МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ»
АВТОМОБІЛЬНО-ДОРОЖНІЙ ІНСТИТУТ

Кафедра «Інформаційні системи в економіці»

МЕТОДИЧНІ ВКАЗІВКИ ДО ВИКОНАННЯ ПРАКТИЧНИХ РОБІТ З ДИСЦИПЛІНИ «ОБ'ЄКТНЕ ПРОГРАМУВАННЯ» (ДЛЯ СТУДЕНТІВ НАПРЯМУ ПІДГОТОВКИ 6.030502 «ЕКОНОМІЧНА КІБЕРНЕТИКА»)

07/ -2012-04

АДІ ДВНЗ «ДонНТУ»

Горлівка - 2012

МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ»
АВТОМОБІЛЬНО-ДОРОЖНІЙ ІНСТИТУТ

«ЗАТВЕРДЖУЮ»
Директор АДІ ДВНЗ «ДонНТУ»
М. М. Чальцев

Кафедра «Інформаційні системи в економіці»

МЕТОДИЧНІ ВКАЗІВКИ ДО ВИКОНАННЯ ПРАКТИЧНИХ РОБІТ З ДИСЦИПЛІНИ «ОБ'ЄКТНЕ ПРОГРАМУВАННЯ» (ДЛЯ СТУДЕНТІВ НАПРЯМУ ПІДГОТОВКИ 6.030502 «ЕКОНОМІЧНА КІБЕРНЕТИКА»)

07/ -2011-04

«РЕКОМЕНДОВАНО»
Методична комісія факультету
«Економіка та управління»
протокол № від

«РЕКОМЕНДОВАНО»
Кафедра «Інформаційні системи
в економіці»
протокол №3 від 03.11.2011 р.

Горлівка – 2012

УДК

Методичні вказівки до виконання практичних робіт з дисципліни «Об'єктне програмування» (для студентів спеціальностей 6.050100 – Економічна кібернетика) [Електронний курс] / укладач Д.В. Ніколаєнко. – Електрон. дані – Горлівка: ДВНЗ «ДонНТУ» АДІ, 2012. – 1 електрон. опт. диск (CD-R); 12 см. – Систем. вимоги: Pentium; 32 RAM; WINDOWS 98/2000/NT/XP; MS Word 2000/ - Назва з титул. екрану.

Наведені відповідні теоретичні відомості з навчальної дисципліни, сформульовані задачі щодо розв'язування та тестування алгоритмічних задач, вказані вимоги до виконання й оформлення практичних робіт.

Укладачі: Ніколаєнко Д.В., к.т.н., доц.
каф. «Інформаційні системи в економіці»

Відповідальний за випуск: Ніколаєнко В.Л., к.т.н., доц.
каф. «Інформаційні системи в економіці»

Рецензент: Гуменюк М.М., к.е.н., доц.
каф. «Інформаційні системи в економіці»

ЗМІСТ

ВСТУП

Метою виконання практичних робіт є:

- засвоєння засобів об'єктно-орієнтованого програмування – організації властивостей класів, методів, подій, побудови конструкторів класу, перевантаження конструкторів і методів класу;
- освоєння елементарних прийомів об'єктного аналізу систем;
- закріплення навичок роботи в IDE;
- освоєння засобів створення програмних оболонок.

В результаті виконання практичних робіт студент повинен знати:

- засоби об'єктного програмування конструкторів, властивостей, методів і подій класів;
- засоби об'єктного програмування механізму перевантаження конструкторів і методів.

В результаті виконання практичних робіт студент повинен вміти:

- використовувати засоби об'єктного програмування властивостей, методів і подій класів;
- використовувати засоби об'єктного програмування механізму перевантаження конструкторів і методів;
- будувати і обробляти меню програми.

При виконанні індивідуальних завдань студенти отримують перші навики об'єктного програмування – використання властивостей, методів і подій об'єктів.

В процесі виконання завдань студенти освоюють методи налагодження алгоритмів, прийоми візуального проектування програм. Освоюють технології тестування програмних моделей систем.

Отримані навички в подальшому можуть бути використані при побудові програмних моделей економічних систем.

ВКАЗІВКИ ДО ОФОРМЛЕННЯ ТА ВИКОНАННЯ РОБОТИ

Порядок виконання роботи

Крок 1 Підготовка шаблону файлу звіту Звіт_ПрРоб_Ek_XXy_ОП_Прізвище.doc.

Крок 2 Побудова оболонки програмної моделі.

Крок 3 Розв’язання задач контрольної роботи.

Крок 4 Редагування файлу звіту.

Крок 5 Захист контрольної.

ЗАВДАННЯ

Практичні роботи мають проект програми, що містить меню, в якому підключаються виконувані модулі проектів завдань. Таким чином, комплекс практичних робіт – це проект-оболонка.

Крім того, до практичних робіт слід зробити пояснювальну записку, яка має титульний лист, лист завдань і текстову частину – розв’язання задач.

Нижче наведено лист завдань до практичних робіт.

Лист заданий

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

```
File            KontrRab            Help
  Exit          Properties          Contents
                  ReadWrite         About
                  ReadOnly
                  WriteOnly
                  WORM
                Consrtuctor
                  ConsrtuctorOverloads
                Method
                  Method
                  MethodOverloads
                Inherits
                  Inherit
                Events
                  EventDynamic
```

2 Построить пояснительную записку к проекту Отчет_ПрРаб_Ek_XXy_ПО_Фамилия.doc

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

4 Условия задач: получить у преподавателя.

НЕОБХІДНІ ВІДОМОСТІ

Організація властивостей класів

Властивості – це те, що визначає стан об'єкта. Властивості – це поля класу або процедури властивостей.

Відкрите поле – це Public змінна класу. Властивості, побудовані на основі Public змінних класу, не відповідають принципу інкапсуляції.

Закрите поле – це Private змінна класу.

Властивості, що відповідають принципу інкапсуляції будують на основі закритих полів класу і процедур властивостей.

Властивості для читання і запису

Загальний вид властивості для читання і запису:

```
Private ІмʼяЗакритоїЗмінної as ТипВластивості ' Оголошується закрита змінна класу
Public Property ІмʼяВластивості(Параметри) as ТипВлавстивості 'Будується процедура властивості
 get
  Return ІмʼяЗакритоїзмінної
 end get
 set (ByVal арг as ТипВластивості) as ТипВластивості
  ІмʼяЗакритоїЗмінної= арг
 end set
end Property
```

Так організована властивість дозволяє себе встановлювати і читати.

Властивість тільки для читання

Загальний вид властивості тільки для читання:

```
Private ІмʼяЗакритоїЗмінної as ТипВластивості ' Оголошується закрита змінна класу
Public ReadOnly Property ІмʼяВластивості(Параметри) as ТипВлавстивості 'Будується процедура властивості з ключовим словом ReadOnly
 Get
  Return ІмʼяЗакритоїЗмінної
 end get
end Property
```

Властивість тільки для запису

Загальний вид властивості тільки для запису:

```
Private I’мяЗакритоїЗмінної as ТипВластивості
Будується процедура властивості з ключовим словом WriteOnly
Public WriteOnly Property I’мяВластивості (Параметри) as ТипВластивості
   set (ByVal арг as ТипВластивості) as ТипВластивості
    I’мяЗакритоїЗмінної = арг
   end set
end Property
```

Властивість тільки для Одного запису і багаторазового читання

```
Private I’мяЗакритоїЗмінної as ТипВластивості 'Оголошується закрита змінна класу
Private I’мяЗакритоїЛогічноїЗмінної as Boolean = False 'Оголошується закрита мінлива класу
'Будується процедура властивості
Public Property I’мяВластивості (Параметри) as ТипВластивості
   Get
     Return I’мяЗакритоїЗмінної
   end get
   set (ByVal арг as ТипВластивості) as Тип властивості
     if NOT I’мяЗакритоїЛогічноїЗмінної then
      I’мяЗакритоїЗмінної = арг
      I’мяЗакритоїЛогічноїЗмінної = true
     Else
       msgbox ("Властивість вже встановлено!")
     end if
   end set
end Property
```

Задача «ReadWrite»

Умова: створити клас A з ReadWrite властивістю Name String типу, ініціалізація якого виконується значенням NameA при створенні екземпляра класу.

Сценарій тестування:

1. Створюємо об'єкт класу A, передавши конструктору класу значення імені об'єкту NameA.
2. Відображаємо значення властивості Name.
3. Міняємо властивість Name на нове значення.
4. Відображаємо значення властивості Name.

Розв'язок:

Обговорення алгоритму: побудуємо Windows Додаток, головне вікно якого містить кнопки Start і Close.

Сценарій інтерфейсу:

1. Натиснення Close-завершення Додатку.
2. Натиснення Start-виконання Додатку.

Ідея алгоритму.

Властивості об'єктів проекту:

Об'єкт	**Ім'я**	**Напис**
Форма	frmReadWrite	ReadWrite
Кнопка	btnStart	Start
Кнопка	btnClose	Close

Вихідний код алгоритму:

```
Sub Main()
   Dim oA As New A("Юля")
   'Створити клас A з ReadWrite властивістю Name String типу, ініціалізація
якого виконується значенням NameA при створенні екземпляра класу.
   MsgBox("Старе ім'я " & oA.Name)
   oA.Name = InputBox("Введіть нове ім'я", "Вікно введення", "Петя")
   MsgBox("Нове ім'я " & oA.Name)
End Sub

Class A
   Private mName As String
    Public Sub New(ByVal aName As String)
     mName = aName
    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
```

Тестування алгоритму.

Якщо ініціалізація класу проводиться значенням імені Юля, а нове ім'я: Петя. То очікується відповідь: зміна властивості Name класу.

Дійсно:

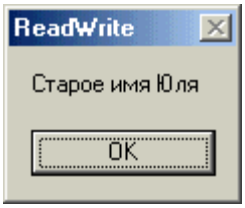

Рисунок 1.1 – ReadWrite(1)

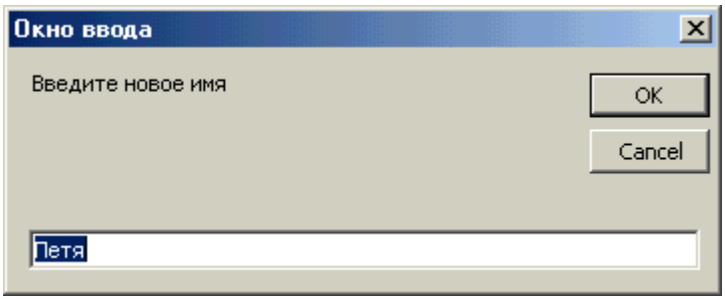

Рисунок 1.2 – ReadWrite(Вікно вводу)

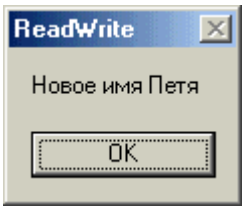

Рисунок 1.3 – ReadWrite(2)

Задача «ReadOnly»

Умова: створити клас A з ReadOnly властивістю Name String типу і ReadOnly властивістю Salary Single типу, ініціалізація яких виконується значеннями NameA і 1000 відповідно при створенні екземпляра класу.

Сценарій тестування:

1. Створюємо об'єкт класу A, передавши конструктору класу значення імені об'єкту NameA і величину зарплати 1000.
2. Відображаємо значення властивості Name і Salary.
3. Міняємо властивості Name і Salary на нові значення
4. Відображаємо значення властивості Name і Salary.

Розв'язок:

Обговорення алгоритму: побудуємо Windows Додаток, головне вікно якого містить кнопки Start і Close.

Сценарій інтерфейсу:

1. Натиснення Close-завершення Додатку.
2. Натиснення Start-виконання Додатку.

Ідея алгоритму

Властивості об'єктів проекта:

Об'єкт	**Ім'я**	**Напис**
Форма	frmReadOnly	ReadOnly
Кнопка	btnStart	Start
Кнопка	btnClose	Close

Вихідний код алгоритму:

```
Sub Main()
 Dim zarplata, newZarplata As Integer
 Dim name, newName As String
 Dim oA As New A("NAMEA", 1000)
 MessageBox.Show("Ім'я співробітника " & oA.Name & " Зарплата співробітника "
& oA.Salary)
 newZarplata = InputBox("Введіть нову зарплату співробітника", "Вікно
введення", "2000")
 newName = InputBox("Введіть нове ім'я співробітника", "Вікно введення",
"NAMEB")
 'oA.Name = newName
 'oA.Salary = newSalary
 MessageBox.Show("Неможливо змінити ім'я і зарплату співробітника т.к вони
ReadOnly ")
End Sub

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

```
    Return mName
  End Get
End Property
Public ReadOnly Property Salary() As Integer
  Get
    Return mSalary
  End Get
End Property
End Class
```

Тестування алгоритму.

Якщо при ініціалізації ім'я співробітника – NameA, а заробітна плата 1000. Внесемо зміни, а саме: задамо ім'я NameB, а заробітну плату 2000, то очікується відповідь: дістанемо відмову в зміні, оскільки ці властивості тільки для читання. Дійсно:

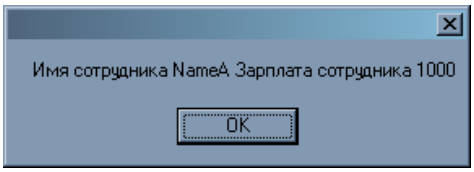

Рисунок 1.4 – ReadOnly(1)

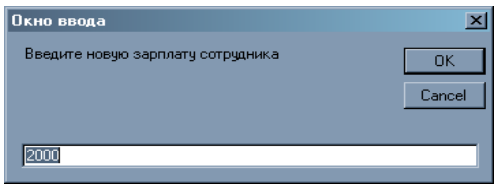

Рисунок 1.5 – ReadOnly(Окно вводу 1)

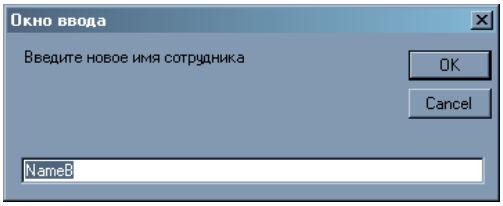

Рисунок 1.6 – ReadOnly(Окно вводу 2)

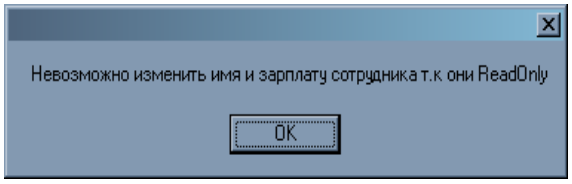

Рисунок 1.7 – ReadOnly(2)

Задача «WriteOnly»

Умова: створити клас A з WriteOnly властивостями Name String типу і WriteOnly властивістю Salary Single типу, ініціалізація яких виконується значеннями NameA і 1000 відповідно при створенні екземпляра класу.

Сценарій тестування:

1. Створюємо об'єкт класу A, передавши конструктору класу значення імені об'єкту NameA і величину зарплати 1000.
2. Відображаємо значення властивості Name і Salary.
3. Міняємо властивості Name і Salary на нові значення.
4. Відображаємо значення властивості Name і Salary.

Розв'язок:

Обговорення алгоритму: побудуємо Windows Додаток, головне вікно якого містить кнопки Start і Close.

Сценарій інтерфейсу:

1. Натиснення Close-завершення Додатку.
2. Натиснення Start-виконання Додатку.

Ідея алгоритму.

Властивості об'єктів проекту:

Об'єкт	**Ім'я**	**Напис**
Форма	frmWriteOnly	WriteOnly
Кнопка	btnStart	Start
Кнопка	btnClose	Close

Вихідний код алгоритму:

```
Sub Main()
 Dim zarplata, newZarplata As Integer
 Dim name, newName As String
 Dim oA As New A("NameA", 1000)
```

```
      MessageBox.Show("Ім'я співробітника " & oA.Name & " Зарплата співробітника "
 & oA.Salary)
      newZarplata = InputBox("Введіть нову зарплату співробітника", "Вікно
 введення", "2000")
      newName = InputBox("Введіть нове ім'я співробітника", "Вікно введення",
 "NAMEB")
       'oA.Name = newName
       'oA.Salary = newSalary
       MessageBox.Show("Неможливо змінити ім'я і зарплату співробітника т.к вони
 ReadOnly ")
     End Sub

     Class A
      Private mSalary As Integer
      Private mName As String
      Public Sub New(ByVal aName As String, ByVal aSalary As Integer)
        mName = aName
        mSalary = aSalary
      End Sub
      Public WriteOnly Property Name() As String
        Set(ByVal Value As String)
      mName = Value
     End Set
End Property
    Public WriteOnly Property Salary() As Integer
     Set(ByVal Value As String)
      mSalary = Value
     End Set
 End Property
End Class
```

Тестування алгоритму.

Якщо: при ініціалізації ім'я співробітника – NameA, а заробітна плата 1000. Внесемо зміни, а саме: задамо ім'я NameB, а заробітну плату 2000, то очікується відповідь: властивості приймуть нові значення, але відобразити їх на екрані буде неможливо, оскільки вони тільки для запису. Дійсно: немає вікон.

Задача «WriteOnceReadMany»

Умова: створити клас А з ReadWrite властивістю Name String типу і WriteOnceReadMany властивістю Salary Single типу, ініціалізація яких виконується значеннями NameA і 1000 відповідно при створенні екземпляра класу.

Сценарій тестування:

1. Створюємо об'єкт класу А, передавши конструктору класу значення імені об'єкту NameA і величину зарплати 1000.

2. Відображаємо значення властивості Name і Salary.

3. Міняємо властивості Name і Salary на нові значення.
4. Відображаємо значення властивості Name і Salary.
5. Міняємо властивість Salary на нове значення.
6. Відображаємо значення властивості Salary.

Розв'язок:

Обговорення алгоритму: побудуємо Windows Додаток, головне вікно якого містить кнопки Start і Close.

Сценарій інтерфейсу:

1. Натиснення Close-завершення Додатку.
2. Натиснення Start-виконання Додатку.

Ідея алгоритму

Властивості об'єктів проекту:

Об'єкт	**Ім'я**	**Напис**
Форма	frmWORM	WORM
Кнопка	btnStart	Start
Кнопка	btnClose	Close

Вихідний код алгоритму:

```
Sub Main()
 Dim oA As New A("NameA", 1000)
 MsgBox("Початкове ім'я" & oA.Name & "Початкова зарплата" & oA.Salary)
   oA.Name = InputBox("Введіть нове ім'я", "Вікно введення" "   ")
   oA.Salary = InputBox("Введіть нову зарплату", "Вікно введення" "   ")
 MsgBox("Нове ім'я" & oA.Name & "Нова зарплата" & oA.Salary)
   oA.Salary = InputBox("Зміните зарплату", "Вікно введення" "   ")
 MsgBox("Змінена зарплата" & oA.Salary)
End Sub

Class A
 Private mName As String, mSalary As Decimal
 Private 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 установлена ранее!!!")
    End If
  End Set
 End Property
End Class
```

Тестування алгоритму.

Якщо: ініціалізація відбувається значеннями NameA і 1000, потім відбувається перепризначення значеннями NameB і 2000 відповідно, потім повторно змінити заробітну плату, то очікується відповідь: Заробітна плата встановиться 1000, а потім зміниться. При другій спробі змінити заробітну плату дістанемо відмову, оскільки властивість Salary організовано як WriteOnce, n/t його можна змінити одного разу. Дійсно:

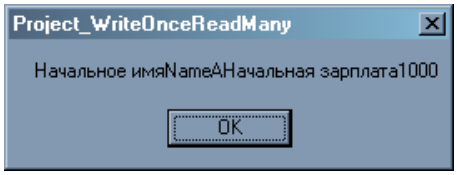

Рисунок 1.8 – WriteOnceReadMany(1)

Рисунок 1.9 – WriteOnceReadMany(вікно вводу 1)

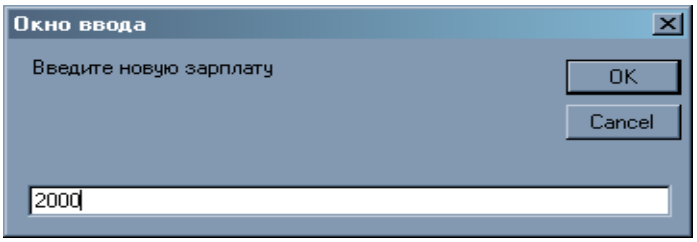

Рисунок 1.10 – WriteOnceReadMany(вікно вводу 2)

Рисунок 1.11 – WriteOnceReadMany(2)

Рисунок 1.12 – WriteOnceReadMany(вікно вводу 3)

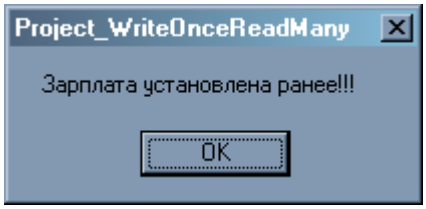

Рисунок 1.13 – WriteOnceReadMany(3)

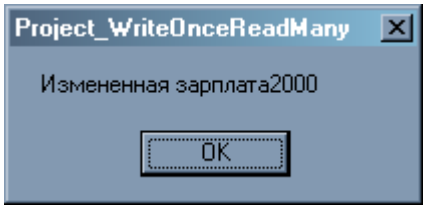

Рисунок 1.14 – WriteOnceReadMany(4)

Задача «ConstructorOverload»

Умова: Створити клас A з ReadOnly властивістю Name String типу і ReadOnly властивістю Salary Single типу, ініціалізація яких виконується значеннями NameA і 1000 відповідно при створенні екземпляра класу. Додати властивість Password String типа, закрите поле, якого ініціалізувався при оголошенні значенням «пароль». Перенавантажувати конструктор, який в першому випадку ініціалізував властивості Name і Salary значеннями NameA і 1000 відповідно, а в іншому випадку - властивості Name, Salary і властивість Password значенням пароля.

Сценарій тестування:

1. Створюємо об'єкт класу A, передавши конструктору класу значення імені об'єкту NameA і величину зарплати 1000.

2. Створюємо інший об'єкт класу A, передавши конструктору класу значення імені об'єкту NameA, величину зарплати 1000 і пароль.

3. Відображаємо значення властивості Name, Salary і пароль першого об'єкту.

4. Відображаємо значення властивості Name, Salary і пароль другого об'єкту.

Розв'язок:

Обговорення алгоритму: побудуємо Windows Додаток, головне вікно якого містить кнопки Start і Close.

Сценарій інтерфейсу:

1. Натиснення Close-завершення Додатку.

2. Натиснення Start-виконання Додатку.

Ідея алгоритму.

Властивості об'єктів проекту:

Об'єкт	**Ім'я**	**Напис**
Форма	frmConstructorOverload	ConstructorOverload
Кнопка	btnStart	Start
Кнопка	btnClose	Close

Вихідний код алгоритму:

```
   Sub Main()
       Dim Password As String
        Password = InputBox("Введіть пароль другого співробітника ", "Вікно
введення" " ")
       Dim oA As New A("Aname", 900)
       Dim bA As New A("Bname", 1000, Password)
        MsgBox("Ім'я першого співробітника " & oA.Name & " Зарплата першого
співробітника " & oA.Salary & "Пароль = " & oA.Password)
        MsgBox("Ім'я другого співробітника " & bA.Name & " Зарплата другого
співробітника " & bA.Salary & "Пароль = " & bA.Password)
```

```
        End Sub

         Class A
            Private mName As String
            Private mSalary As Single
            Private mPassword As String
             'Перевантаження конструкторів.
             'Конструктор для першого співробітника
            Public Sub New(ByVal aName As String, ByVal aSalary As Single)
                 mName = aName
                 mSalary = aSalary
                 mPassword = "Пароль 1"
            End Sub
             'Конструктор для другого співробітника
            Public Sub New(ByVal aName As String, ByVal aSalary As Single, ByVal
aPassword As String)
                 mName = aName
                 mSalary = aSalary
                 mPassword = aPassword
            End Sub
             'Властивість Name
            Public ReadOnly Property Name() As String
                 Get
                     Return mName
                End Get
             End Property
             ' Властивість Salary
            Public ReadOnly Property Salary() As Single
                 Get
                     Return mSalary
                End Get
             End Property
             Public aPassword As String
             'Властивість пароль
            Public Property Password() As String
                 Get
                     Return mPassword
                End Get
                 Set(ByVal Value As String)
                     mPassword = Value
                End Set
             End Property
         End Class
```

Тестування алгоритму.

Якщо створити об'єкт від класу A, передавши конструктору класу значення імені об'єкту NameA і величину зарплати 1000, створити інший об'єкт від класу A, передавши конструктору класу значення імені об'єкту NameA, величину зарплати 1000 і пароль, відобразити значення властивості Name, Salary і пароль першого об'єкту, відобразити значення властивості Name, Salary і пароль другого об'єкту, то очікується відповідь: ініціалізація першого і другого об'єкту пройде успішно, закриті поля будуть проініціалізовані, для першого об'єкту повинні будуть відобразитися значення його закритих полів встановленими значеннями, а пароль повинен

бути порожній, для другого об'єкту значення пароля повинне співпадати із значенням, яким воно ініціалізувалося. Дійсно:

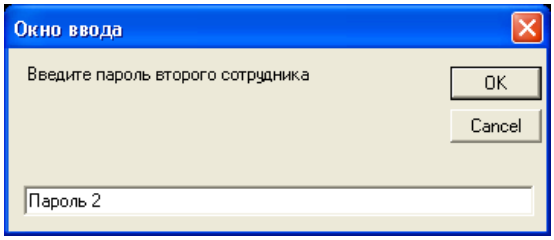

Рисунок 1.15 – ConstructorOverload (Вікно вводу)

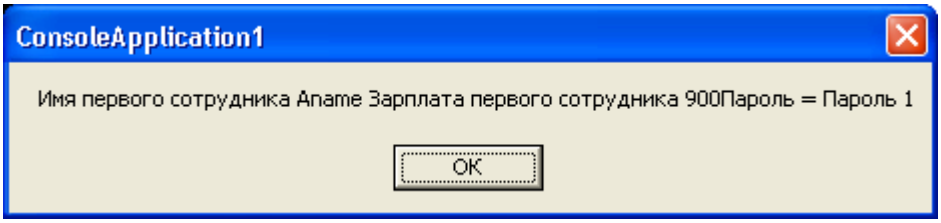

Рисунок 1.16 – ConstructorOverload (1)

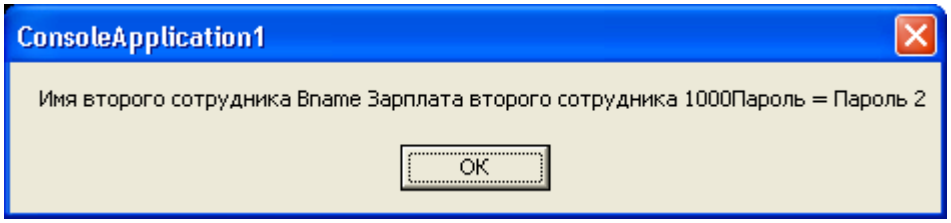

Рисунок 1.17 – ConstructorOverload (2)

Задача «Metod»

Умова: Створити клас A з ReadOnly властивістю Name String типу і ReadOnly властивістю Salary Single типу, ініціалізація яких виконується значеннями NameA і 1000 відповідно при створенні екземпляра класу.

Додати до класу метод RaiseSalary, який по заданому числу відсотків збільшує зарплату на це число відсотків.

Сценарій тестування:

1. Створюємо об'єкт класу A, передавши конструктору класу значення імені об'єкту NameA величину зарплати 1000.

2. Відображаємо значення властивості Name і Salary

3. Викликаємо метод RaiseSalary і задаємо нове значення властивості Salary

4. Відображаємо значення властивості Name і Salary

Розв'язок:

Обговорення алгоритму: побудуємо Windows Додаток, головне вікно якого містить кнопки Start і Close.

Сценарій інтерфейсу:

1. Натиснення Close-завершення Додатку.

2. Натиснення Start-виконання Додатку.

Ідея алгоритму.

Властивості об'єктів проекту:

Об'єкт	**Ім'я**	**Напис**
Форма	FrmMetod	Metod
Кнопка	btnStart	Start
Кнопка	btnClose	Close

Вихідний код алгоритму:

```
Sub Main()
 Dim percent As Single
 Dim oA As New A("NAMEA", 1000)
 MessageBox.Show("Ім'я співробітника " & oA.Name & " Зарплата співробітника " & oA.Salary)
 percent = InputBox("Введіть відсоток підвищення зарплати співробітника", "Вікно введення ", "20")
 oA.RaiseSalary(percent)
 MessageBox.Show("Ім'я співробітника " & oA.Name & " Нова зарплата співробітника " & oA.Salary)
End Sub

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

```
    mSalary = mSalary + 0.01 * aPercent * mSalary
  End Sub
End Class
```

Тестування алгоритму.

Якщо створити об'єкт класу A, передавши конструктору класу значення імені об'єкту NameA величину зарплати 1000, спробувати відобразити значення властивості Name і Salary, а потім викликати метод RaiseSalary і підвищити заробітну плату на нове значення (20 відсотків) і знов відобразити значення властивості Name і Salary, то очікується відповідь: ім'я співробітника NameA, заробітна плата 1200. Дійсно:

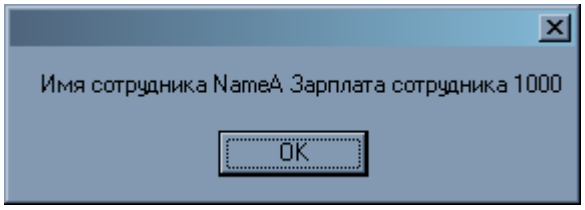

Рисунок 1.18 – Metod (1)

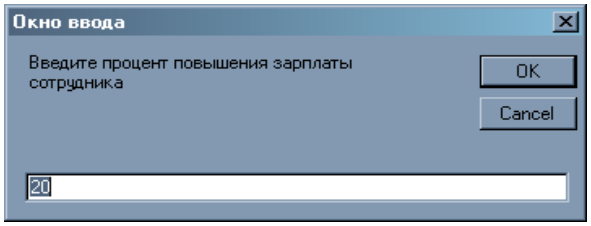

Рисунок 1.19 – Metod (2)

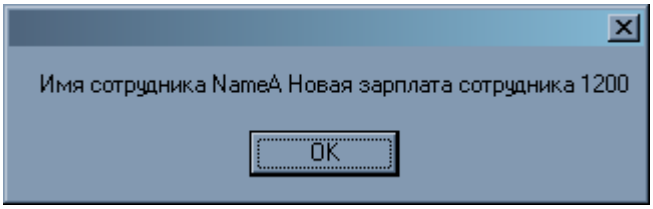

Рисунок 1.20 – Metod (3)

Задача «MetodOverload»

Умова: Створити клас A з ReadOnly властивістю Name String типу і ReadOnly властивістю Salary Single типу, ініціалізація яких виконується значеннями NameA і 1000 відповідно при створенні екземпляра класу.

Додати до класу закрите поле Password String типу, ініціалізація якого проводиться за умовчанням значенням номер варіанту.

Додати до класу метод RaiseSalary, який по заданому числу відсотків збільшує зарплату на це число відсотків, а якщо підвищення зарплати перевищує 10%, то запрошує при цьому ще і пароль.

Перенавантажувати метод RaiseSalary так, щоб він по заданому числу відсотків і заданому паролю, збільшував зарплату і менш і більш ніж на 10%.

Сценарій тестування:

1. Створюємо об'єкт класу A, передавши конструктору класу значення імені об'єкту NameA і величину зарплати 1000.
2. Відображаємо значення властивості Name, Salary і пароль об'єкту
3. Викликаємо метод RaiseSalary з рівнем підвищення зарплати менше 10%
4. Відображаємо значення властивості Name і Salary
5. Викликаємо метод RaiseSalary з рівнем підвищення зарплати менше 20%
6. При запиті пароля ввести правильне значення паорля.
7. Відображаємо значення властивості Name і Salary
8. Викликаємо метод RaiseSalary з рівнем підвищення зарплати менше 20%
9. При запиті пароля ввести неправильне значення паорля.
10. Відображаємо значення властивості Name і Salary
11. Викликаємо для першого об'єкту метод RaiseSalary з рівнем підвищення зарплати менше 10% і вказівкою правильного пароля.
12. Відображаємо значення властивості Name і Salary
13. Викликаємо для другого об'єкту метод RaiseSalary з рівнем підвищення зарплати менше 10% і вказівкою неправильного пароля.
14. Відображаємо значення властивості Name і Salary
15. Викликаємо для першого об'єкту метод RaiseSalary з рівнем підвищення зарплати більше 10% і вказівкою правильного пароля.
16. Відображаємо значення властивості Name і Salary
17. Викликаємо для другого об'єкту метод RaiseSalary з рівнем підвищення зарплати більше 10% і вказівкою неправильного пароля.

18. Відображаємо значення властивості Name і Salary

Розв'язок:

Обговорення алгоритму: побудуємо Windows Додаток, головне вікно якого містить кнопки Start і Close.

Сценарій інтерфейсу:

1. Натиснення Close-завершення Додатку.
2. Натиснення Start-виконання Додатку.

Ідея алгоритму.

Властивості об'єктів проекту:

Об'єкт	**Ім'я**	**Напис**
Форма	frmMetodOverload	MetodOverload
Кнопка	btnStart	Start
Кнопка	btnClose	Close

Вихідний код алгоритму:

```
   Sub Main()
        ' Оголошуємо змінні
       Dim percent As Single           ' відсоток підвищення зарплати
       Dim oA As New A("NAMEA", 1000)  ' екземпляр від класу A
        ' протестуємо метод підвищення заробітної плати на величину менше 10
відсотків без передачі пароля
       MsgBox("Ім'я співробітника " & oA.Name & Chr(13)& " Зарплата
співробітника " & oA.Salary)
        percent = InputBox("Введіть відсоток підвищення зарплати
співробітника", "Вікно введення ", "10")
        oA.RaiseSalary(percent)
        ' протестуємо метод підвищення заробітної плати на величину більше 10
відсотків без передачі пароля
       MsgBox("Ім'я співробітника " & oA.Name & Chr(13)& " Нова зарплата
співробітника " & oA.Salary)
        percent = InputBox("Введіть відсоток підвищення зарплати
співробітника", "Вікно введення ", "20")
        oA.RaiseSalary(percent)
        ' протестуємо метод підвищення заробітної плати на величину менше 10
відсотків без передачі пароля
       MsgBox("Ім'я співробітника " & oA.Name & Chr(13)& " Нова зарплата
співробітника " & oA.Salary)
        percent = InputBox("Введіть відсоток підвищення зарплати
співробітника", "Вікно введення ", "20")
        oA.RaiseSalary(percent)
       MsgBox("Ім'я співробітника " & oA.Name & Chr(13)& " Нова зарплата
співробітника " & oA.Salary)
        '
       MsgBox("Ім'я співробітника " & oA.Name & Chr(13)& " Нова зарплата
співробітника " & oA.Salary)
        percent = InputBox("Введіть відсоток підвищення зарплати
співробітника", "Вікно введення ", "20")
        oA.RaiseSalary(percent)
       MsgBox("Ім'я співробітника " & oA.Name & Chr(13)& " Нова зарплата
співробітника " & oA.Salary)
        '
       MsgBox("Ім'я співробітника " & oA.Name & Chr(13)& " Нова зарплата
співробітника " & oA.Salary)
```

```
            percent    =    InputBox("Введіть    відсоток    підвищення    зарплати
співробітника", "Вікно введення ", "20")
            oA.RaiseSalary(percent)
           MsgBox("Ім'я співробітника " & oA.Name & Chr(13)& " Нова зарплата
співробітника " & oA.Salary)
        End Sub
    Class A
        Private mSalary As Integer
        Private mName As String
        Private Password As String
        Public Sub New(ByVal aName As String, ByVal aSalary As Integer)
            mName = aName
            mSalary = aSalary
            Password = "1"
        End Sub
        Public ReadOnly Property Name() As String
            Get
                Return mName
            End Get
        End Property
        Public ReadOnly Property Salary() As Single
            Get
                Return mSalary
            End Get
        End Property
        Public Sub RaiseSalary(ByVal aPercent As Single)
            If aPercent <= 10 Then
                mSalary = mSalary + 0.01 * aPercent * mSalary
            Else
                If Password = InputBox("Введіть пароль для підвищення заробітної
плати", "Введення пароля", "1") Then
                    mSalary = mSalary + 0.01 * aPercent * mSalary
                Else
                    MsgBox("Пароль  не  вірний,  підвищення  заробітної  плати  не
відбулося")
                End If
            End If
        End Sub
       Public Sub RaiseSalary(ByVal aPercent As Single, ByVal pwd As String)
           If aPercent <= 10 Then
                mSalary = mSalary + 0.01 * aPercent * mSalary
           Else
               If Password = pwd Then
                    mSalary = mSalary + 0.01 * aPercent * mSalary
               Else
                   MsgBox("У переобтяжений метод пароль переданий не вірний,
підвищення заробітної плати не відбулося")           End If
            End If
        End Sub
    End Class
```

Тестування алгоритму.

Якщо створити об'єкт класу A, передавши конструктору класу значення імені об'єкту NameA і величину зарплати 1000, спробувати відображеннєм значення властивості Name, Salary і паролем об'єкту, викликати метод RaiseSalary з рівнем підвищення зарплати менше 10%, відобразити значення властивості Name і Salary, викликати метод RaiseSalary з рівнем підвищення зарплати менше 20%, при запиті пароля ввести правильне значення пароля,

відобразити значення властивості Name і Salary, викликати метод RaiseSalary з рівнем підвищення зарплати менше 20%, при запиті пароля ввести неправильне значення пароля, відобразити значення властивості Name і Salary, викликати для першого об'єкту метод RaiseSalary з рівнем підвищення зарплати менше 10% і вказівкою правильного пароля, відобразити значення властивості Name і Salary, викликати для другого об'єкту метод RaiseSalary з рівнем підвищення зарплати менше 10% і вказівкою неправильного пароля, відобразити значення властивості Name і Salary, викликати для першого об'єкту метод RaiseSalary з рівнем підвищення зарплати більше 10% і вказівкою правильного пароля, відобразити значення властивості Name і Salary, викликати для другого об'єкту метод RaiseSalary з рівнем підвищення зарплати більше 10% і вказівкою неправильного пароля, відобразити значення властивості Name і Salary, то очікується відповідь:

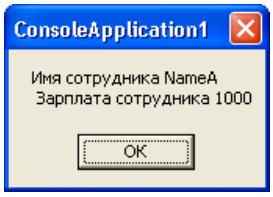

Рисунок 1.21 – MetodOverload (1)

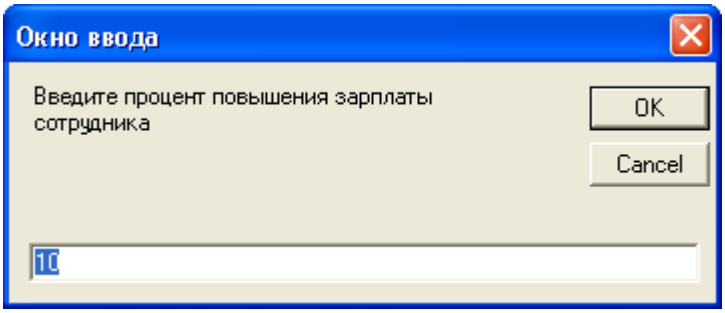

Рисунок 1.22 – MetodOverload (2)

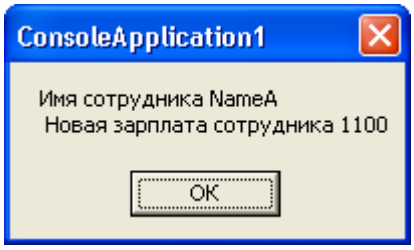

Рисунок 1.23 – MetodOverload (3)

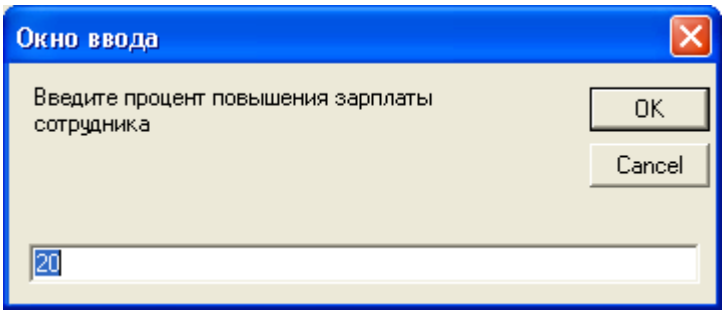

Рисунок 1.24 – MetodOverload (4)

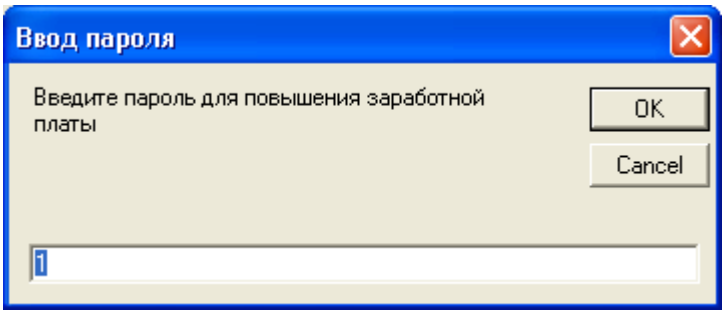

Рисунок 1.25– MetodOverload (5)

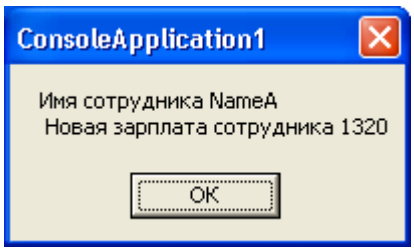

Рисунок 1.26 – MetodOverload (6)

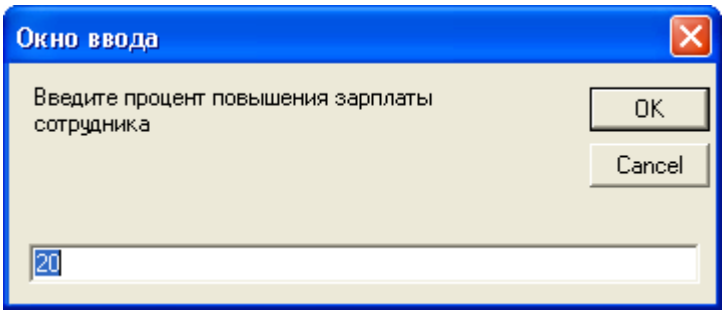

Рисунок 1.27 – MetodOverload (7)

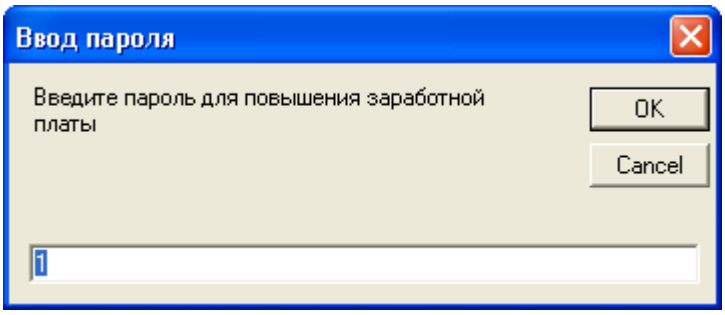

Рисунок 1.28 – MetodOverload (8)

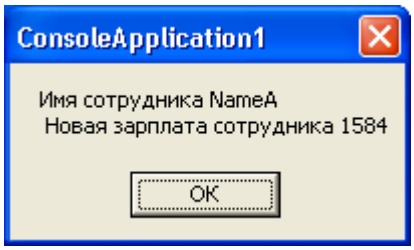

Рисунок 1.29 – MetodOverload (9)

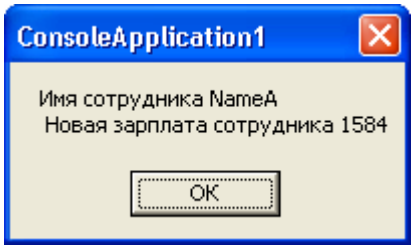

Рисунок 1.30 – MetodOverload (10)

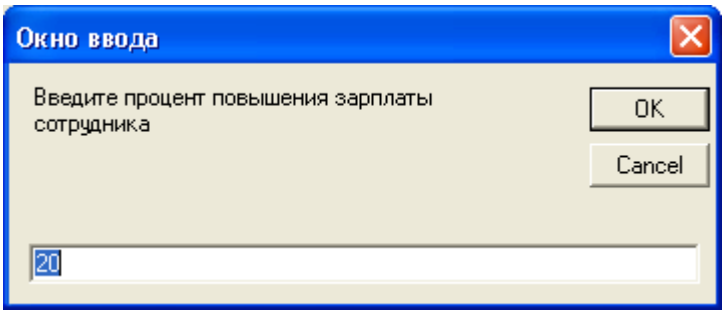

Рисунок 1.31 – MetodOverload (11)

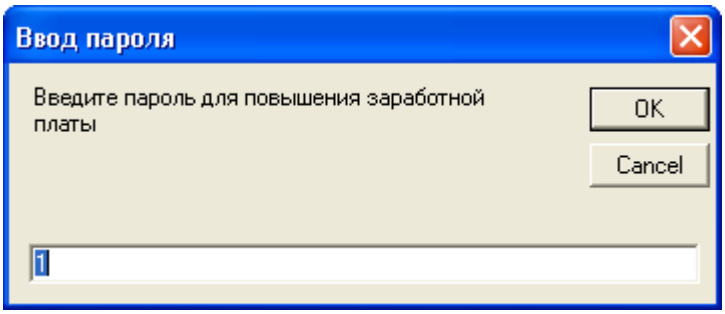

Рисунок 1.32 – MetodOverload (12)

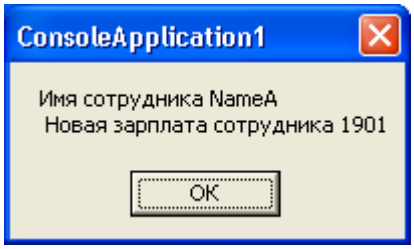

Рисунок 1.33 – MetodOverload (13)

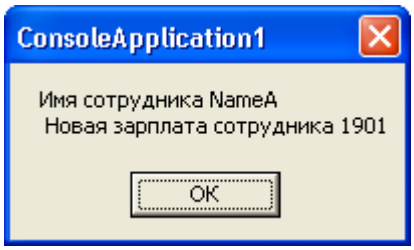

Рисунок 1.34 – MetodOverload (14)

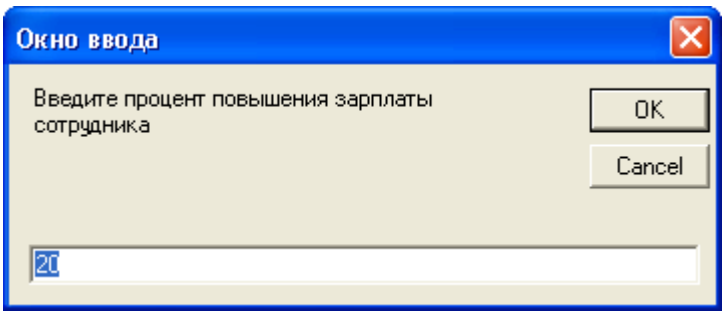

Рисунок 1.35 – MetodOverload (15)

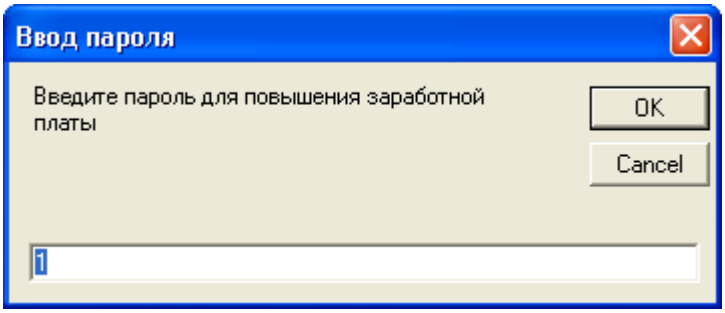

Рисунок 1.36 – MetodOverload (16)

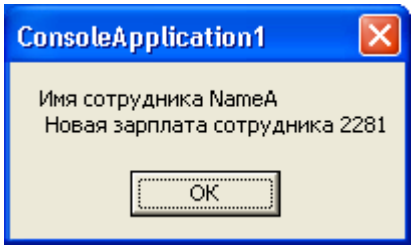

Рисунок 1.37 – MetodOverload (17)

Задача «Inherits»

Умова: дана ОМ деякої ЕС:

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

Нехай кожен клас об'єктної моделі має деякий Public Main метод – MainИмяКласса.

Побудувати класи для об'єктів X, A, B, C, D, E.

Сценарій тестування:

Тестування слова Protected.

1. Нехай клас верхнього рівня ієрархії має закрите Protected ReadWrite String властивість ProtectedPropertyІмяКласу, значення якої ініціалізувалося при створенні екземпляра класу величиною «ProtectedReadWritePropertyІмяКласу».

У Main методі похідного класу самого нижнього рівня ієрархії:

1) відобразити значення властивості захищеного в класі самого верхнього рівня ієрархії;

2) встановити нове значення цієї властивості величиною «ProtectedReadWritePropertyІмяПохідногоКласу»;

3) відобразити нове значення цієї властивості;

4) у Main методі класу, що знаходиться поза ієрархією, відобразити значення цієї властивості.

2. Хай клас верхнього рівня ієрархії має закрите ReadWrite String властивість ReadWritePropertyИмяКласса, значення якої ініціалізувалося при створенні екземпляра класу величиною «ReadWritePropertyИмяКласса».

1) у Main методі класу, що знаходиться поза ієрархією, відобразити значення цієї властивості;

2) встановити нове значення цієї властивості величиною «ReadWritePropertyІмяКласуПозаІєрархією»;

3) відобразити нове значення цієї властивості.

Тестування слова MyBase

3. Хай базовий клас для класу самого нижнього рівня ієрархії має метод Q, що приймає аргумент цілого типу і що повертає його подвоєну величину.

1) у похідному класі перевизначити цей метод так щоб він повертав потрійну величину отриманого аргументу;

2) викликати перевизначений метод, а також викликати відповідний метод базового класу, що перевизначиться.

Розв'язок

Обговорення алгоритму: побудуємо Windows Додаток, головне вікно якого містить кнопки Start і Close.

Сценарій інтерфейсу:

1. Натиснення Close-завершення Додатку.

2. Натиснення Start-виконання Додатку.

Ідея алгоритму

Властивості об'єктів проекту:

Об'єкт	**Ім'я**	**Напис**
Форма	frmInherits	Inherits
Кнопка	btnStart	Start
Кнопка	btnClose	Close

Вихідний код алгоритму:

```
Sub Main()
  Dim oSystemX As New SystemX
  oSystemX.MainSystemX()
  Dim oA As New A("NameA")
  oA.MainA()
  Dim oB As New B("NameB")
  oB.MainB()
  oB.Q(2)
  Dim oC As New C("NameC")
  oC.MainC()
```

```
    Dim oD As New D("NameD")
    oD.MainD()
    Dim oE As New E("NameE")
    oE.MainE()
  End Sub

  Class SystemX
    Public Sub MainSystemX()
      MessageBox.Show("Це Main метод класу SystemX ")
    End Sub
  End Class

  Class A
    Inherits SystemX
    Private mName As String
    Public Sub New(ByVal aName As String)
      mName = aName
    End Sub
    Public Property Name() As String
      Get
        Return mName
      End Get
      Set(ByVal Value As String)
        mName = Value
      End Set
    End Property
   Public Sub MAINA()
     MessageBox.Show("Це Main метод класу A ")
     Dim oB As New B("NameB")
     'MessageBox("Закрита властивість класу B:" & oB.ReadWritePropertyB)
     MessageBox.Show("Неможливо      відобразити      закриту      властивість
ReadWritePropertyB класу ")
      'oB.ReadWritePropertyB="Name"
     MessageBox.Show("Неможливо   поміняти   значення   закритої   властивості
ReadWritePropertyB класу ")
   End Sub
  End Class

  Class B
    Inherits SystemX
    Public Sub New(ByVal aName As String)
      mName = aName
    End Sub
    Dim mName As String
    Protected Property ProtectedPropertyB() As String
      Get
        Return mName
      End Get
      Set(ByVal Value As String)
        mName = Value
      End Set
    End Property
    Private Property ReadWritePropertyB() As String
      Get
        Return mName
      End Get
      Set(ByVal Value As String)
        mName = Value
      End Set
    End Property
    Public Sub MainB()
      MessageBox.Show("Це Main метод класу B ")
```

```
        End Sub
        Public Overridable Sub Q(ByVal arg As Integer)
          MessageBox.Show("Це Overridable процедура класу B")
          MessageBox.Show("Подвоєне значення " & 2 * arg)
        End Sub
      End Class

      Class C
       Inherits B
       Public Sub New(ByVal aName As String)
         MyBase.New(aName)
       End Sub
       Public Sub MAINC()
         MessageBox.Show("Це Main метод класу З ")
         MessageBox.Show("Ім'я властивості клас B " & ProtectedPropertyB, "",
MessageBoxButtons.OK, MessageBoxIcon.Information)
          ProtectedPropertyB = "ProtectedReadWritePropertyC"
         MessageBox.Show("Нове ім'я властивості клас B " & ProtectedPropertyB,
"Повідомлення", MessageBoxButtons.OK, MessageBoxIcon.Information)
         MyClass.Q(2)
         MyBase.Q(2)
       End Sub
       Public Overrides Sub Q(ByVal arg As Integer)
         MessageBox.Show("Це Overrides процедура класу C ", "Повідомлення",
MessageBoxButtons.OK, MessageBoxIcon.Information)
         MessageBox.Show("Потрійне значення аргументу рівне " & 3 * arg,
"Повідомлення", MessageBoxButtons.OK, MessageBoxIcon.Information)
       End Sub
      End Class

      Class D
        Inherits SystemX
        Private mName As String
        Public Sub New(ByVal aName As String)
          mName = aName
        End Sub
        Public Property Name() As String
          Get
            Return mName
          End Get
          Set(ByVal Value As String)
            mName = Value
          End Set
        End Property
        Public Sub MainD()
          MessageBox.Show("Це Main метод класу D ")
        End Sub
      End Class

      Class E
        Inherits D
        Private mName As String
        Public Sub New(ByVal aName As String)
          MyBase.New(aName)
        End Sub
        Public Sub MainE()
          MessageBox.Show("Це Main метод класу E ", "Повідомлення",
MessageBoxButtons.OK, MessageBoxIcon.Information)
        End Sub
      End Class
```

Тестування алгоритму.

Якщо звернутися до Main методам об'єктів (класів), то отримаємо повідомлення про виклик Main методів, а при зверненні (передачі числа 2 в метод) до методу Q базового класу набудемо подвоєного значення - 4. При зверненні до методу Q поточного класу набудемо потрійного значення переданого аргументу. То очікується відповідь: MsgBox Main методів і результат функції Q. Дійсно:

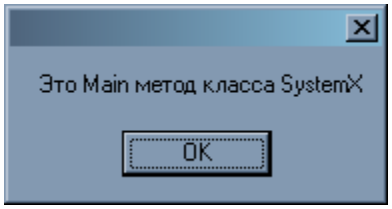

Рисунок 1.38 – Inherits (1)

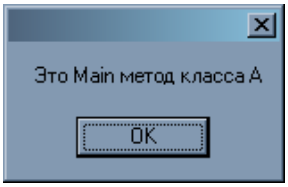

Рисунок 1.39 – Inherits (2)

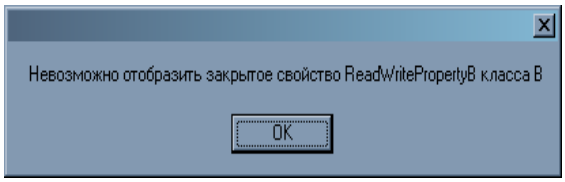

Рисунок 1.40 – Inherits (3)

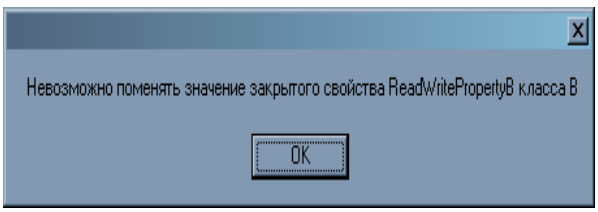

Рисунок 1.41 – Inherits (4)

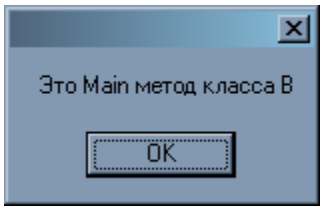

Рисунок 1.42 – Inherits (5)

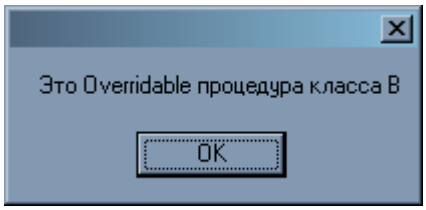

Рисунок 1.43 – Inherits (6)

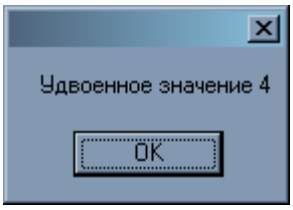

Рисунок 1.44 – Inherits (7)

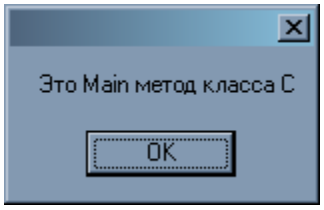

Рисунок 1.45 – Inherits (8)

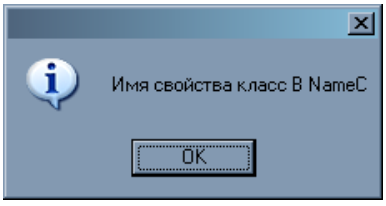

Рисунок 1.46 – Inherits (9)

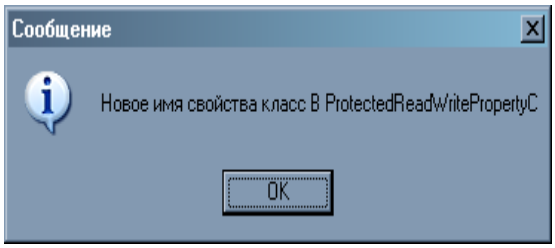

Рисунок 1.47 – Inherits (10)

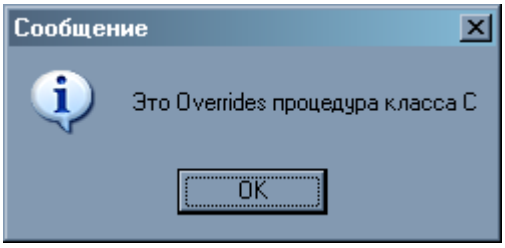

Рисунок 1.48 – Inherits (11)

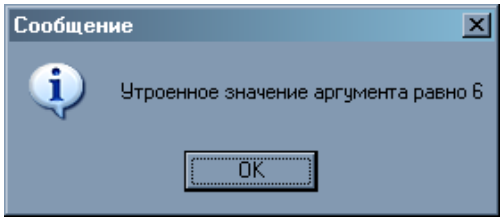

Рисунок 1.49 – Inherits (12)

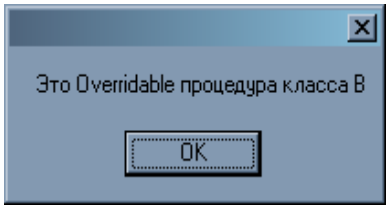

Рисунок 1.50 – Inherits (13)

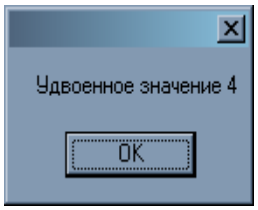

Рисунок 1.51 – Inherits (14)

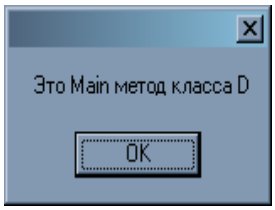

Рисунок 1.52 – Inherits (15)

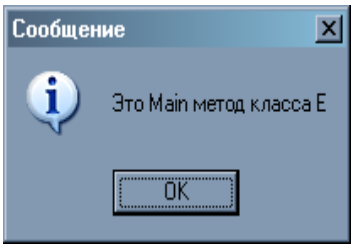

Рисунок 1.53 – Inherits (16)

Задача «EventDynamic»

Умова. Дана об'єктна модель і потік подій деякої економічної системи. Побудувати програмну модель об'єктної моделі економічної системи.

```
X           6  7  8  9 10 11 12 13 14
|           ------------------------> Час
|_ A        A     B     C     D     E
|_ B              ------------>
|  |_ C                 ------>
|_ D        <-----------------------
   |_ E     <-----
```

Сценарій тестування

Розв'язок

Обговорення алгоритму: побудуємо Windows Додаток, головне вікно якого містить кнопки Start і Close.

Сценарій інтерфейсу:

1. Натиснення Close-завершення Додатку.
2. Натиснення Start-виконання Додатку.

Ідея алгоритму

Властивості об'єктів проекту:

Об'єкт	**Ім'я**	**Напис**
Форма	frmEventDynamic	EventDynamic
Кнопка	btnStart	Start
Кнопка	btnClose	Close

Вихідний код алгоритму:

```
Sub Main()
  Dim oA As New A()
  Dim oB As New B()
  Dim oC As New C()
  Dim oD As New D()
  Dim oE As New E()
  Dim Day As New Date(2006, 3, 5)
  Dim i As Integer, myDay As String
  For i = 1 To 11
    myDay = Day.ToShortDateString
     MessageBox.Show("День " & myDay & " почався", "Повідомлення",
MessageBoxButtons.OK, MessageBoxIcon.Information)
    oB.MainB(myDay)
    oC.MainC(myDay)
    oE.MainE(myDay)
    MessageBox.Show("День " & myDay & " закінчився", "Повідомлення",
MessageBoxButtons.OK, MessageBoxIcon.Information)
    Day = Day.AddDays(1)
  Next i
End Sub

Module myModule
  Dim oA As New A()
  Dim oB As New B()
  Dim oC As New C()
  Dim oD As New D
  Dim oE As New E
```

```
      Class SystemX
      End Class

      Class A
       Inherits SYSTEMX
       Public Sub revFromBforA()
         MessageBox.Show("Обробка   події   від   B   для   A",   "Повідомлення",
MessageBoxButtons.OK, MessageBoxIcon.Information)
       End Sub
       Public Sub revFromEforA()
         MessageBox.Show("Обробка   події   від   E   для   A",   "Повідомлення",
MessageBoxButtons.OK, MessageBoxIcon.Information)
       End Sub
      End Class

      Class B
        Inherits SystemX
        Public Event ev1FromBforA()
        Public Event ev1FromBforD()
        Public Sub MainB(ByVal myDay As String)
          Static RunEv1FromBforA As Boolean
          If "08.03.2006" = myDay Then
            If Not RunEv1FromBforA Then
              AddHandler Me.ev1FromBforA, AddressOf oA.revFromBforA
              RaiseEvent ev1FromBforA()
              RemoveHandler Me.ev1FromBforA, AddressOf oA.revFromBforA
              RunEv1FromBforA = True
            End If
          End If
          Static RunEv1FromBforD As Boolean
          If myDay >= "08.03.2006" And myDay <= "12.03.2006" Then
            If Not RunEv1FromBforD Then
              AddHandler Me.ev1FromBforD, AddressOf oC.revFromBforD
              RaiseEvent ev1FromBforD()
              RemoveHandler Me.ev1FromBforD, AddressOf oC.revFromBforD
              RunEv1FromBforD = True
            End If
          End If
        End Sub
      ublic Sub revFromBforA()
         MessageBox.Show("Обробка   події   від   B   для   A",   "Повідомлення",
MessageBoxButtons.OK, MessageBoxIcon.Information)
       End Sub
       Public Sub revFromBforD()
         MessageBox.Show("Обробка   події   від   B   для   D",   "Повідомлення",
MessageBoxButtons.OK, MessageBoxIcon.Information)
        End Sub
      End Class

      Class C
        Inherits B
        Public Event ev1FromCforD()
        Public Sub MainC(ByVal myDay As String)
          Static RunEv1FromCforD As Boolean
          If myDay >= "10.03.2006" And myDay <= "12.03.2006" Then
            If Not RunEv1FromCforD Then
              AddHandler Me.ev1FromCforD, AddressOf oC.revFromCforD
              RaiseEvent ev1FromCforD()
              RemoveHandler Me.ev1FromCforD, AddressOf oC.revFromCforD
              RunEv1FromCforD = True
            End If
```

```
          End If
        End Sub
        Public Sub revFromCforD()
         MessageBox.Show("Обробка    події    від    C    для    D",    "Повідомлення",
MessageBoxButtons.OK, MessageBoxIcon.Information)
       End Sub
      End Class
      Class D
       Inherits SYSTEMX
       Public Sub revFromCforD()
         MessageBox.Show("Обробка    події    від    C    для    D",    "Повідомлення",
MessageBoxButtons.OK, MessageBoxIcon.Information)
       End Sub
      End Class

      Class E
        Inherits D
        Public Event ev1FromEforA()
        Public Sub MainE(ByVal myDay As String)
          Static RunEv1FromEforA As Boolean
          If "14.03.2006" = myDay Then
            If Not RunEv1FromEforA Then
              AddHandler Me.ev1FromEforA, AddressOf oA.revFromEforA
              RaiseEvent ev1FromEforA()
              RemoveHandler Me.ev1FromEforA, AddressOf oA.revFromEforA
              RunEv1FromEforA = True
            End If
          End If
        End Sub
      End Class
      End Module
```

Тестування алгоритму:

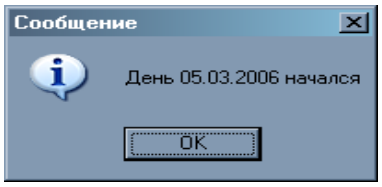

Рисунок 1.54 – EventDynamic (1)

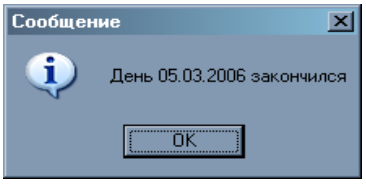

Рисунок 1.55 – EventDynamic (2)

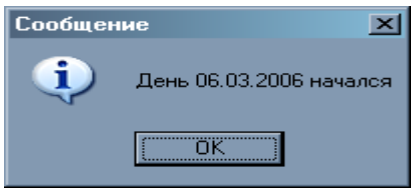

Рисунок 1.56 – EventDynamic (3)

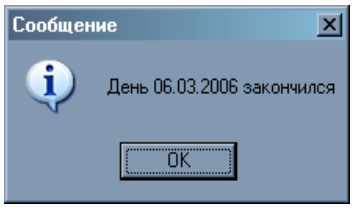

Рисунок 1.57 – EventDynamic (4)

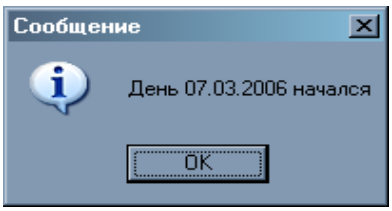

Рисунок 1.58 – EventDynamic (5)

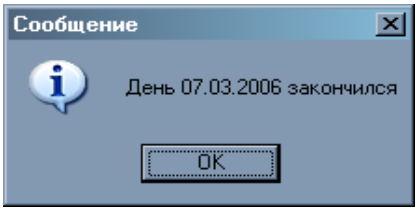

Рисунок 1.59 – EventDynamic (6)

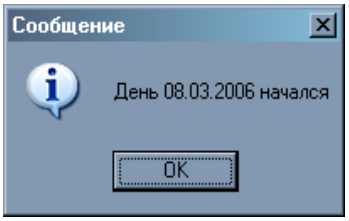

Рисунок 1.60 – EventDynamic (7)

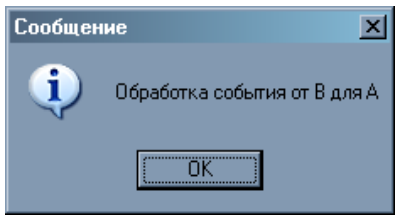

Рисунок 1.61 – EventDynamic (8)

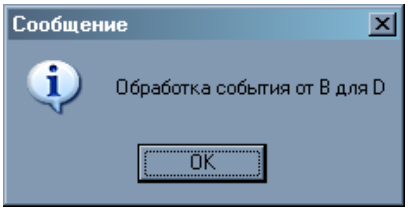

Рисунок 1.62 – EventDynamic (9)

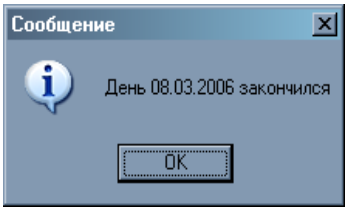

Рисунок 1.63 – EventDynamic (10)

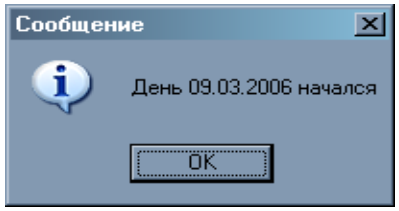

Рисунок 1.64 – EventDynamic (11)

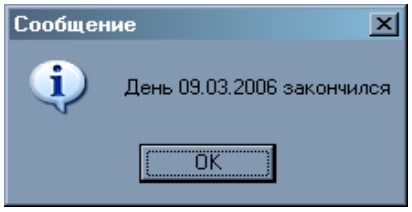

Рисунок 1.65 – EventDynamic (12)

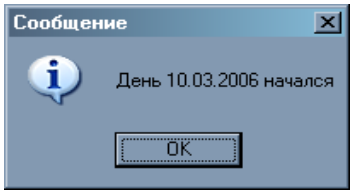

Рисунок 1.66 – EventDynamic (13)

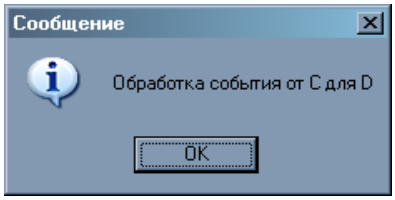

Рисунок 1.67 – EventDynamic (14)

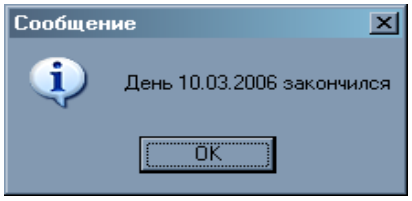

Рисунок 1.68 – EventDynamic (15)

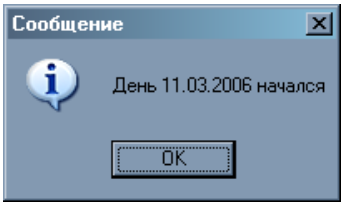

Рисунок 1.69 – EventDynamic (16)

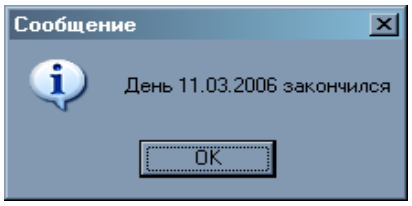

Рисунок 1.70 – EventDynamic (17)

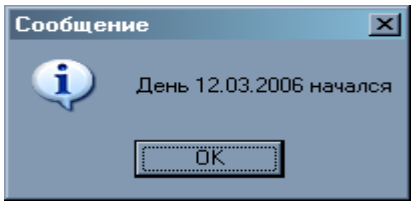

Рисунок 1.71 – EventDynamic (18)

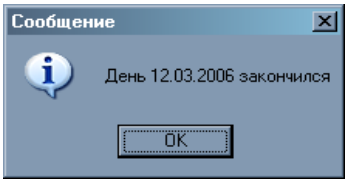

Рисунок 1.72 – EventDynamic (19)

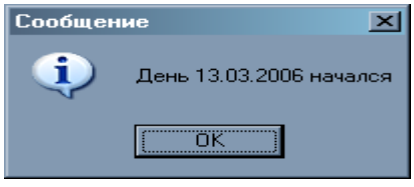

Рисунок 1.73 – EventDynamic (20)

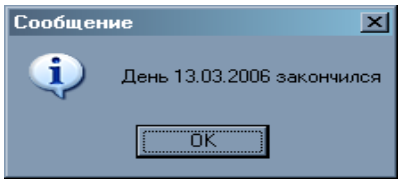

Рисунок 1.74 – EventDynamic (21)

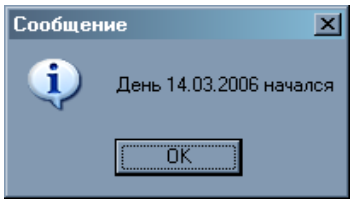

Рисунок 1.75 – EventDynamic (22)

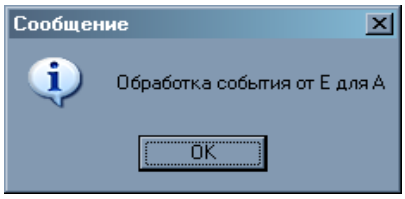

Рисунок 1.76 – EventDynamic (23)

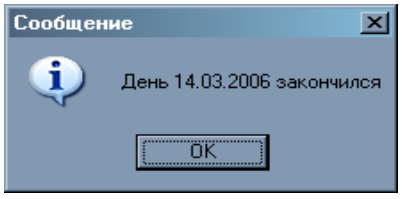

Рисунок 1.77 – EventDynamic (24)

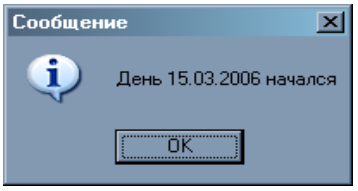

Рисунок 1.78 – EventDynamic (25)

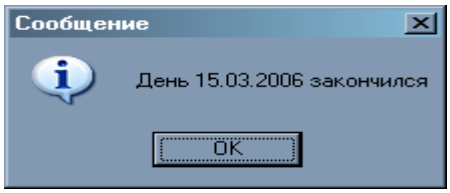

Рисунок 1.79 – EventDynamic (26)

Задача «AbstractClass»

Умова. Хай об'єктний аналіз деякої ЕС приводить до наступної об'єктної моделі ЕС.

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

Нехай класи B і C мають однакову по сенсу властивість «Код службовця», але що має різний формат. Для класу B - це Вномерслужбовця, а для класу C - це Сномерслужбовця. Хай класи B і C має ReadOnly String типу властивість Ім'я (Name), що ініціюється при створенні екземпляра класу значеннями NameB і NameC відповідно. Хай клас A має MustOverride String типу властивість ID - Кодслужбовця.

Завдання:

1. Створити об'єкти oB і oC, ініціюючи властивість Name значеннями NameB і NameC відповідно.

2. Визначити значення властивості ID об'єктів oB і oC деяким значенням.

3. Відобразити значення властивості ID.

Примітка:

1. Спадкоємство класу реалізують словом Inherits і далі пишуть ім'я успадкованого класу

2. У заголовку абстрактного класу SystemX повинне бути слово MustInherit.

3. Цей клас має конструктор для ініціалізації властивості Name

4. У цьому класі реалізоване ReadOnly властивість Name, а властивість ID не має тіла і помічено словом MustOverride

5. У похідних класах властивість ID повинна бути реалізована і помічена словом Overrides

Розв'язок

Обговорення алгоритму: побудуємо Windows Додаток, головне вікно якого містить кнопки Start і Close.

Сценарій інтерфейсу:

1. Натиснення Close-завершення Додатку.

2. Натиснення Start-виконання Додатку.

Ідея алгоритму

Властивості об'єктів проекту:

Об'єкт	**Ім'я**	**Напис**
Форма	frmAbstractClass	AbstractClass
Кнопка	btnStart	Start
Кнопка	btnClose	Close

Вихідний код алгоритму:

```
Module Module1
    Sub Main()
        Dim oA As New A("NameA  ", "A1")
        Dim oB As New B("NameB  ", "B1")
        Dim oC As New C("NameC  ", "C1")
        Dim oD As New D("NameD  ")
        Dim oE As New E("NameE  ")
        oD.ID = "D1"
        oE.ID = "E1"
        MsgBox("Name- " & oA.Name & "  ID - " & oA.ID)
        MsgBox("Name- " & oB.Name & "  ID - " & oB.ID)
        MsgBox("Name- " & oC.Name & "  ID - " & oC.ID)
        MsgBox("Name- " & oD.Name & "  ID - " & oD.ID)
        MsgBox("Name- " & oE.Name & "  ID - " & oE.ID)

        oA.ID = InputBox("ID", "", "A1")
```

```
        oB.ID = InputBox("ID", "", "B1")
        oC.ID = InputBox("ID", "", "C1")
        oD.ID = InputBox("ID", "", "D1")
        oE.ID = InputBox("ID", "", "E1")

        MsgBox("Name- " & oA.Name & "ID - " & oA.ID)
        MsgBox("Name- " & oB.Name & "ID - " & oB.ID)
        MsgBox("Name- " & oC.Name & "ID - " & oC.ID)
        MsgBox("Name- " & oD.Name & "ID - " & oD.ID)
        MsgBox("Name- " & oE.Name & "ID - " & oE.ID)
    End Sub
    'Класс SystemX
    Public MustInherit Class SystemX
        Private mName As String
        Public Sub New(ByVal aName As String)
            mName = aName
        End Sub
        'Реализовываем свойство Name
        Public ReadOnly Property Name() As String
            Get
                Return mName
            End Get
        End Property
        'Не реализованное свойство Name
        Public MustOverride Property ID() As String
    End Class
    'Класс A
    Public Class A
        Inherits SystemX
        Private nID As String
        Dim firstSimvol As String
        Dim Dlina As String
        Public Sub New(ByVal aName As String, ByVal aID As String)
            MyBase.New(aName)
            nID = aID
        End Sub
        Public Overrides Property ID() As String
            Get
                Return nID
            End Get
            Set(ByVal Value As String)
                firstSimvol = Mid(Value, 1, 1)
                Dlina = Len(Value)
                If UCase(firstSimvol) = "A" And IsNumeric(Mid(Value, 2, Dlina - 1))
= True Then
                    nID = Value
                Else
                    MsgBox("Не тот формат для A")
                    ID = InputBox("ID", "", "A1")
                End If
            End Set
        End Property
    End Class
    'Класс B
    Public Class B
        Inherits SystemX
        Private nID As String
        Dim firstSimvol As String
        Dim Dlina As String
        Public Sub New(ByVal aName As String, ByVal aID As String)
            MyBase.New(aName)
            nID = aID
```

```
        End Sub
        Public Overrides Property ID() As String
            Get
                Return nID
            End Get
            Set(ByVal Value As String)
                firstSimvol = Mid(Value, 1, 1)
                Dlina = Len(Value)
                If UCase(firstSimvol) = "B" And IsNumeric(Mid(Value, 2, Dlina - 1))
= True Then
                    nID = Value
                Else
                    MsgBox("Не тот формат для B")
                    ID = InputBox("ID", "", "B1")
                End If
            End Set
        End Property
    End Class
    'Класс C
    Public Class C
        Inherits SystemX
        Private nID As String
        Dim firstSimvol As String
        Dim Dlina As String
        Public Sub New(ByVal aName As String, ByVal aID As String)
            MyBase.New(aName)
            nID = aID
        End Sub
        Public Sub New(ByVal aName As String)
            MyBase.New(aName)
        End Sub
        Public Overrides Property ID() As String
            Get
                Return nID
            End Get
            Set(ByVal Value As String)
                firstSimvol = Mid(Value, 1, 1)
                Dlina = Len(Value)
                If UCase(firstSimvol) = "C" And IsNumeric(Mid(Value, 2, Dlina - 1))
= True Then
                    nID = Value
                Else
                    MsgBox("Не тот формат для C")
                    ID = InputBox("ID", "", "C1")
                End If
            End Set
        End Property
    End Class
    'Класс D
    Public Class D
        Inherits C
        Private nID As String
        Dim firstSimvol As String
        Dim dlina As Integer
        Public Sub New(ByVal aName As String)
            MyBase.New(aName)
        End Sub
        Public Overrides Property ID() As String
            Get
                Return nID
            End Get
            Set(ByVal Value As String)
```

```
                firstSimvol = Mid(Value, 1, 1)
                dlina = Len(Value)
                If UCase(firstSimvol) = "D" And IsNumeric(Mid(Value, 2, dlina - 1))
= True Then
                    nID = Value
                Else
                    MsgBox("Не тот формат!!!")
                    ID = InputBox("ID", "", "D1")
                End If
            End Set
        End Property
    End Class
    'Класс E
    Public Class E
        Inherits D
        Private nID As String
        Public Sub New(ByVal aName As String)
            MyBase.New(aName)
        End Sub
        Dim firstSimvol As String
        Dim dlina As Integer
        Public Overrides Property ID() As String
            Get
                Return nID
            End Get
            Set(ByVal Value As String)
                firstSimvol = Mid(Value, 1, 1)
                dlina = Len(Value)
                If UCase(firstSimvol) = "E" And IsNumeric(Mid(Value, 2, dlina - 1))
= True Then
                    nID = Value
                Else
                    MsgBox("Не тот формат!!!")
                    ID = InputBox("ID", "", "E1")
                End If
            End Set
        End Property
    End Class
End Module
```

Тестування алгоритму: тестування може мати вигляд:

Тест 1

Якщо: oB.ID =5, oC.ID=C1

То очікується відповідь: «Не той формат для B». Дійсно:

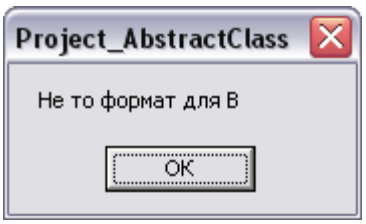

Рисунок 1.80 – AbstractClass (1)

Тест 2

Якщо: oB.ID =B1, oC.ID=C1

То очікується відповідь: «Властивість ID об'єкту NameИ є B1» і «Свойство ID об'єкту NameC1 є C1». Дійсно:

Рисунок 1.81 – AbstractClass (2)

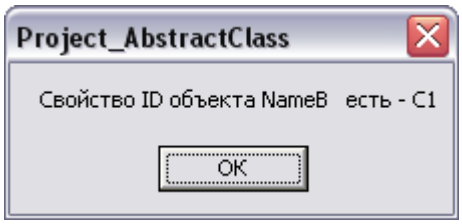

Рисунок 1.82 – AbstractClass (3)

Задача «Polimorfizm»

Умова. Нехай об'єктний аналіз деякої ЕС приводить до наступної об'єктної моделі ЕС

```
Employee (Employees)
|
|__A
|__B
   |__C
   |__D
```

|__E

Нехай клас Employee є абстрактний клас. Має конструктор властивості Name. Реалізує властивість Name. Має MustOverride ReadOnly властивість Salary Decimal типу. Має MustOverride метод RaiseSalary без параметрів.

Нехай кожен клас (A,B,C,D,E) має ReadOnly String типу властивість Ім'я (Name), що ініціюється при створенні екземпляра класу значенням NameИмяКласса.

Нехай кожен клас має ReadOnly Decimal типу властивість Зарплата, що ініціюється при створенні екземпляра класу деяким значенням.

Нехай кожен клас має переобтяжений метод ПідвищенняЗарплати (RaiseSalary(ByVal ПроцентПідвищенняЗарплати as Byte)), що змінює властивість Salary і що реалізовує деякий алгоритм підвищення зарплати.

Нехай алгоритми підвищення зарплати для всіх класів різні.

Завдання:

1. Утворити екземпляри класів (об'єкти)
2. Помістити об'єкти в колекцію Employees (на основі класу CollectionBase)
3. Ітеруя по колекції, відобразити значення властивостей – Ім’яОб'єкта і Зарплата
4. Ітеруя по колекції, викликати метод ПідвищенняЗарплати для кожного об'єкту і встановити нову зарплату
5. Ітеруя по колекції, відобразити значення властивостей - Ім’яОб'єкта і Зарплата

Розв’язок

Обговорення алгоритму: побудуємо Windows Додаток, головне вікно якого містить кнопки Start і Close.

Сценарій інтерфейсу:

1. Натиснення Close-завершення Додатку.
2. Натиснення Start-виконання Додатку.

Ідея алгоритму

Властивості об'єктів проекту:

Об’єкт	**Ім’я**	**Напис**
Форма	frmPolimorfizm	Polimorfizm
Кнопка	btnStart	Start
Кнопка	btnClose	Close

Вихідний код алгоритму:

```
Public Class A
  Inherits Employee
  Private mSalary As Decimal
  Public Sub New(ByVal aName As String, ByVal aSalary As Decimal)
    MyBase.New(aName)
    mSalary = aSalary
  End Sub
  Public Overrides ReadOnly Property Salary() As Decimal
    Get
      Return mSalary
    End Get
  End Property
  Public Overrides Sub RaiseSalary()
    mSalary = InputBox("Salary A = ", "", "200")
  End Sub
End Class
Public Class B
  Inherits Employee
  Private mSalary As Decimal
  Public Sub New(ByVal aName As String, ByVal aSalary As Decimal)
    MyBase.New(aName)
    mSalary = aSalary
  End Sub
  Public Overrides ReadOnly Property Salary() As Decimal
    Get
      Return mSalary
    End Get
  End Property
  Public Overrides Sub RaiseSalary()
    mSalary = InputBox("Salary B = ", "", "200")
  End Sub
End Class
Public Class C
  Inherits B
  Private mSalary As Decimal
  Public Sub New(ByVal aName As String, ByVal aSalary As Decimal)
    MyBase.New(aName, aSalary)
    mSalary = aSalary
  End Sub
  Public Overrides ReadOnly Property Salary() As Decimal
    Get
      Return mSalary
    End Get
  End Property
  Public Overrides Sub RaiseSalary()
    mSalary = InputBox("Salary C = ", "", "200")
  End Sub
End Class
Public Class D
  Inherits B
  Private mSalary As Decimal
  Public Sub New(ByVal aName As String, ByVal aSalary As Decimal)
    MyBase.New(aName, aSalary)
    mSalary = aSalary
  End Sub
  Public Overrides ReadOnly Property Salary() As Decimal
    Get
      Return mSalary
    End Get
  End Property
```

```
    Public Overrides Sub RaiseSalary()
      mSalary = InputBox("Salary D = ", "", "200")
    End Sub
  End Class
  Public Class E
    Inherits D
    Private mSalary As Decimal
    Public Sub New(ByVal aName As String, ByVal aSalary As Decimal)
      MyBase.New(aName, aSalary)
      mSalary = aSalary
    End Sub
    Public Overrides ReadOnly Property Salary() As Decimal
      Get
        Return mSalary
      End Get
    End Property
    Public Overrides Sub RaiseSalary()
      mSalary = InputBox("Salary E = ", "", "200")
    End Sub
  End Class
  Public Class Employees
    Inherits System.Collections.CollectionBase
    Default Public ReadOnly Property Item(ByVal Index As Integer) As Object
      Get
        Return CType(List.Item(Index), Object)
      End Get
    End Property
    Public Sub Add(ByVal aObj As Object)
          List.Add(aObj)
    End Sub
  End Class
  Public MustInherit Class Employee
    Private mName As String
    Private mSalary As Decimal
    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 MustOverride ReadOnly Property Salary() As Decimal
    Public MustOverride Sub RaiseSalary()
  End Class
  Sub Main()
      Dim oA As New A("NameA", 100)
      Dim oB As New B("NameB", 100)
      Dim oC As New C("NameC", 100)
      Dim oD As New D("NameD", 100)
      Dim oE As New E("NameE", 100)
      Dim myEmployees As New Employees()
      myEmployees.Add(oA)
      myEmployees.Add(oB)
      myEmployees.Add(oC)
      myEmployees.Add(oD)
      myEmployees.Add(oE)
      Dim en As Employee
      For Each en In myEmployees
          MsgBox("Ім’я об’єкта = " & en.Name & "     Зарплата об’єкта = " &
en.Salary)
      Next
```

```
            oA.RaiseSalary()
            MsgBox("Зарплата" & oA.Salary)
            oB.RaiseSalary()
            oC.RaiseSalary()
            oD.RaiseSalary()
            oE.RaiseSalary()
            myEmployees.Add(oA)
            For Each en In myEmployees
                MsgBox("Имя объекта = " & en.Name & "      Зарплата об'єкта = " &
en.Salary)
            Next
        End Sub
```

Тестування алгоритму:

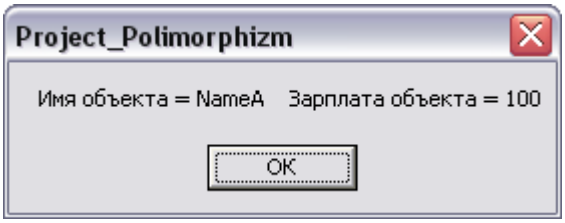

Рисунок 1.83 – Polimorfizm (1)

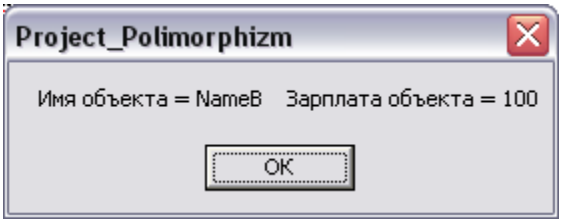

Рисунок 1.84 – Polimorfizm (2)

Рисунок 1.85 – Polimorfizm (3)

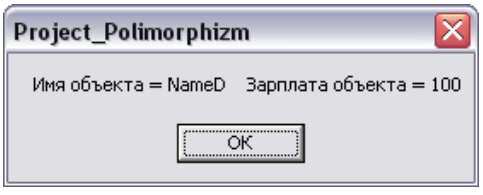

Рисунок 1.86 – Polimorfizm (4)

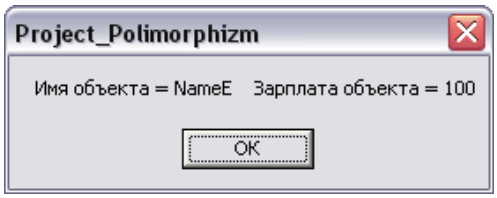

Рисунок 1.87 – Polimorfizm (5)

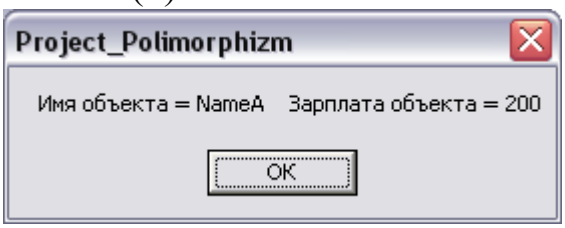

Рисунок 1.88 – Polimorfizm (6)

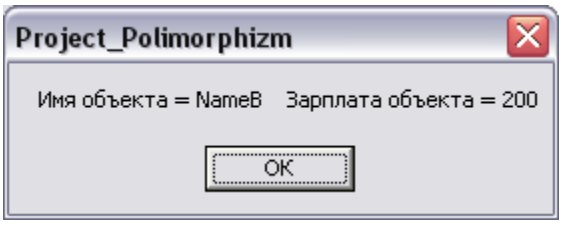

Рисунок 1.89 – Polimorfizm (7)

Рисунок 1.90 – Polimorfizm (8)

Рисунок 1.91 – Polimorfizm (9)

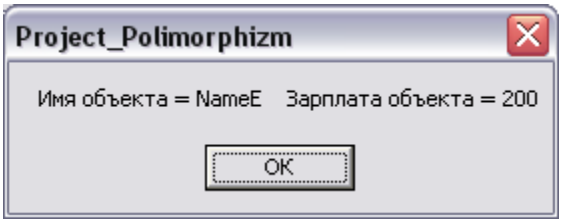

Рисунок 1.92 – Polimorfizm (10)

Обробка меню додатку

Нехай на етапі проектування на форму додатка з ім’ям frmMainLabPr2 додан екземпляр стандартного класу MainMenu з ім’ям MainMenuLabPr2.

Організуємо реакцію по властивості Text на клацання елементу меню CType(sender, MenuItem).Text оператором Select Case.

Нехай проект має форму з ім’ям frmAbout.

Для відображення форми виконаємо код:

```
Dim oAbout As New frmAbout()
oAbout.Show
```

Нижче приведено зразковий код обробки меню додатка.

```
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 реалізовано --")
  End Select
End Sub
```

Питання самоконтролю

Завдання «ReadWrite»
Покажіть процедуру властивості.
Покажіть точку виклику властивості.
Назвіть ім'я властивості?
З яких блоків складається процедура властивості?
Який за рахунком повинна бути процедура властивості в класі?
Яким словом позначається процедура властивості?
Завдання «ReadOnly»
Покажіть процедуру властивості.
Покажіть точку виклику властивості.
Назвіть ім'я властивості?
Який блок забороняє ключове слово ReadOnly?
Який за рахунком повинна бути процедура властивості в класі?
Яким словом позначається процедура властивості?
Чи можна зробити властивість Read тільки на 10-й раз?
Завдання «WriteOnly»
Покажіть процедуру властивості.
Покажіть точку виклику властивості.
Назвіть ім'я властивості?
Який блок забороняє ключове слово WriteOnly?
Який за рахунком повинна бути процедура властивості в класі?
Яким словом позначається процедура властивості?
Завдання «WriteOnce»
Покажіть процедуру властивості.
Покажіть точку виклику властивості.
Назвіть ім'я властивості?
Який блок забороняє ключове слово WriteOnly?
Який за рахунком повинна бути процедура властивості в класі?
Яким словом позначається процедура властивості?
Який блок забороняє це ключове слово?
Чи можна зробити властивість Write тільки на 10-й раз?
Завдання «WriteOnceReadMany»
Покажіть процедуру властивості.
Покажіть точку виклику властивості.

Назвіть ім'я властивості?

З яких блоків складається процедура властивості?

Який за рахунком повинна бути процедура властивості в класі?

Яким словом позначається процедура?

Зробіть властивість ПісатьДваждиЧітатьМного

Завдання «OverloadsConstructor»

Обведіть перевантажені конструктори.

Обведіть точки виклику перевантажених конструкторів.

З'єднайте їх.

Що визначає виклик перевантажених конструкторів?

А якщо число аргументів однаково?

Ім'я переобтяженого конструктора? А іншого?

Який за рахунком повинна бути процедура конструктора в класі?

Яким словом позначається перевантажений конструктор?

Завдання «OverloadsMethod»

Покажіть перевантажені методи.

Покажіть точки виклику перевантажених методів.

З'єднайте їх.

Що визначає виклик перевантажених методів?

Що визначає виклик перевантажених методів, якщо число аргументів однаково?

Ім'я перевантаженого методу? А іншого?

Який за рахунком повинна бути процедура перевантаженого методу в класі?

Яким словом позначається перевантажений метод?

Завдання «Inherits»

Яким словом успадковують класи?

На якому місці в класі пишуть слово спадкування базового класу?

Чи використовуєте Ви захищені члени базового класу?

Чи використовуєте MyBase для виклику членів базового класу?

Сенс слова Protected?

Сенс слова MyBase?

Намалюйте діаграму об'єктної моделі системи?

Завдання «EventDynamic»

Намалюйте діаграму об'єктної моделі системи?

Намалюйте діаграму моделі потоку подій?

Намалюйте «розташування учасників Битви?»

Де оголошується подію?
Де призначається обробник?
Де генерується подія?
Де обробник події?
Яким словом події динамічно призначають обробник?
Яким словом знищують динамічно призначений обробник?
Якщо в класі є RaiseEvent, що ще має бути в цьому класі?
Чи використовуєте власні класи подій?
Який стандартний клас події використовується за умовчанням?
Завдання «AbstractClass»
Що таке абстракція?
Покажіть абстракцію в класі
Що таке інтерфейс?
Яким словом позначається абстрактний клас?
Що має бути присутнім в абстрактному класі?
Завдання «Polimorfizm»
Що таке поліморфізм?
Що таке інтерфейс?
Знайдіть інтерфейс в задачі.
Що визначає поліморфний клас?

ВИСНОВОК

Розглянуті засоби системи VB. NET досить ефективно дозволяють створювати віртуальні образи елементів реальних систем.

В результаті тестування програмних моделей фрагментів систем виявлено особливості роботи інструментальних засобів.

В роботі побудовані програмні моделі фрагментів систем, які можуть бути використані при побудові віртуальних образів явищ і сутностей економічних систем.

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

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

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

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

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

5. Джеймс Д. Фокселл «Освой самостоятельно Visual Basic .NET за 24 часа.»: Пер. с англ. – М.: Издательский дом "Вильямс", 2002 г. – 242 с.

6. Культин Н.Б. Visual Basic. Освой на примерах. / Н.Б. Культин. – СПб.: БХВ-Петербург, 2004 г. – 528 с.

7. Лукин С.Н. Понятно о Visual Basic.NET. Самоучитель. / С.Н. Лукин. – М.: Диалог-МИФИ, 2005 г. – 736 с.

8. Б. Сайлер, Дж. Использование Visual Basic 6. / Б. Сайлер, Дж. Споттс. – М.: «Вильямс», 2007. – 832 с.

9. Трусов М.А. Visual Basic .NET. Создание графических объектов и основы программирования. / М.А. Трусов. – М.: НТ Пресс, 2006 г. – 160 с.

ЕЛЕКТРОННЕ НАВЧАЛЬНО-МЕТОДИЧНЕ ВИДАННЯ

Ніколаєнко Денис Володимирович

МЕТОДИЧНІ ВКАЗІВКИ ДО ВИКОНАННЯ ПРАКТИЧНИХ РОБІТ З ДИСЦИПЛІНИ «ОБ'ЄКТНЕ ПРОГРАМУВАННЯ» (ДЛЯ СТУДЕНТІВ НАПРЯМКУ 0501 - ЕКОНОМІКА І ПІДПРИЄМНИЦТВО СПЕЦІАЛЬНІСТЬ 6.050100 - ЕКОНОМІЧНА КІБЕРНЕТИКА)

Підписано до випуску 2012 р. Гарнітура Times New.
Умов. друк. арк. Зам. №

Державний вищий навчальний заклад
«Донецький національний технічний університет»
Автомобільно-дорожній інститут
84646, м. Горлівка, вул. Кірова, 51
E-mail: druknf@rambler.ru

Редакційно-видавничий відділ

АДІ ДВНЗ «ДонНТУ»

Свідоцтво про внесення до Державного реєстру видавців, виготовників і розповсюджувачів продукції ДК № 2982 від 21.09.2007 р.