**Лабораторная работа № 6. Методы. Делегаты**

**Цель работы:** Приобретение навыков практического применения, закрепление знаний при создания простейших программ с использованием методов, делегатов.

**Теоретические сведения**

Делегат – это тип, который представляет собой ссылки на методы с определенным списком параметров и возвращаемым типом.

Делегаты используются для передачи методов в качестве аргументов к другим методам. Обработчики событий — это ничто иное, как методы, вызываемые с помощью делегатов. Создание пользовательского метода и класса, например, элемента управления Windows, может вызывать метод при возникновении определенного события. В следующем примере показано объявление делегата:

delegate возвращаемый\_тип имя (список\_параметров);

public delegate int PerformCalculation(int x, int y);

Делегату можно назначить любой метод из любого доступного класса или структуры, соответствующий сигнатуре делегата, которая состоит из типа возвращаемого значения и параметров. Этот метод должен быть статическим методом или методом экземпляра. Это позволяет программно изменять вызовы метода, а также включать новый код в существующие классы. Если сигнатура делегата известна, то можно назначить собственный метод.

 В следующем примере объявляется делегат с именем Del, который может инкапсулировать метод, использующий в качестве аргумента значение [string](http://msdn.microsoft.com/ru-ru/library/362314fe.aspx) и возвращающий значение [void](http://msdn.microsoft.com/ru-ru/library/yah0tteb.aspx):

public delegate void Del(string message);

Объект делегата обычно создается указанием имени метода, для которого делегат будет служить оболочкой, или с помощью [анонимного метода](http://msdn.microsoft.com/ru-ru/library/0yw3tz5k.aspx). После создания экземпляра делегата вызов метода, выполненный в делегате, передается делегатом в этот метод. Параметры, передаваемые делегату вызывающим объектом, передаются в метод, а возвращаемое методом значение (при его наличии) возвращается делегатом в вызывающий объект. Эта процедура называется вызовом делегата. Делегат, для которого создан экземпляр, можно вызвать, как если бы это был метод, для которого создается оболочка. Примеры.

// Create a method for a delegate.

Public static void DelegateMethod(string message)

{

System.Console.WriteLine(message);

}

// Instantiate the delegate.

Del handler = DelegateMethod;

// Call the delegate.

handler("Hello World");

В следующем примере метод использует тип Del как параметр:

public void MethodWithCallback(int param1, int param2, Del callback)

{

callback("The number is: " + (param1 + param2).ToString());

}

Затем можно передать созданный ранее делегат в данный метод:

MethodWithCallback(1, 2, handler);

и получить следующие выходные данные в окне консоли:

The number is: 3

**Задание**

1. Разработайте консольное приложение, которое в зависимости от условия вызывает нужный делегат и рассчитывает значение функции.
2. Разработайте приложение, работающее как в режиме консольного приложения, так и в режиме Windows Form. Оба приложения используют общий класс и выдают сообщения о произведённых расчетах.

**Варианты задания 1**



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Функция1 | Функция 2 | Функция 3 | Параметр а | Интервал | Шаг табулирования |
| 1 |  |  |  | 2 | [-1,6; 3,7] | 0,3 |
| 2 |  |  |  | 5 | [-2,8; 8,2] | 0,4 |
| 3 |  |  |  | 1 | [-1,7; 2,6] | 0,25 |
| 4 |  |  |  | 3 | [-2,2; 7,4] | 0,23 |
| 5 |  |  |  | 4 | [-1,1; 7,9] | 0,8 |
| 6 |  |  |  | 1 | [-1,2; 2,6] | 0,1 |
| 7 |  |  |  | 1 | [-1,7; 2,4] | 0,3 |
| 8 |  |  |  | 5 | [-4,3; 8,0] | 0,5 |
| 9 |  |  |  | 3 | [-9,1; 5,8] | 0,14 |
| 10 |  |  |  | 1 | [-3,4; 2,5] | 0,23 |

**Варианты задания 2**

|  |  |  |
| --- | --- | --- |
| **Вариант** | **Фигура** | **Рассчитать** |
| 1, 4, 7, 10 | Прямоугольник | Периметр и площадь |
| 2, 5, 8 | Квадрат | Периметр и площадь |
| 3, 6, 9 | Круг | Длину окружности и площадь |

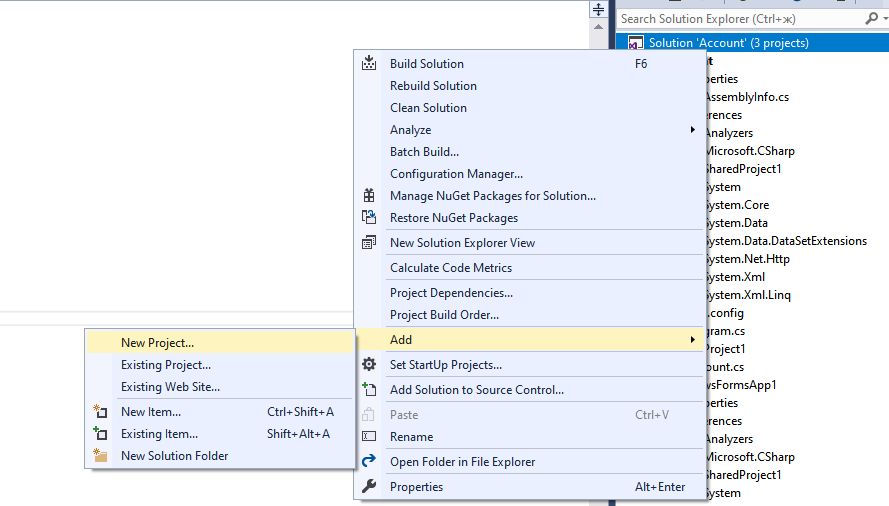
**Пример выполнения**

**Задание1**

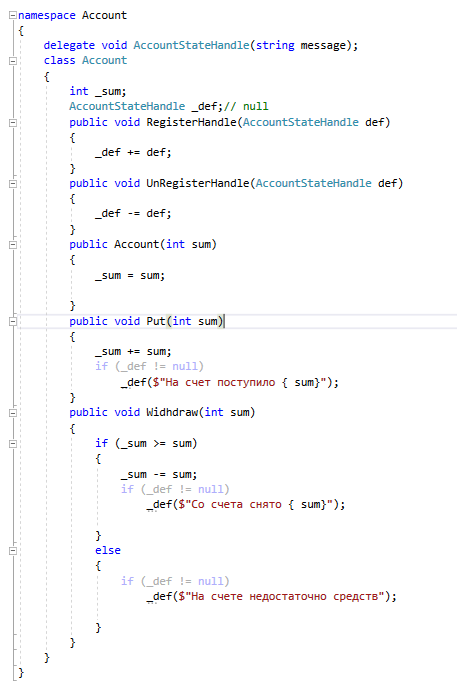
****

**Задание 2**

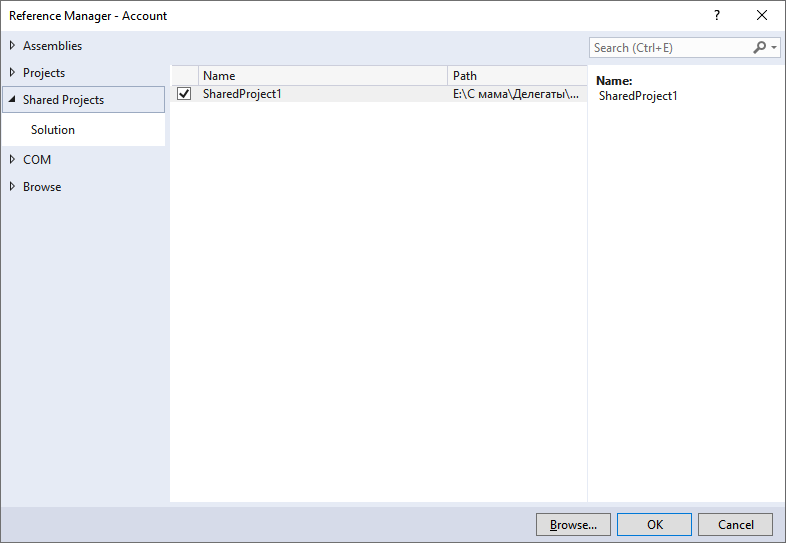
1. Создать консольный проект.
2. Добавить в консольный проект новый проект SharedProject.



1. В SharedProject добавить класс и описать структуру класса.



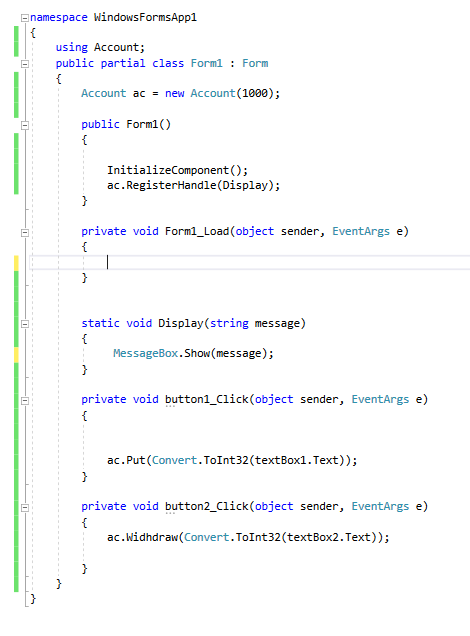
1. Предоставить консольному приложению доступ к созданному классу.



1. Реализовать логику работы консольного приложения



1. Добавить в решение новый проект WindowsForm. Настроить внешний вид формы и реализовать логику работы. Не забыть подключить ссылки на класс.



Дополнительная информация:

<https://metanit.com/sharp/tutorial/3.13.php>