Name и Salary

Свойства

Name – имя объекта ReadWrite String типа

Salary – зарплата объекта WORM String типа

Методы

-

Идея алгоритма

-

Схема алгоритма

-

События

-

Исходный код

Класс A

```
Class A
  Private mName As String, mSalary As Decimal
  Dim AllreadySetSalary As Boolean = False
  Public Sub New(ByVal aName As String, _
                 ByVal aSalary As Decimal)
    mName = aName
    mSalary = aSalary
  End Sub
  Public Property Name() As String
    Get
      Return mName
    End Get
    Set(ByVal Value As String)
      mName = Value
    End Set
  End Property
  Public Property Salary() As Decimal
    Get
      Return mSalary
    End Get
    Set(ByVal Value As Decimal)
      If Not AllreadySetSalary Then
        mSalary = Value
        AllreadySetSalary = True
      Else
        MsgBox("Зарплата уже установлена !")
      End If
    End Set
  End Property
End Class
```

Управляющий модуль

```
Sub Main()
  Dim oA As New A("NameA", 1000)
  MsgBox("Имя объекта 1 - " & oA.Name & "Зарплата 1 -" & oA.salary)
  oA.Name = "Radasta"
  oA.salary = 2000
```

```
  MsgBox("Имя объекта 2 - " & oA.Name & "Зарплата 2 -" & oA.salary)
  oA.salary = 3000
End Sub
```

Тестирование алгоритма

Тест 1

Если

Создаем объект класса A, передав конструктору класса значение имени объекта NameA и величину зарплаты 1000.

Отображаем значение свойства Name и Salary

Меняем свойства Name и Salary на новые значения

Отображаем значение свойства Name и Salary

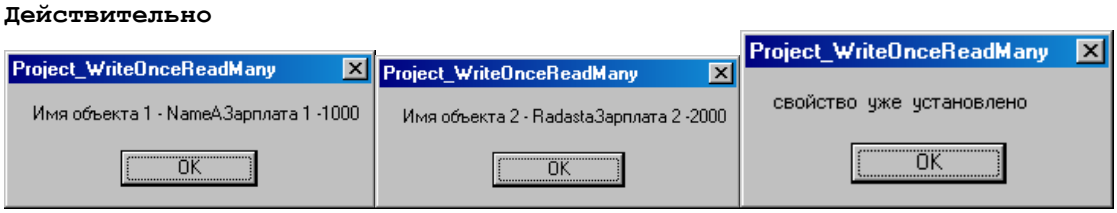
Меняем свойство Salary на новое значение

Отображаем значение свойства Salary

То ожидается ответ

-

Действительно



2.1.3 Задача «ReadOnly»

Постановка задачи

Создать класс A с ReadOnly свойством Name String типа и ReadOnly свойством Salary Decimal типа, инициализация которых выполняется значениями NameA и 1000 соответственно при создании экземпляра класса.

Сценарий тестирования

Создаем объект класса A, передав конструктору класса значение имени объекта NameA величину зарплаты 1000.

Отображаем значение свойства Name и Salary

Меняем свойства Name и Salary на новые значения

Объектный анализ

Объектная модель

Пусть объектный анализ системы приводит к следующей объектной модели:

A

Модель потока событий

-

Объектное проектирование

Строим класс A

Спецификации классов

Класс A

Абстракция служащего

Конструктор

Инициирует свойство Name, Salary

Свойства

Name – имя объекта ReadOnly String типа

Salary – зарплата объекта ReadOnly String типа

Методы

-

Идея алгоритма

-

Схема алгоритма

-

События

-

Исходный код

Класс A

```
Class A
  Private mName As String
  Private mSalary As Decimal
  Public Sub New(ByVal aName As String, _
                 ByVal aSalary As Decimal)
    mName = aName
    mSalary = aSalary
  End Sub
  Public ReadOnly Property Name() As String
    Get
      Return mName
    End Get
  End Property
  Public ReadOnly Property Salary() As Decimal
    Get
      Return mSalary
    End Get
  End Property
End Class
```

Управляющий модуль

```
Sub Main()
  Dim oA As New A("Rada", 1000)
  MsgBox("Имя объекта 1 - " & oA.Name)
  MsgBox("Зарплата объекта 1 - " & oA.Salary)
  oA.Name = InputBox("Введите новое имя - ", "", "NameAAA")
  oA.Salary = InputBox("Введите новое значение зарплаты - ", "", "2000")
End Sub
```

Тестирование алгоритма

Тест 1

Если

Создаем объект класса A, передав конструктору класса значение имени объекта NameA.

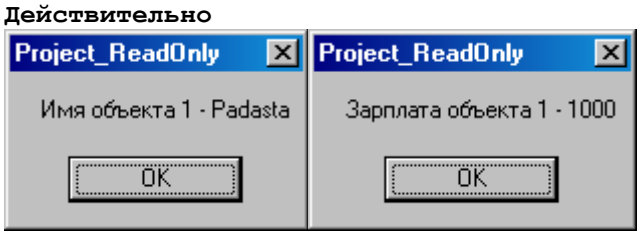
Отображаем значение свойства Name

Меняем свойство Name на новое значение

То ожидается ответ

-

Действительно



2.1.4 Задача «WriteOnce»

Постановка задачи

Создать класс A с ReadOnly свойством Name String типа и ReadOnly свойством Salary Single типа, инициализация которых выполняется значениями NameA и 1000 соответственно при создании экземпляра класса.

Добавить свойство Password WriteOnce String типа, которое инициализируется при создании экземпляра класса значением «q».

Добавить к классу метод RaiseSalary, который по заданному числу процентов увеличивает зарплату на это число процентов, а если повышение зарплаты превышает 10%, то запрашивает при этом еще и пароль.

Сценарий тестирования

Создаем объект класса A, передав конструктору класса значение имени объекта NameA величину зарплаты 1000.

Отображаем значение свойства Name и Salary

Вызываем метод RaiseSalary с уровнем повышения зарплаты менее 10%

Отображаем значение свойства Salary

Вызываем метод RaiseSalary с уровнем повышения зарплаты более 10%

Отображаем значение свойства Salary

Объектный анализ

Объектная модель

Пусть объектный анализ системы приводит к следующей объектной модели:

A

Модель потока событий

-

Объектное проектирование

Строим класс A

Спецификации классов

Класс A

Абстракция служащего

Конструктор

Инициирует свойство Name, Salary, Password

Свойства

Name – имя объекта ReadOnly String типа

Salary - зарплата объекта ReadOnly Decimal типа

Password – пароль WriteOnce Staring типа

Методы

RaiseSalary(ByVal aPercent As Decimal)

Увеличивает зарплату на aPercent процентов, если это не более 10%

Идея алгоритма

-

Схема алгоритма

-

События

-

Исходный код

Класс A

```
Class A
  Private mPassword As String
  Private mName As String
  Private mSalary As Decimal
  Public Sub New(ByVal aName As String, _
                 ByVal aSalary As Decimal)
    mName = aName
    mSalary = aSalary
```

```
    mPassword = ""
  End Sub
  Public ReadOnly Property Name() As String
    Get
      Return mName
    End Get
  End Property
  Public WriteOnly Property Password() As String
    Set(ByVal Value As String)
      mPassword = Value
    End Set
  End Property
  Public ReadOnly Property Salary() As Decimal
    Get
      Return mSalary
    End Get
  End Property
  Public Sub RaiseSalary(ByVal aPercent As Integer)
    If aPercent <= 10 Then
      mSalary = (1 + 0.01 * aPercent) * mSalary
    ElseIf InputBox("Введите пароль - ", "", "w") = mPassword Then
      mSalary = (1 + 0.01 * aPercent) * mSalary
    Else
      MsgBox("Не тот пароль")
    End If
  End Sub
```

Управляющий модуль

```
Sub Main()
  Dim oA As New A("Radasta", 1000)
  MsgBox("Имя объекта 1 - " & oA.Name)
  MsgBox("Зарплата объекта 1 - " & oA.Salary)
  oA.RaiseSalary(10)
  MsgBox("Зарплата объекта 2 - " & oA.Salary)
  oA.Password = "q"
  oA.RaiseSalary(50)
  MsgBox("Зарплата объекта 3 - " & oA.Salary)
End Sub
```

Тестирование алгоритма

Тест 1

Если

Создаем объект класса A, передав конструктору класса значение имени объекта NameA величину зарплаты 1000.

Отображаем значение свойства Name и Salary

Вызываем метод RaiseSalary с уровнем повышения зарплаты менее 10%

Отображаем значение свойства Salary

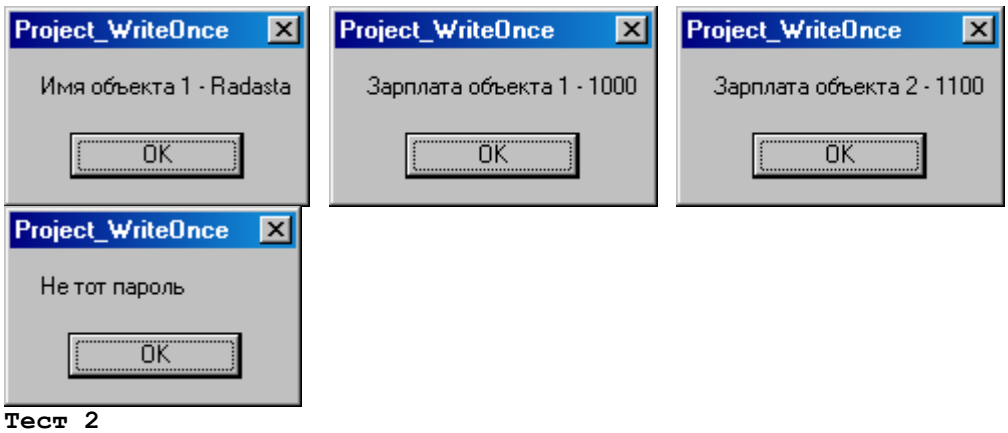
Вызываем метод RaiseSalary с уровнем повышения зарплаты более 50% и паролем «w»

Отображаем значение свойства Salary

То ожидается ответ

-

Действительно



Тест 2

Если

Создаем объект класса A, передав конструктору класса значение имени объекта NameA величину зарплаты 1000.

Отображаем значение свойства Name и Salary

Вызываем метод RaiseSalary с уровнем повышения зарплаты менее 10%

Отображаем значение свойства Salary

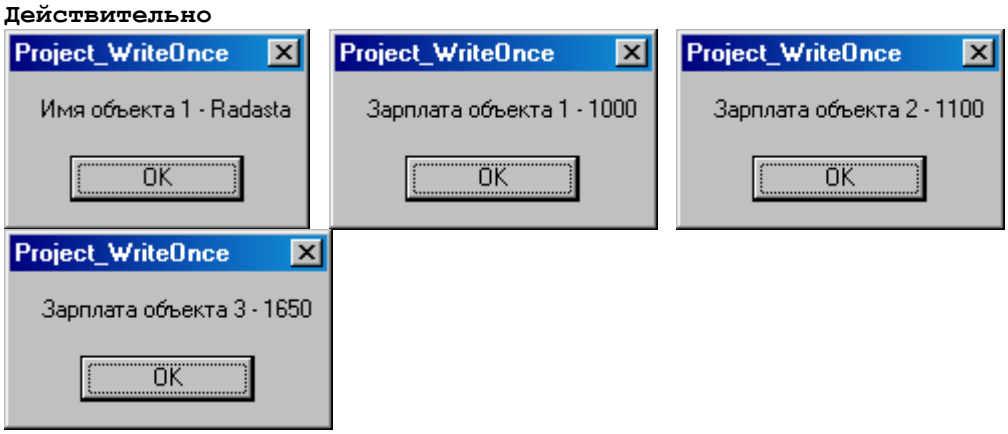
Вызываем метод RaiseSalary с уровнем повышения зарплаты более 50% и паролем «q»

Отображаем значение свойства Salary

То ожидается ответ

-

Действительно



2.2 Организация методов классов

Метод – это то, что «умеет» объект.

Метод – это Sub или Function процедура класса.

Перегружаемый метод помечается словом Overloads.

Выбор перегруженных методов определяется числом параметров, а при их равенстве – их типом.

В перегрузке методов реализуется принцип полиморфизма.

2.2.1 Задача «Method»

Постановка задачи

Создать класс A с ReadOnly свойством Name String типа и ReadOnly свойством Salary Single типа, инициализация которых выполняется значениями NameA и 1000 соответственно при создании экземпляра класса.

Добавить свойство Password WriteOnce String типа, закрытое поле, которого инициализируется при создании экземпляра класса значением «пробел».

Перегрузить конструктор, который в первом случае инициализирует свойства Name и Salary значениями NameA и 1000 соответственно, а в другом случае - свойства Name, Salary и свойство Password значением пароля.

Добавить к классу метод RaiseSalary, который по заданному числу процентов, увеличивает зарплату, если повышение зарплаты не превышает 10%.

Перегрузить метод RaiseSalary так, чтобы он по заданному числу процентов и заданному паролю, увеличивал зарплату более чем на 10%.

Сценарий тестирования

Создаем объект класса A, передав конструктору класса значение имени объекта NameA и величину зарплаты 1000.

Создаем другой объект класса A, передав конструктору класса значение имени объекта NameA, величину зарплаты 1000 и пароль равный «q».

Отображаем значение свойства Name, Salary первого объекта

Отображаем значение свойства Name, Salary второго объекта

Вызываем для первого объекта метод RaiseSalary с уровнем повышения зарплаты менее 10%

Отображаем значение свойства Salary

Вызываем для первого объекта метод RaiseSalary с уровнем повышения зарплаты более 10%

Отображаем значение свойства Salary

Вызываем для второго объекта метод RaiseSalary с уровнем повышения зарплаты менее 10%

Отображаем значение свойства Salary

Вызываем для второго объекта метод RaiseSalary с уровнем повышения зарплаты более 10%

Отображаем значение свойства Salary

Вызываем для второго объекта метод RaiseSalary с уровнем повышения зарплаты более 10% и неверным паролем

Объектный анализ

Объектная модель

Пусть объектный анализ системы приводит к следующей объектной модели:

A

Модель потока событий

-

Объектное проектирование

Строим класс A

Спецификации классов

Класс A

Абстракция служащего

Конструктор 1

Инициирует свойство Name, Salary и закрытое поле mPassword значением пробел

Конструктор 2

Инициирует свойство Name, Salary и Password значением пароля

Свойства

Name – имя объекта ReadOnly String типа

Salary - зарплата объекта ReadOnly Decimal типа

Password – пароль WriteOnce Staring типа

Методы

RaiseSalary(ByVal aPercent As Decimal)

Увеличивает зарплату на aPercent процентов, если это не более 10%

RaiseSalary(ByVal aPercent As Decimal, ByVal aPassword As String)

Увеличивает зарплату на aPercent процентов, если это не более 10%, а если более, тот используется пароль aPassword

Идея алгоритма

-

Схема алгоритма

-

События

-

Исходный код

Класс A

```
Class A
  Private mPassword As String
  Private mName As String
  Private mSalary As Decimal
  Public Sub New(ByVal aName As String, _
                 ByVal aSalary As Decimal)
    mName = aName
    mSalary = aSalary
    mPassword = "q"
  End Sub
  Public ReadOnly Property Name() As String
    Get
      Return mName
    End Get
  End Property
  Public ReadOnly Property Password() As String
    Get
      Return (mSalary)
    End Get
  End Property
  Public ReadOnly Property Salary() As Decimal
    Get
      Return mSalary
    End Get
  End Property
  Public Sub RaiseSalary(ByVal aPercent As Integer)
    If aPercent <= 10 Then
      mSalary = (1 + 0.01 * aPercent) * mSalary
    ElseIf mPassword = " " Then
      MsgBox("Повышение  зарплаты  не  предусмотрено")
    End If
  End Sub
  Public Sub RaiseSalary(ByVal aPercent As Integer, _
                         ByVal aPassword As String)
    If aPercent <= 10 Then
    mSalary = (1 + 0.01 * aPercent) * mSalary
  ElseIf aPassword = mPassword Then
    mSalary = (1 + 0.01 * aPercent) * mSalary
  Else
    MsgBox("Не  тот  пароль")
  End If
End Sub
End Class
```

Управляющий модуль

```
Sub Main()
  Dim oA As New A("Radasta", 1000)
  oA.RaiseSalary(10)
  MsgBox("Зарплата объекта 1 - " & oA.Salary)
  oA.RaiseSalary(20)
  oA.RaiseSalary(10, "w")
  MsgBox("Зарплата объекта 1 - " & oA.Salary)
  oA.RaiseSalary(20, "www")
```

```
  oA.RaiseSalary(20, "q")
  MsgBox("Зарплата объекта  - " & oA.Salary)
End Sub
```

Тестирование алгоритма

Тест 1

Если

Создаем объект класса A, передав конструктору класса значение имени объекта NameA и величину зарплаты 1000.

Создаем другой объект класса A, передав конструктору класса значение имени объекта NameA, величину зарплаты 1000 и пароль равный «q».

Отображаем значение свойства Name, Salary первого объекта

Отображаем значение свойства Name, Salary второго объекта

Вызываем для первого объекта метод RaiseSalary с уровнем повышения зарплаты менее 10%

Отображаем значение свойства Salary

Вызываем для первого объекта метод RaiseSalary с уровнем повышения зарплаты более 10%

Отображаем значение свойства Salary

Вызываем для второго объекта метод RaiseSalary с уровнем повышения зарплаты менее 10%

Отображаем значение свойства Salary

Вызываем для второго объекта метод RaiseSalary с уровнем повышения зарплаты более 10%

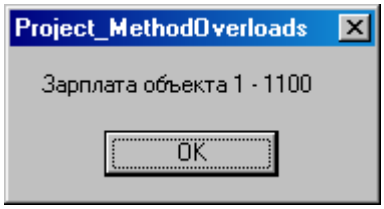
Отображаем значение свойства Salary

Вызываем для второго объекта метод RaiseSalary с уровнем повышения зарплаты более 10% и неверным паролем

То ожидается ответ

-

Действительно



Тест 2

Если

Создаем объект класса A, передав конструктору класса значение имени объекта NameA и величину зарплаты 1000.

Создаем другой объект класса A, передав конструктору класса значение имени объекта NameA, величину зарплаты 1000 и пароль равный «q».

Отображаем значение свойства Name, Salary первого объекта

Отображаем значение свойства Name, Salary второго объекта

Вызываем для первого объекта метод RaiseSalary с уровнем повышения зарплаты менее 10%

Отображаем значение свойства Salary

Вызываем для первого объекта метод RaiseSalary с уровнем повышения зарплаты более 10%

Отображаем значение свойства Salary

Вызываем для второго объекта метод RaiseSalary с уровнем повышения зарплаты менее 10%

Отображаем значение свойства Salary

Вызываем для второго объекта метод RaiseSalary с уровнем повышения зарплаты более 10%

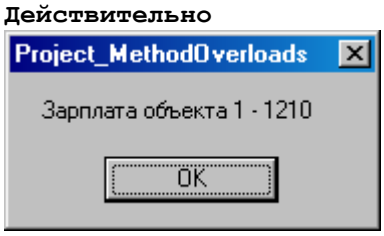
Отображаем значение свойства Salary

Вызываем для второго объекта метод RaiseSalary с уровнем повышения зарплаты более 10% и неверным паролем

То ожидается ответ

…

Действительно



Тест 3

Если

Создаем объект класса A, передав конструктору класса значение имени объекта NameA и величину зарплаты 1000.

Создаем другой объект класса A, передав конструктору класса значение имени объекта NameA, величину зарплаты 1000 и пароль равный «q».

Отображаем значение свойства Name, Salary первого объекта

Отображаем значение свойства Name, Salary второго объекта

Вызываем для первого объекта метод RaiseSalary с уровнем повышения зарплаты менее 10%

Отображаем значение свойства Salary

Вызываем для первого объекта метод RaiseSalary с уровнем повышения зарплаты более 10%

Отображаем значение свойства Salary

Вызываем для второго объекта метод RaiseSalary с уровнем повышения зарплаты менее 10%

Отображаем значение свойства Salary

Вызываем для второго объекта метод RaiseSalary с уровнем повышения зарплаты более 10%

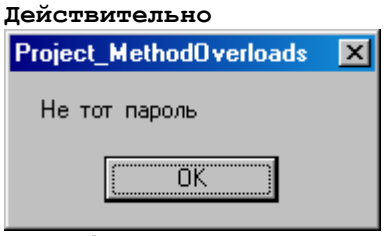
Отображаем значение свойства Salary

Вызываем для второго объекта метод RaiseSalary с уровнем повышения зарплаты более 10% и неверным паролем

То ожидается ответ

-

Действительно



Тест 4

Если

Создаем объект класса A, передав конструктору класса значение имени объекта NameA и величину зарплаты 1000.

Создаем другой объект класса A, передав конструктору класса значение имени объекта NameA, величину зарплаты 1000 и пароль равный «q».

Отображаем значение свойства Name, Salary первого объекта

Отображаем значение свойства Name, Salary второго объекта

Вызываем для первого объекта метод RaiseSalary с уровнем повышения зарплаты менее 10%

Отображаем значение свойства Salary

Вызываем для первого объекта метод RaiseSalary с уровнем повышения зарплаты более 10%

Отображаем значение свойства Salary

Вызываем для второго объекта метод RaiseSalary с уровнем повышения зарплаты менее 10%

Отображаем значение свойства Salary

Вызываем для второго объекта метод RaiseSalary с уровнем повышения зарплаты более 10%

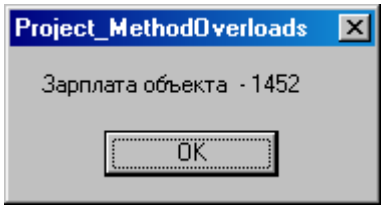
Отображаем значение свойства Salary

Вызываем для второго объекта метод RaiseSalary с уровнем повышения зарплаты более 10% и неверным паролем

То ожидается ответ

-

Действительно



2.2.2 Задача «MethodOverload»

Постановка задачи

Пусть объект Служащий имеет ReadOnly свойство Имя и ReadOnly свойство Зарплата.

Пусть объект Служащий имеет метод RaiseSalary – повышение зарплаты на задаваемое число процентов, а если процент повышения зарплаты выше 10%, то пусть это повышение произойдет только после проверки пароля.

Выполнить тестирование механизма перегрузки методов.

Объектный анализ

Пусть объектный анализ некоторой ЭС приводит к следующей объектной модели.

Employee

Объектное проектирование

Построим класс Employee

Спецификации классов
Класс Employee
Конструктор
Инициализирует свойство Name значением Vita при создании объекта

Инициализирует свойство Salary значением 200 при создании объекта

Свойства
`Name`

имя служащего – закрытое, только для чтения, тип String

`Salary`

зарплата служащего – закрытое, только для записи, тип Decimal

Методы
`RaiseSalary(ПроцентПовышения)`

повышение зарплаты на задаваемое число процентов, если оно не превышает 10%.

`RaiseSalary(ПроцентПовышения, Пароль)`

повышение зарплаты без пароля на задаваемое число процентов, если оно не превышает 10% и повышение зарплаты более чем на 10% при вводе пароля.

Идея алгоритма

-

Схема алгоритма

-

События

-

Исходный код

Класс Employee

```
Class Employee
  Private mName As String
  Private mSalary As Decimal
  Private FlagSalary As Boolean
  Private mPassword As String
  Public Sub New(ByVal argName As String)
      mName = argName
      mSalary = 100
      mPassword = "Пароль"
  End Sub
  Public ReadOnly Property Name() As String
    Get
      Return mName
    End Get
  End Property
  Public Property Salary() As Decimal
    Get
      Return mSalary
    End Get
    Set(ByVal Value As Decimal)
      If FlagSalary Then
        mSalary = Value
      Else
        MsgBox("Свойство Зарплата уже установлено")
      End If
    End Set
  End Property
  Public Overloads Sub RaiseSalary(ByVal arg As Decimal)
    mSalary = (1 + 0.01 * arg) * mSalary
  End Sub
  Public Overloads Sub RaiseSalary(
    ByVal arg As Decimal,
    ByVal Password As String)
      If arg <= 10 Then
        mSalary = (1 + 0.01 * arg) * mSalary
      Else
        Password = InputBox("Введите пароль", "", "")
        If Password = mPassword Then
          mSalary = (1 + 0.01 * arg) * mSalary
        End If
      End If
  End Sub
End Class
```

Управляющий модуль

```
Sub Main()
  Dim oEmployee As New Employee("Vita")
  Dim PercentRaiseSalary As Decimal
  MsgBox("Имя - " & oEmployee.Name)
  MsgBox("Имя - " & oEmployee.Name & "     Зарплата - " & oEmployee.Salary)
  PercentRaiseSalary =
  InputBox("Введите процент повышения зарплаты - ", "", "10")
  oEmployee.RaiseSalary(PercentRaiseSalary)
  MsgBox("Имя - " & oEmployee.Name & "     Зарплата после повышения - " & oEm-
ployee.Salary)
End Sub
```

Тестирование алгоритма

Тест 1

Если

Создадим объект от класса Employee инициализировав через Конструктор свойство Name значением Vita и свойство Salary значением 100.

Отобразим свойство Name.

Вызовем метод RaiseSalary, задав увеличение зарплаты на 10%.

Отобразим свойство Salary.

То ожидается ответ

…

Действительно

…

Тест 2

Создадим объект от класса Employee инициализировав через Конструктор свойство Name значением Vita и свойство Salary значением 100.

Отобразим свойство Name.

Вызовем метод RaiseSalary, задав увеличение зарплаты на 20% и пароль «Пароль».

Отобразим свойство Salary.

То ожидается ответ

…

Действительно

…

2.3 Организация конструкторов классов

Конструктор – это Sub процедура класса с именем New и параметрами, используемыми для инициализации полей класса в момент создания объекта.

Ибо созданный объект без инициализированных членов потенциально опасен при его использовании.

Где-то при создании объекта пишем

```
Dim ИмяОб as ИмяКласса(арг1, арг2,...)
```

Где-то в классе объявляем Поля

```
Dim ИмяПоля1 as тип
Dim ИмяПоля2 as тип
```

и пишем конструктор

```
Public Sub New (арг1 as тип, арг2 as тип,...)
  ИмяПоля1=арг1
  ИмяПоля2=арг2
...
```

End Sub

Перегрузка конструкторов позволяет порождать от класса объекты, отличающиеся инициализированными полями.

В перегрузке конструкторов реализуется принцип полиморфизма.

Где-то при создании объекта пишем

```
Dim ИмяОб1 as ИмяКласса(арг1, арг2,...)
Dim ИмяОб2 as ИмяКласса(арг1, арг2,арг3,...)
Где-то в классе объявляем Поля
Dim ИмяПоля1 as тип
Dim ИмяПоля2 as тип
Dim ИмяПоля3 as тип
```

и пишем конструктор 1

```
Public Sub New (арг1 as тип, арг2 as тип,...)
  ИмяПоля1=арг1
  ИмяПоля2=арг2
...
End Sub
```

и конструктор 2

```
Public Sub New (арг1 as тип, арг2 as тип, арг3 as тип,...)
  ИмяПоля1=арг1
  ИмяПоля2=арг2
  ИмяПоля3=арг3
...
End Sub
```

Бывает удобно писать так

```
Public Sub New (арг1 as тип, арг2 as тип, арг3 as тип,...)
  MyClass.New(арг1,арг2)
  ИмяПоля3=арг3
...
End Sub
```

Выбор перегруженных конструктов определяется числом параметров, а при их равенстве – их типом.

2.3.1 Задача «ConstructorOverload»

Постановка задачи

Пусть объект Служащий имеет ReadOnly свойство Имя, ReadOnly свойство Зарплата и закрытое поле Пароль.

Пусть объект имеет два Конструктора – один с параметром Имя, а другой с параметрами Имя и Пароль.

Пусть объект Служащий имеет метод RaiseSalary – повышение зарплаты на задаваемое число процентов, а если процент повышения зарплаты выше 10%, то пусть это повышение произойдет только после проверки пароля на совпадение с закрытым полем Пароль.

Выполнить тестирование механизма перегрузки конструкторов.

Объектный анализ

Пусть объектный анализ некоторой ЭС приводит к следующей объектной модели.

Employee

Объектное проектирование

Построим класс Employee

Спецификации классов
Класс Employee
Конструктор 1
Инициализирует свойство Name значением Vita при создании объекта

Инициализирует свойство Salary значением 100 при создании объекта

Конструктор 2
Инициализирует свойство Name значением Vita при создании объекта

Инициализирует свойство Salary значением 100 при создании объекта

Инициализирует закрытое поле mPassword значением «Пароль» при создании объекта

Свойства

Name

имя служащего – закрытое, только для чтения, тип String

Salary

зарплата служащего – закрытое, только для записи, тип Decimal

Методы

RaiseSalary(ПроцентПовышения)

повышение зарплаты на задаваемое число процентов, если оно не превышает 10% и повышение зарплаты более чем на 10 процентов, если введен правильный пароль.

Идея алгоритма

-

Схема алгоритма

-

События

-

Исходные коды классов проекта

Класс Employee

```
Class Employee
  Private mName As String
  Private mSalary As Decimal
  Private FlagSalary As Boolean
  Private mPassword, InPassword As String
  Public Sub New(ByVal argName As String)
    mName = argName
    mSalary = 100
  End Sub
  Public Sub New(ByVal argName As String, _
                 ByVal argPassword As String)
    MyClass.New(argName)
    mPassword = argPassword
  End Sub
  Public ReadOnly Property Name() As String
    Get
      Return mName
    End Get
  End Property
  Public Property Salary() As Decimal
    Get
      Return mSalary
    End Get
    Set(ByVal Value As Decimal)
      If FlagSalary Then
        mSalary = Value
      Else
        MsgBox("Свойство Зарплата уже установлено")
      End If
    End Set
  End Property
  Public Sub RaiseSalary(ByVal arg As Decimal)
    If arg <= 10 Then
      mSalary = (1 + 0.01 * arg) * mSalary
    Else
     InPassword = InputBox("Введите пароль", "", "***")
     If InPassword = mPassword Then
       mSalary = (1 + 0.01 * arg) * mSalary
     End If
    End If
  End Sub
End Class
```

Управляющий модуль

```
Sub Main()
  Dim PercentRaiseSalary As Decimal
  Dim oEmployee As New Employee("Vita")
  MsgBox("Имя - " & oEmployee.Name)
  MsgBox("Имя - " & oEmployee.Name & "     Зарплата - " & oEmployee.Salary)
  PercentRaiseSalary = InputBox("Введите процент повышения зарплаты - ", "",
"10")
  oEmployee.RaiseSalary(PercentRaiseSalary)
  MsgBox("Имя - " & oEmployee.Name & "     Зарплата после повышения - " & oEm-
ployee.Salary)
  oEmployee = Nothing
  Dim oEmployee_2 As New Employee("Toma", "Пароль")
  MsgBox("Имя - " & oEmployee_2.Name)
  MsgBox("Имя - " & oEmployee_2.Name & "     Зарплата - " & oEmployee_2.Salary)
  PercentRaiseSalary = InputBox("Введите процент повышения зарплаты - ", "",
"10")
  oEmployee_2.RaiseSalary(PercentRaiseSalary)
  MsgBox("Имя - " & oEmployee_2.Name & "     Зарплата после повышения - " & oEm-
ployee_2.Salary)
  oEmployee_2 = Nothing
End Sub
```

Тестирование проекта

Тест 1

Если

Создадим объект от класса Employee инициализировав через Конструктор свойство Name значением Vita и свойство Salary значением 100.

Отобразим свойство Name.

Вызовем метод RaiseSalary, задав увеличение зарплаты на 10%.

Отобразим свойство Salary.

Создадим объект от класса Employee инициализировав через Конструктор свойство Name значением Toma и свойство Salary значением 100.

Отобразим свойство Name.

Вызовем метод RaiseSalary, задав увеличение зарплаты на 10%.

Отобразим свойство Salary.

То ожидается ответ

…

Действительно

…

Тест 2

Создадим объект от класса Employee инициализировав через Конструктор свойство Name значением Vita и свойство Salary значением 100.

Отобразим свойство Name.

Вызовем метод RaiseSalary, задав увеличение зарплаты на 20% и пароль «Пароль».

Отобразим свойство Salary.

Создадим объект от класса Employee инициализировав через Конструктор свойство Name значением Toma и свойство Salary значением 100.

Отобразим свойство Name.

Вызовем метод RaiseSalary, задав увеличение зарплаты на 20% и пароль «Пароль».

Отобразим свойство Salary.

2.4 Организация наследования

В соответствии с философией объектного анализа и объектного проектирования, когда виртуальная модель системы есть совокупность взаимосвязанных объектов, когда один объект есть частный случай другого, наследование играет существенную роль.

Базовый класс, тот, которого наследуют.

Производный класс, тот, кто наследует.

Общий вид наследования

Inherits ИмяБазового класса

Глубина наследования не ограничена.

Атрибут базового класса свойство, метод могут быть переопределены в производном классе. Обычно переопределяют методы базового класса.

Словом Overridable в базовом классе помечают переопределяемый атрибут, а словом Overrides переопределяющий.

Словом MyBase в производном классе может быть «взят» член базового класса в случае его переопределения в производном классе.

MyBase.ИмяЧленаБазовогоКласса

Для использования в производном классе членов базового класса более высокого уровня, нежели один, требуется, чтобы используемый член был отмечен словом Protected в базовом классе

Например, Protected ReadOnly Rroperty ИмяСвойства as ТипСвйства

И тогда в производном классе можно воспользоваться Protected членом без образования экземпляра базового класса.

В классах же, не входящих в иерархию классов, воспользоваться Protected членом нельзя.

2.4.1 Задача «Inherit»

Постановка задачи

Дана объектная модель некоторой системы.

Вариант xx

```
System X
|
|_ A
|_ B
   |_ C
   |  |_ D
   |_ E
```

Пусть каждый класс объектной модели имеет некоторый Public Main метод – MainИмяКласса.

Тестирование слова Protected

1) Пусть класс верхнего уровня иерархии имеет закрытое Protected ReadWrite String свойство ProtectedPropertyИмяКласса, значение которого инициализируется при создании экземпляра класса величиной «ProtectedReadWritePropertyИмяКласса».

В Main методе производного класса самого нижнего уровня иерархии

a) Отобразить значение свойства защищенного в классе самого верхнего уровня иерархии.

b) Установить новое значение этого свойства величиной «ProtectedReadWritePropertyИмяПроизводногоКласса»

c) Отобразить новое значение этого свойства

d) В Main методе класса, находящегося вне иерархии, отобразить значение этого свойства.

2) Пусть класс верхнего уровня иерархии имеет закрытое ReadWrite String свойство ReadWritePropertyИмяКласса, значение которого инициализируется при создании экземпляра класса величиной «ReadWritePropertyИмяКласса».

a) В Main методе класса, находящегося вне иерархии, отобразить значение этого свойства.

b) Установить новое значение этого свойства величиной «ReadWritePropertyИмяКлассаВнеИерархии»

c) Отобразить новое значение этого свойства

Тестирование слова MyBase

3) Пусть базовый класс для класса самого нижнего уровня иерархии имеет метод Q, принимающий аргумент целого типа и возвращающий его удвоенную величину.

a) В производном классе переопределить этот метод так чтобы он возвращал утроенную величину полученного аргумента.

Вызвать переопределенный метод, а также вызвать соответствующий переопределяемый метод базового класса.

Замечание

1 Отображение значений свойств выполнить оператором MsgBox.

2 Тело Main метода любого класса начинать оператором MsgBox(“Это Main метод класса ИмяКласса”)

Объектный анализ

Объектная модель

Пусть объектный анализ системы приводит к следующей объектной модели:

```
System X
|
|_ A
|_ B
   |_ C
   |  |_ D
   |_ E
```

Модель потока событий

-

Объектное проектирование

Строим классы SystemX, A, B, C, D, E

Спецификации классов

Класс SystemX

Класс самого верхнего уровня иерархии

Конструктор

-

Свойства

-

Методы

-

Идея алгоритма

-

Схема алгоритма

-

События

-

Класс A

Наследует класс SystemX

Конструктор

-

Свойства

-

Методы
MainA

Тела не имеет

Идея алгоритма
-

Схема алгоритма
-

События
-

Класс B
Наследует класс A

Конструктор
Инициирует свойство ProtectedReadWritePropertyA

Инициирует свойство ReadWritePropertyA

Свойства
-

Методы
Q(ByVal arg As Integer)

Отображает - "Это переопределяемый метод базового класса " & 2 * arg)

MainB

Тела не имеет

Идея алгоритма
-

Схема алгоритма
-

События
-

Класс C
Наследует класс B

Конструктор
Без параметров

Свойства
-

Методы
MainC

Отображает значение Protected свойства базового класса ProtectedReadWritePropertyA

Устанавливает новое значение не Protected свойства ProtectedReadWritePropertyA значением ProtectedReadWritePropertyB

Отображает Новое значение Protected свойства базового класса ProtectedReadWritePropertyA

Вводит аргумент для переопределяемого метода базового класса = '', "", "8")

Вызывает переопределенный метода базового класса Q(arg)

Вызывает переопределяемый метода базового класса MyBase.Q(arg)

Идея алгоритма
-

Схема алгоритма
-

События

-

Класс D
Наследует класс SystemX

Конструктор
-

Свойства
-

Методы
MainD

Тела не имеет

Идея алгоритма
-

Схема алгоритма
-

События
-

Класс E
Наследует класс SystemX

Конструктор
-

Свойства
-

Методы
MainE

Вызывает защищенный метод класса иерархии

Изменяет и отображает защищенное свойство класса иерархии

Идея алгоритма
-

Схема алгоритма
-

События
-

Исходный код
```
Class SystemX
End Class
Class A
  Inherits SystemX
  Public Sub MainA()
  End Sub
End Class
Class B
  Inherits SystemX
  Public Sub MainB()
  End Sub
End Class
Class C
  Inherits B
  Private mRWPC As String
  Private mPRWPC As String
  Public Sub New(ByVal aPRWPC As String, _
                 ByVal aRWPC As String)
    mPRWPC = aPRWPC
    mRWPC = aRWPC
  End Sub
```

```
  Protected Property PRWPC() As String
    Get
      Return mPRWPC
    End Get
    Set(ByVal Value As String)
      mPRWPC = Value
    End Set
  End Property
  Public Property RWPC() As String
    Get
      Return mRWPC
    End Get
    Set(ByVal Value As String)
       mRWPC = Value
    End Set
  End Property
  Overridable Function Q(ByVal arg As Integer) _
    As Integer
    Return 2 * arg
  End Function
  Public Sub MainC()
  End Sub
End Class
Class D
  Inherits C
  Public Sub New(ByVal aPRWPC As String, _
                 ByVal aRWPC As String)
    MyBase.New(aPRWPC, aRWPC)
    End Sub
    Overrides Function Q(ByVal arg As Integer) _
      As Integer
      Return 3 * arg
    End Function
    Public Sub MainD()
      MsgBox("Значение старое защищенного  свойства  базового  класса" & PRWPC)
      PRWPC = "NameD"
      MsgBox("Значение новое защищенного  свойства  базового  класса" & PRWPC)
      MsgBox("Значение старое незащищенного  свойства  базового  класса" & RWPC)
      RWPC = "NameDDD"
      MsgBox("Значение новое незащищенного  свойства  базового  класса" & RWPC)

      MsgBox("Переопределённый  метод  " & Q(2))
      MsgBox("Базовый  метод  " & MyBase.Q(2))
    End Sub
End Class
Class E
  Inherits B
  Public Sub MainE()
   'MsgBox("Значение старое защищенного  свойства  базового  класса" & oD.PRWPC)
    MsgBox("Значение старое незащищенного  свойства  базового  класса" &
oD.RWPC)
  End Sub
End Class
```

Управляющий модуль

```
Dim oD As New D("NameC", "NameCC")
Sub Main()
  oD.MainD()
End Sub
```

Тестирование алгоритма

Тест 1

Если

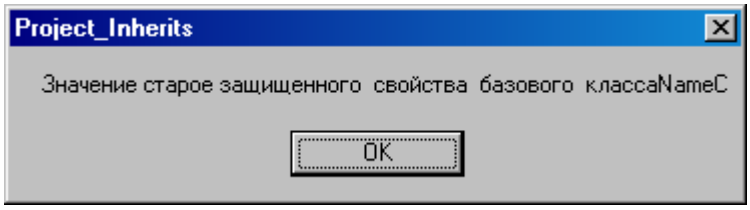
Если вызвать защищенное свойство «внизу» иерархии, то оно будет работать

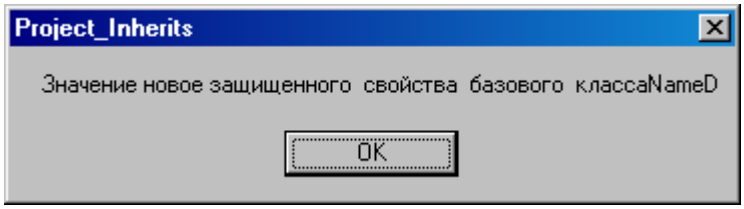
То ожидается ответ

" Значение старое защищенного свойства базового класса" & PRWPC

"Значение новое защищенного свойства базового класса" & PRWPC

Действительно



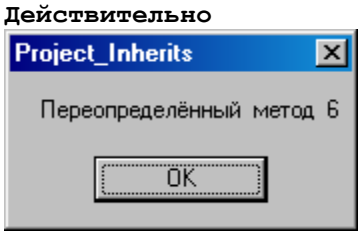


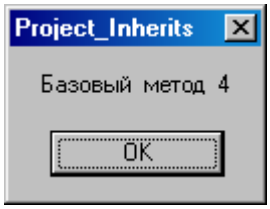
Тест 2
Если
Если вызвать переопределяемый атрибут в производном классе, то он будет работать

То ожидается ответ
"Переопределённый метод " & Q(2)

"Базовый метод " & MyBase.Q(2)

Действительно

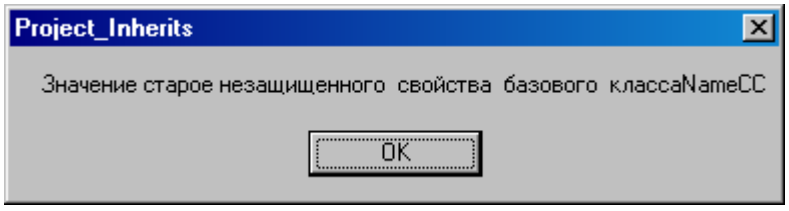




Тест 3
Если
Если вызвать значение не Protected свойства"Значение старое незащищенного свойства базового класса" & RWPC то оно будет работать

То ожидается ответ
"Значение старое незащищенного свойства базового класса" & RWPC

Действительно

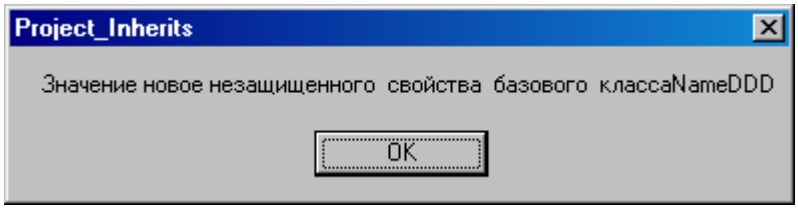


Тест 4
Если
Если изменить значение не Protected свойства "Значение новое незащищенного свойства базового класса" & RWPC, то оно будет работать

То ожидается ответ

"Значение новое незащищенного свойства базового класса" & RWPC

Действительно



2.5 Обработка событий

В соответствии с философией объектного анализа и проектирования, когда виртуальная модель системы есть совокупность взаимодействующих объектов, события, которые генерируются и обрабатываются объектами, играют заметную роль.

Модель обработки событий в VB .NET

При возникновении события, объект источник вызывает определенный метод объекта получателя. Поэтому объекты получатели и их методы, обрабатывающие события, должны быть заранее зарегистрированы в объекте источнике – оповещение путем обратного вызова – Call Back Notification

Источник генерирует событие, но на него откликаются лишь те объекты получатели, которые на него «подписались» фраза AddressOf.

Объекты получатели заранее «подписавшиеся» на событие, которое они собираются обрабатывать, зарегистрированы объектом источником - фраза AddHandler

Схема динамической организации и обработки событий на основе WithEvents.

Где-то в классе, генерирующем событие, а GD объявляем событие

```
...
Public Event ИмяСоб(
  ByVal Sender as ИмяКлГдеГенСоб,
  ByVal e      as ИмяКлОписСоб)
```

Где-то в классе, в процедуре, генерирующей событие, генерируем событие

```
...
RaiseEvent ИмяСоб(ИмяКлГдеГенСоб, New ИмяКлОписСоб())
```

В процедуре, обрабатывающей событие, GD объявляем перехватчик события

```
...
Private WithEvents ИмяПерехвСоб as ИмяКлГдеГенСоб
...
```

и пишем процедуру обработки события

```
Public Sub ИмяПерехвСоб_ИмяСоб(
  ByVal Sender as ИмяРешения.ИмяКлПоляБитвы. ИмяКлГдеГенСоб,
  ByVal e      as ИмяКлОписСоб)
  Handles ИмяПерехвСоб.ИмяСоб
...
```

код обработки события

```
...
End Sub
```

Где-то в Main модуля создаем экземпляр класса, в котором генерируется событие

```
Dim ИмяОб as New ИмяКлГдеГенСоб
```

Динамически связываем источник события ИмяОб.ИмяСоб с процедурой обработки события ИмяПерехвСоб_ИмяСоб

```
AddHandler ИмяОб.ИмяСоб, AddressOf ИмяПерехвСоб_ИмяСоб
```

Вызываем метод класса, в котором генерируется событие

```
ИмяОб.ИмяMainКлГдеГенСоб()
```

Удаляем связь источника события с процедурой обработки события

```
RemoveHandler ИмяОб.ИмяСоб, AddressOf ИмяПерехвСоб_ИмяСоб
```

Схема динамической организации и обработки событий без использования WithEvents.

Где-то в классе, генерирующем событие, в GD объявляем событие

```
Public Event ИмяСоб()
```

Динамически связываем источник события
ИмяОб.ИмяСоб с процедурой обработки события ИмяОб.ИмяПроцОбрабСоб

```
AddHandler ИмяОб.ИмяСоб, _
  AddressOf ИмяОб.ИмяПроцОбрабСоб
```

Генерируем событие

```
RaiseEvent ИмяСоб()
```

Удаляем связь источника события с процедурой обработки события

```
RemoveHandler ИмяОб.ИмяСоб, _
  AddressOf ИмяОб.ИмяПроцОбрабСоб
```

и пишем процедуру обработки события

```
Public Sub ИмяПроцОбрабСоб()
...
```

код обработки события

```
...
End Sub
```

Где-то в Main модуля создаем экземпляр класса, в котором генерируется событие

```
Dim ИмяОб as New ИмяКлГдеГенСоб
AddHandler ИмяОб.ИмяСоб, AddressOf ИмяПерехвСоб_ИмяСоб
```

Вызываем метод класса, в котором генерируется событие

```
ИмяОб.ИмяMainКлГдеГенСоб()
```

Можно сказать, что событие проявляет себя в следующих точках:

1 Точка объявления события

2 Точка связывания события с обработчиком

3 Тока генерации

4 Точка удаления связи события с обработчиком

5 Точка обработки события – обработчик

2.5.1 Задача «EventDynamic»

Постановка задачи

Дана объектная модель и Поток событий некоторой системы. Построить программную модель объектной модели системы.

Вариант xx

```
SystemX
|
```

```
|_ A
|_ B
   |_ C
   |  |_ D
   |_ E
16 17 18 19 20 21 22 23 24
------------------------>
                      Время
A     B     C     D     E
------------>
      ------>
      ------------>
<------------------------
```

Объектный анализ

Объектная модель

Пусть объектный анализ системы приводит к следующей объектной модели:

```
SystemX
|
|_ A
|_ B
   |_ C
   |  |_ D
   |_ E
```

Модель потока событий

Пусть объектный анализ системы приводит к следующей модели Потока событий:

```
16 17 18 19 20 21 22 23 24
------------------------>
                      Время
A     B     C     D     E
------------>
      ------>
      ------------>
<------------------------
```

Объектное проектирование

Строим классы SystemX, A, B, C, D, E

Спецификации классов

Класс SystemX

Класс самого верхнего уровня иерархии

Конструктор

-

Свойства

-

Методы

MainSystemX()

Без тела

Идея алгоритма

-

Схема алгоритма

-

События

-

Класс A

Базовый класс для классов B, C

Наследует класс SystemX

Конструктор

-

Свойства

-

Методы

pev1fromEforA()

Отображает - "Обработано событие 1 от E для A"

MainA

Если пришло время и события ev1fromAforCB не было ранее, то оно генерируется

Идея алгоритма

-

Схема алгоритма

-

События

`ev1fromAforC()`

Событие 1 от Адля C

Класс B

Наследует класс SystemX

Конструктор

-

Свойства

-

Методы

`Runev1fromBforC`

`Runev1fromBforD`

Отображает - "Обработано событие 1 от B для C"

Отображает - "Обработано событие 1 от B для D"

MainB

Если пришло время и события ev1fromBforC не было ранее, то оно генерируется

Если пришло время и события ev1fromBforD не было ранее, то оно генерируется

Идея алгоритма

-

Схема алгоритма

-

События

-

Класс C

Наследует класс B

Конструктор

-

Свойства

-

Методы

`MainC`

Идея алгоритма

-

Схема алгоритма

-

События
pev1fromAforC()

Событие 1 от A для C

Класс D
Наследует класс C

Конструктор
-

Свойства
-

Методы
`pev1fromBforD()`

Отображает - "Обработано событие 1 от B для D"

`MainD`

Если пришло время и события ev1fromBforD не было ранее, то оно генерируется

Если пришло время и события ev1fromBforD не было ранее, то оно генерируется

Идея алгоритма
-

Схема алгоритма
-

События
ev1fromBforD()

Событие 1 от B для D

Класс E
Наследует класс B

Конструктор
-

Свойства
-

Методы
`ev1fromEforA()`

Отображает - "Обработано событие 1 от E для A"

`MainA`

Если пришло время и события ev1fromEforA не было ранее, то оно генерируется

Идея алгоритма
-

Схема алгоритма
-

События
-

Исходный код
```
Class SystemX
End Class

Class A
  Inherits SystemX
  Private mName As String
  Public Event ev1fromAforC()
  Private Runev1fromAforC As Boolean = False
  Public Sub MainA(ByVal myD As String)
```

```
    If "16.04.05" <= myD And myD <= "20.04.05" Then
      If Not Runev1fromAforC Then
        AddHandler Me.ev1fromAforC, _
          AddressOf oC.pev1fromAforC
        RaiseEvent ev1fromAforC()
        RemoveHandler Me.ev1fromAforC, _
          AddressOf oC.pev1fromAforC
        Runev1fromAforC = True
      End If
    End If
  End Sub
  Public Sub New(ByVal aName As String)
    mName = aName
  End Sub
  Public ReadOnly Property Name() As String
    Get
      Return (mName)
    End Get
  End Property
  Public Sub pev1fromEforA()
    MsgBox("Обработанно  событие 1 от E для A")
  End Sub
End Class
Class B
  Inherits SystemX
  Private mName As String
  Public Event ev1fromBforC()
  Public Event ev1fromBforD()
  Private Runev1fromBforC As Boolean = False
  Private Runev1fromBforD As Boolean = False
  Public Sub MainB(ByVal myD As String)
  If "18.04.05" <= myD And myD <= "20.04.05" Then
    If Not Runev1fromBforC Then
      AddHandler Me.ev1fromBforC, _
        AddressOf oC.pev1fromBforC
      RaiseEvent ev1fromBforC()
      RemoveHandler Me.ev1fromBforC, _
        AddressOf oC.pev1fromBforC
      Runev1fromBforC = True
     End If
  End If
  If "18.04.05" <= myD And myD <= "22.04.05" Then
    If Not Runev1fromBforD Then
      AddHandler Me.ev1fromBforD, _
        AddressOf oD.pev1fromBforD
      RaiseEvent ev1fromBforD()
      RemoveHandler Me.ev1fromBforD, _
        AddressOf oD.pev1fromBforD
      Runev1fromBforD = True
    End If
  End If
  End Sub
  Public Sub New(ByVal aName As String)
    mName = aName
  End Sub
  Public ReadOnly Property Name() As String
    Get
      Return (mName)
    End Get
  End Property
End Class
Class C
  Inherits B
  Private mName As String
```

```
  Public Sub New(ByVal aName As String)
    MyBase.New(aName)
  End Sub
  Public Sub pev1fromAforC()
    MsgBox("Обработанно  событие 1 от A для C")
  End Sub
  Public Sub pev1fromBforC()
    MsgBox("Обработанно  событие 1 от B для C")
  End Sub
End Class
Class D
  Inherits C
  Private mName As String
  Public Sub New(ByVal aName As String)
    MyBase.New(aName)
  End Sub
  Public Sub pev1fromBforD()
    MsgBox("Обработанно  событие 1 от B для D")
  End Sub
End Class
Class E
  Inherits B
  Private mName As String
  Public Event ev1fromEforA()
  Private Runev1fromEforA As Boolean = False
  Public Sub MainE(ByVal myD As String)
    If "16.04.05" <= myD And myD <= "24.04.05" Then
      If Not Runev1fromEforA Then
        AddHandler Me.ev1fromEforA, _
          AddressOf oA.pev1fromEforA
        RaiseEvent ev1fromEforA()
        RemoveHandler Me.ev1fromEforA, _
          AddressOf oA.pev1fromEforA
        Runev1fromEforA = True
      End If
    End If
  End Sub
  Public Sub New(ByVal aName As String)
    MyBase.New(aName)
  End Sub
End Class
```

Управляющий модуль

```
Dim oA As New A("NameA")
Dim oB As New B("NameB")
Dim oC As New C("NameC")
Dim oD As New D("NameD")
Dim oE As New E("NameE")
Sub Main()
  Dim myDay As New Date(2005, 4, 15)
  Dim myD As String, i As Integer
  For i = 1 To 11
    myD = myDay.ToShortDateString
    MsgBox("Начался день - " & myD)
    oA.MainA(myD)
    oB.MainB(myD)
    oE.MainE(myD)
    MsgBox("Закончился день - " & myD)
    myDay = myDay.AddDays(1)
  Next i
End Sub
```

Тестирование алгоритма

Тест 1

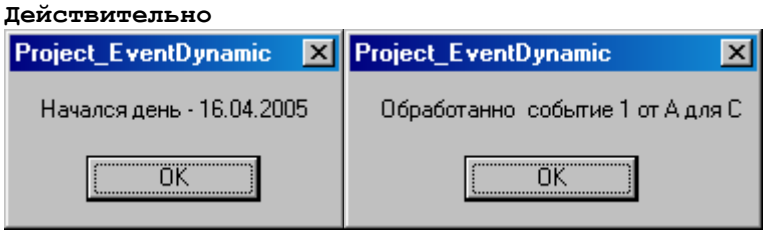
Если

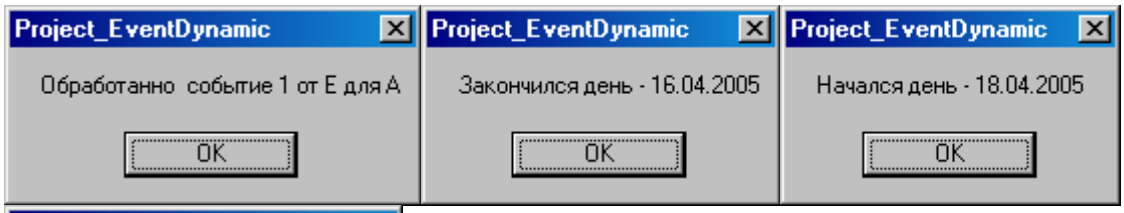
Задана модель потока событий

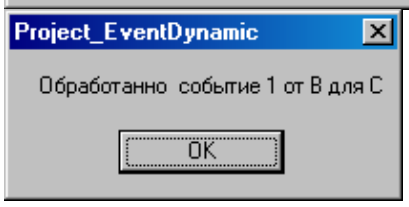
То ожидается ответ

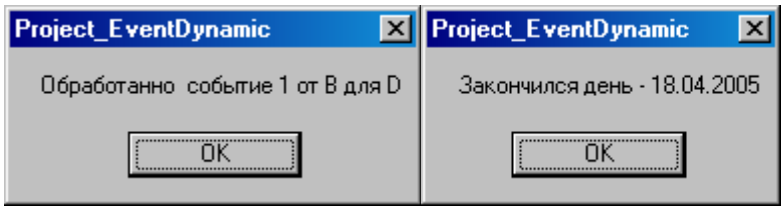
```
"Обработано событие 1 от A для B")
"Обработано событие 1 от C для D")
"Обработано событие 1 от D для E")
"Обработано событие 1 от D для A")
```

Действительно









2.6 Обработка меню приложения

Пусть на этапе проектирования на форму приложения с именем frmMainLabPr2 добавлен экземпляр стандартного класса
MainMenu с именем MainMenuLabPr2.

Организуем реакцию по свойству Text на щелчок элемента меню CType(sender, MenuItem).Text оператором Select Case.

Пусть проект имеет форму с именем frmAbout.

Для отображения формы выполним код:
Dim oAbout As New frmAbout()

oAbout.Show

Ниже приведен примерный код обработки меню приложения.

```
Private Sub MainMenu_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) _
  Handles _
    MenuItemFile.Click, _
    MenuItemExit.Click, _
     MenuItemDigit.Click, _
     MenuItemSysDigit.Click, _
     MenuItemCepochki.Click, _
     MenuItemPosled.Click, _
     MenuItemPerestanovki.Click, _
     MenuItemDichotom.Click, _
     MenuItemGold.Click, _
     MenuItemFibonachi.Click, _
     MenuItemVector.Click, _
```

```
  MenuItemMatrix.Click, _
MenuItemLabPr_2.Click, _
  MenuItemProperties.Click, _
  MenuItemReadOnly.Click, _
  MenuItemReadWrite.Click, _
  MenuItemReadOnly.Click, _
  MenuItemWriteOnceReadMany.Click, _
  MenuItemWrireOnce.Click, _
MenuItemMethods.Click, _
  MenuItemMethod.Click, _
  MenuItemMethodOverloads.Click, _
MenuItemConstructor.Click, _
  MenuItemConstuctorOverloads.Click, _
MenuItemInherits.Click, _
  MenuItemInherit.Click, _
MenuItemEvents.Click, _
  MenuItemEventDynamic.Click, _
MenuItemContents.Click, _
MenuItemAbout.Click
Select Case CType(sender, MenuItem).Text
Case "Exit"
  Me.Close()
Case "Digit"
  Shell("Project_Digit.exe", AppWinStyle.NormalFocus)
Case "SysDigit"
  Shell("Project_SysDigit.exe", AppWinStyle.NormalFocus)
Case "Cepochki"
  Shell("Project_Cepochki.exe", AppWinStyle.NormalFocus)
Case "Posled"
   Shell("Project_Posled.exe", AppWinStyle.NormalFocus)
 Case "Perestanovki"
   Shell("Project_Perestanovki.exe", AppWinStyle.NormalFocus)
 Case "Dichotom"
   Shell("Project_Dichotom.exe", AppWinStyle.NormalFocus)
 Case "Gold"
   Shell("Project_Gold.exe", AppWinStyle.NormalFocus)
 Case "Fibonachi"
   Shell("Project_Fibonachi.exe", AppWinStyle.NormalFocus)
 Case "Vector"
   Shell("Project_Vector.exe", AppWinStyle.NormalFocus)
 Case "Matrix"
   Shell("Project_Matrix.exe", AppWinStyle.NormalFocus)
 Case "ReadWrite"
   Shell("Project_ReadOnly.exe", AppWinStyle.NormalFocus)
 Case "ReadOnly"
   Shell("Project_ReadOnly.exe", AppWinStyle.NormalFocus)
 Case "WriteOnce"
   Shell("Project_WriteOnce.exe", AppWinStyle.NormalFocus)
 Case "WriteOnceReadMany"
   Shell("Project_WriteOnceReadMany.exe", AppWinStyle.NormalFocus)
 Case "Method"
   Shell("Project_Method.exe", AppWinStyle.NormalFocus)
 Case "MethodOverloads"
   Shell("Project_MethodOverloads.exe", AppWinStyle.NormalFocus)
 Case "ConstuctorOverloads"
   Shell("Project_ConstructorOverloads.exe", AppWinStyle.NormalFocus)
 Case "Inherit"
   Shell("Project_Inherits.exe", AppWinStyle.NormalFocus)
 Case "EventDynamic"
   Shell("Project_EventDynamic.exe", AppWinStyle.NormalFocus)
 Case "Contents"
   MsgBox("-- Contents --")
 Case "About"
   Dim oAbout As New frmAbout()
```

```
      oAbout.Show()
    Case Else
      MsgBox("-- Меню не реализовано --")
  End Select
End Sub
```

ВОПРОСЫ САМОКОНТРОЛЯ

Задача «ReadWrite»

1 Покажите процедуру свойства
2 Покажите точку вызова свойства
3 Назовите имя свойства?
4 Из каких блоков состоит процедура свойства?
5 Какой по счету должна быть процедура свойства в классе?
6 Каким словом помечается процедура свойства?

Задача «ReadOnly»

1 Покажите процедуру свойства
2 Покажите точку вызова свойства
3 Назовите имя свойства?
4 Какой блок запрещает ключевое слово ReadOnly?
5 Какой по счету должна быть процедура свойства в классе?
6 Каким словом помечается процедура свойства?
7 Можно ли сделать свойство Read только на 10-й раз?

Задача «WriteOnly»

1 Покажите процедуру свойства
2 Покажите точку вызова свойства
3 Назовите имя свойства?
4 Какой блок запрещает ключевое слово WriteOnly?
5 Какой по счету должна быть процедура свойства в классе?
6 Каким словом помечается процедура свойства?

Задача «WriteOnce»

1 Покажите процедуру свойства
2 Покажите точку вызова свойства
3 Назовите имя свойства?
4 Какой блок запрещает ключевое слово WriteOnly?
5 Какой по счету должна быть процедура свойства в классе?
6 Каким словом помечается процедура свойства?
7 Какой блок запрещает это ключевое слово?
8 Можно ли сделать свойство Write только на 10-й раз?

Задача «WriteOnceReadMany»

1 Покажите процедуру свойства
2 Покажите точку вызова свойства
3 Назовите имя свойства?
4 Из каких блоков состоит процедура свойства?
5 Какой по счету должна быть процедура свойства в классе?
6 Каким словом помечается процедура?
7 Сделайте свойство ПисатьДваждыЧитатьМного

Задача «OverloadsConstructor»

1 Обведите перегруженные конструкторы
2 Обведите точки вызова перегруженных конструкторов
3 Соедините их
4 Что определяет вызов перегруженных конструкторов?
5 А если число аргументов одинаково?
6 Имя перегруженного конструктора? А другого?
7 Какой по счету должна быть процедура конструктора в классе?
8 Каким словом помечается перегруженный конструктор?

Задача «OverloadsMethod»

1 Покажите перегруженные методы
2 Покажите точки вызова перегруженных методов
3 Соедините их
4 Что определяет вызов перегруженных методов?
5 Что определяет вызов перегруженных методов, если число аргументов одинаково?
6 Имя перегруженного метода? А другого?
7 Какой по счету должна быть процедура перегруженного метода в классе?
8 Каким словом помечается перегруженный метод?

Задача «Inherits»

1 Каким словом наследуют классы?
2 На каком месте в классе пишут слово наследования базового класса?
3 Используете ли Вы защищенные члены базового класса?
4 Используете ли MyBase для вызова членов базового класса?
5 Смысл слова Protected?
6 Смысл слова MyBase?
7 Нарисуйте диаграмму объектной модели системы?

Задача «DynamicInheritEvent»

1 Нарисуйте диаграмму объектной модели системы?
2 Нарисуйте диаграмму модели потока событий?
3 Нарисуйте «расположение участников Битвы?»
4 Где объявляется событие?
5 Где назначается обработчик?
6 Где генерируется событие?
8 Где обработчик события?
9 Каким словом событию динамически назначают обработчик?
10 Каким словом уничтожают динамически назначенный обработчик?
11 Если в классе есть RaiseEvent, что еще должно быть в этом классе?
12 Используете ли собственные классы событий?
13 Какой стандартный класс события используется по умолчанию?

ЗАКЛЮЧЕНИЕ

Рассмотренные средства системы VB .NET достаточно эффективно позволяют создавать виртуальные образы элементов реальных систем.

В результате тестирования программных моделей фрагментов систем выявлены особенности работы инструментальных средств.

В работе построены программные модели фрагментов систем, которые могут быть использованы при построении виртуальных образов явлений и сущностей экономических систем.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Список использованных источников может иметь вид:

1 Корнелл Г., Моррисон Дж. Программирование на VB.NET: учебный курс.– СПб.: Питер, 2002.–400 с.:

2 Сайлер, Брайан, Споттс, Джефф Использование Visual Basic.NET. Специальное издание.: Пер. с англ. – М.: Издательский дом «Вильямс», 2002. – 752 с.: ил.

3 Поль Киммел Visual Basic.NET. Искусство программирования. Пер. с англ. – СПб: ООО «ДиасофтЮП», 2003. – 720 с.

4 Франклин, Кит. VB.NET для разработчиков.: Пер. с англ. – М.: Издательский дом «Вильямс», 2002. – 272 с.: ил.