**13 СОБЫТИЯ**

Задание 1. Создать 3 метода, подсчитывающих периметр, площадь квадрата, и выводящий сторону треугольника. Создать делегат. В программе

вызывать все три метода при помощи делегата.

Листинг программы:

try

{

Square evt = new Square();

evt.Notify += new SquareDelegate(evt.Perimeter);

evt.Notify += new SquareDelegate(evt.Sqr);

evt.Notify += new SquareDelegate(evt.TriangleSide);

Console.Write("Введите сторону квадрата: ");

int side = Convert.ToInt32(Console.ReadLine());

evt.Start(side);

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

delegate void SquareDelegate(int side);

class Square

{

public event SquareDelegate? Notify;

public void Start(int side)

{

Notify?.Invoke(side);

}

public void Perimeter(int side)

{

int per = side \* 4;

Console.WriteLine(per);

}

public void Sqr(int side)

{

int sqr = side \* side;

Console.WriteLine(sqr);

}

public void TriangleSide(int side)

{

Console.WriteLine(side);

}

}

Таблица 13.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 5 | 20  25  5 |

Анализ результатов:

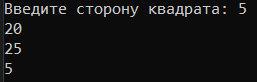


Рисунок 13.1 – Результат работы программы

Источник: собственная разработка

Задание 2. На основе своей программы создать метод, в котором делегат

будет параметром.

Листинг программы:

try

{

Square evt = new Square();

SquareDelegate deleg = evt.Perimeter;

deleg += evt.Sqr;

deleg += evt.TriangleSide;

evt.Notify += new SquareDelegate(evt.Perimeter);

evt.Notify += new SquareDelegate(evt.Sqr);

evt.Notify += new SquareDelegate(evt.TriangleSide);

Console.Write("Введите сторону квадрата: ");

int side = Convert.ToInt32(Console.ReadLine());

evt.Start(side);

evt.DelegateInit(deleg, side);

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

delegate void SquareDelegate(int side);

class Square

{

public event SquareDelegate? Notify;

public void Start(int side)

{

Notify?.Invoke(side);

}

public void Perimeter(int side)

{

int per = side \* 4;

Console.WriteLine(per);

}

public void Sqr(int side)

{

int sqr = side \* side;

Console.WriteLine(sqr);

}

public void TriangleSide(int side)

{

Console.WriteLine(side);

}

public void DelegateInit(SquareDelegate deleg, int side)

{

deleg(side);

}

}

Таблица 13.2 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 5 | 20 25 5 20 25 5 |

Анализ результатов:

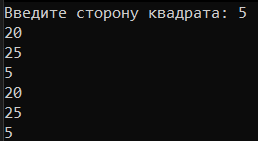


Рисунок 13.2 – Результат работы программы

Источник: собственная разработка

Задание 3. Для класса MyInfo содержащего поле name (Ваше имя) создать событие выводящее оповещение в случае изменения значения поля name.

Листинг программы:

try

{

Console.Write("Введите имя: ");

string name = Console.ReadLine();

MyInfo inf = new MyInfo();

inf.Notify += inf.DisplayMessage;

inf.ChangeName(name);

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

delegate void Info(string message);

class MyInfo

{

public string name = "Tom";

public event Info? Notify;

public void ChangeName(string name)

{

this.name = name;

Notify?.Invoke($"Имя было изменено на {name}");

}

public void DisplayMessage(string message)=>Console.WriteLine(message);

}

Таблица 13.3 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Антон | Имя было изменено на Антон |

Анализ результатов:

C:\Users\1234321\Downloads\govno32.png

Рисунок 13.3 – Результат работы программы

Источник: собственная разработка

Задание 4. Создать класс, в нем делегат и соответствующее ему событие. Создать два класса-наблюдателя, в методах которых будет описана их реакция на событие. Добавить 2 обработчика к событию из первого класса и один из второго. Вывести результат на экран. Удалить один обработчик события и вывести результат на экран.

Листинг программы:

try

{

Console.Write("Введите имя: ");

string name = Console.ReadLine();

MyInfo inf = new MyInfo();

X xObj = new X();

Y yObj = new Y();

inf.Notify += xObj.XHandler;

inf.Notify += yObj.YHandler;

inf.ChangeName(name);

Console.WriteLine();

inf.Notify -= yObj.YHandler;

inf.ChangeName(name);

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

delegate void Info(string message);

class MyInfo

{

public string name = "Tom";

public event Info? Notify;

public void ChangeName(string name)

{

this.name = name;

Console.WriteLine($"Имя было изменено на {name}");

Notify?.Invoke(name);

}

}

class X

{

public void XHandler(string name) => Console.WriteLine($"{name} - хорошее имя");

}

class Y

{

public void YHandler(string name) => Console.WriteLine($"{name} - прекрасное имя");

}

Таблица 13.4 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Антон | Имя было изменено на Антон  Антон - хорошее имя  Антон - прекрасное имя  Имя было изменено на Антон  Антон - хорошее имя |

Анализ результатов:

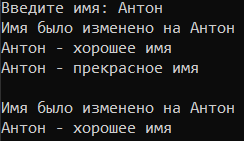


Рисунок 13.4 – Результат работы программы

Источник: собственная разработка