# Лабораторная работа 8. Делегаты, события.

## Задание.

На основе класса из первой работы.

Обязательно проверяйте делегаты на null!

Все типы делегатов объявляйте в общем пространстве имен, вне классов.

1. Делегат без параметров.
2. Добавить к классу поле типа делегат (private).
3. Добавить к классу метод, который заполняет поле типа делегат.
4. Добавьте к классу метод, который запускает делегат, лежащий в поле.
5. В Program создайте объекты, вызывающие данный метод. В качестве параметра передайте подходящий статический метод из Program (сделайте различные варианты разговора). Используйте анонимные делегаты и лямбда-выражения.
6. Создайте групповой делегат.
7. Сделайте питомца, который говорит всеми возможными способами
8. Добавьте к классу метод, который сравнивает эти поля в разных объектах. Сравните 2-х питомцев – они разговаривают одинаково или нет.

public delegate void VoidDelegate();

class Tanker

{

private VoidDelegate \_voidDelegate; //a)

public void RegisterVoidDelegate(VoidDelegate vd) //b)

{

\_voidDelegate = vd;

}

// метод который вызывает делегат без параметров

public void RunVoidDelegate()//c)

{

\_voidDelegate?.Invoke();

}

public bool IsSameVoidDelegate(Tanker t) //g)

{

return this.\_voidDelegate == t.\_voidDelegate;

}

...

}

class Program

{

static void DispMessage()

{

Console.WriteLine(" я - статический метод из класса Program");

}

static void Main(string[] args)

{

// 1. делегат без параметров

// создадим экземпляр танкера

Tanker myTanker = new Tanker("Победа", 100, new int[] { 50, 50, 70, 70, 100, 100 });

// делегат принимает статический метод

VoidDelegate vDelFromStatic = DispMessage;

myTanker.RegisterVoidDelegate(vDelFromStatic);

myTanker.RunVoidDelegate();

// анонимный делегат делегат

VoidDelegate vDelFromAnonimous = delegate

{

Console.WriteLine(" я - анонимный делегат");

};

myTanker.RegisterVoidDelegate(vDelFromAnonimous);

myTanker.RunVoidDelegate();

// лямбда-выражение

VoidDelegate vDelFromLambda = () => Console.WriteLine(" я - лямбда выражение");

myTanker.RegisterVoidDelegate(vDelFromLambda);

myTanker.RunVoidDelegate();

// создадим групповой делегат

VoidDelegate vDelFromGroup = vDelFromStatic;

vDelFromGroup += vDelFromAnonimous;

vDelFromGroup += vDelFromLambda;

// зарегистрируем групповой делегат

myTanker.RegisterVoidDelegate(vDelFromGroup);

Console.WriteLine("Выполним групповой делегат:");

myTanker.RunVoidDelegate();

// создадим еще один экземпляр танкера

Tanker newTanker = new Tanker();

newTanker.RegisterVoidDelegate(vDelFromStatic);

myTanker.RegisterVoidDelegate(vDelFromStatic);

// сравним поля делегатов наших экземпляров

Console.WriteLine("сравним поля делегатов наших экземпляров");

myTanker.RunVoidDelegate();

newTanker.RunVoidDelegate();

if(myTanker.IsSameVoidDelegate(newTanker))

Console.WriteLine("делегаты равны ");

else

Console.WriteLine("делегаты не равны");

Console.WriteLine("Еще раз сравним поля делегатов наших экземпляров");

myTanker.RegisterVoidDelegate(vDelFromLambda);

newTanker.RegisterVoidDelegate(() => Console.WriteLine(" я - лямбда выражение"));

myTanker.RunVoidDelegate();

newTanker.RunVoidDelegate();

if (myTanker.IsSameVoidDelegate(newTanker))

Console.WriteLine("делегаты равны ");

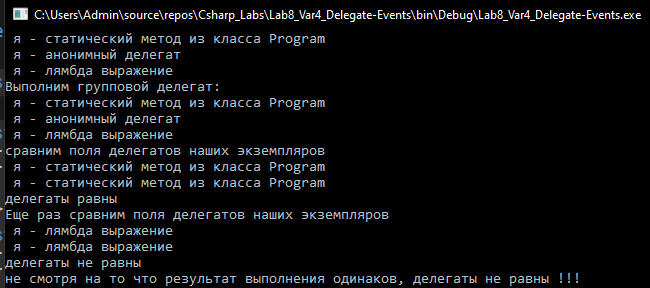
else

Console.WriteLine("делегаты не равны");

Console.WriteLine("не смотря на то что результат выполнения одинаков, делегаты не равны !!!");

}

}



1. Делегат с параметрами
   1. Добавьте к классу делегат с параметром. Например, вы можете задать различные способы набора веса питомцами – добавить аргумент, умножить на аргумент и т.д. Используйте анонимные делегаты и лямбда-выражения.
   2. Аналогично первому заданию создайте метод и вызовите его из Program. (Групповой делегат делать не нужно)

public delegate int ParamDelegate(int f, int k);

class Tanker

{

private ParamDelegate \_paramDelegate;

private event MyEventHandler \_RolloverMayOccur = (k)=> { }; // добавим пустой обработчик

public void RegisterParamDelegate(ParamDelegate pd)

{

\_paramDelegate = pd;

}

// аксессоры для добавления/удаления обработчиков

public event MyEventHandler RolloverMayOccur

{

add

{

\_RolloverMayOccur += value;

}

remove

{

\_RolloverMayOccur -= value;

}

}

// метод изменения загрузки правого борта, принимает делегат с параметрами

public void LoadingStarboard(int k)

{

if (\_paramDelegate != null)

{

int fullnes = 0;

for (int i = 0; i < tanks.Length; i++)

{

if (i % 2 != 0) // применяется только для правого борта(четные танки, отсчет с еденицы)

{

fullnes = \_paramDelegate(tanks[i], k);

if (fullnes < 0)

fullnes = 0;

else if (fullnes > volume)

fullnes = volume;

tanks[i] = fullnes;

}

}

if (!IsStable())

\_RolloverMayOccur("Событие: Возможно опрокидывание танкера!!!");

}

}

...

}

class Program

{

static void DispMessage()

{

Console.WriteLine(" я - статический метод из класса Program");

}

static int MultipleFullnes(int f, int k)

{

return f \* k;

}

static void Main(string[] args)

{

// 2. делегат с параметрами

Console.WriteLine("\tИнфо о танкере до изменения загрузки правого борта(четные танки):");

Console.WriteLine(myTanker.ToString());

myTanker.RegisterParamDelegate((f, k) => f + k);

Console.WriteLine("Добавим к каждому четному танку значение: -40");

myTanker.LoadingStarboard(-40);

Console.WriteLine("\tИнфо о танкере после загрузки правого борта(четные танки):");

Console.WriteLine(myTanker.ToString());

myTanker.RegisterParamDelegate(MultipleFullnes);

Console.WriteLine(" умножим заполненный объем каждого четного танка в 3 раза ");

myTanker.LoadingStarboard(3);

Console.WriteLine("\tИнфо о танкере после загрузки правого борта(четные танки):");

Console.WriteLine(myTanker.ToString());

}

...

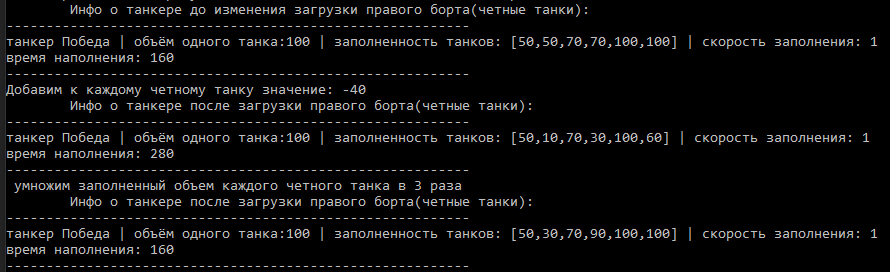
}

static int MultipleFullnes(int f, int k)

{

return f \* k;

}



1. События
   1. Добавьте к классу событие – например, достижение питомцем определенного веса или возраста.
   2. Добавьте классы – обработчики события. (Эти классы могут содержать по 1 методу). Например, в одном классе методы сообщают о недомогании от переедания или старости, во втором эти события приводят к смерти.
   3. Добавьте аксессоры (add/remove) для добавления и удаления обработчиков.

Добавить 2 события – с параметрами и без параметров.

Создать несколько объектов класса и назначить им различные обработчики событий:

* Статический метод любого класса
* Экземплярный метод собственного класса
* Анонимный делегат
* Лямбда-выражение

public delegate int ParamDelegate(int f, int k);

class Tanker

{

private ParamDelegate \_paramDelegate;

private event MyEventHandler \_RolloverMayOccur = (k)=> { }; // добавим пустой обработчик

public void RegisterParamDelegate(ParamDelegate pd)

{

\_paramDelegate = pd;

}

// аксессоры для добавления/удаления обработчиков

public event MyEventHandler RolloverMayOccur

{

add

{

\_RolloverMayOccur += value;

}

remove

{

\_RolloverMayOccur -= value;

}

}

// метод изменения загрузки правого борта, принимает делегат с параметрами

public void LoadingStarboard(int k)

{

if (\_paramDelegate != null)

{

int fullnes = 0;

for (int i = 0; i < tanks.Length; i++)

{

if (i % 2 != 0) // применяется только для правого борта(четные танки, отсчет с еденицы)

{

fullnes = \_paramDelegate(tanks[i], k);

if (fullnes < 0)

fullnes = 0;

else if (fullnes > volume)

fullnes = volume;

tanks[i] = fullnes;

}

}

if (!IsStable())

\_RolloverMayOccur("Событие: Возможно опрокидывание танкера!!!");

}

}

...

}

class EventHandler1

{

public void DisplayRedMessage(string message)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Экземплярный метод: " + message);

Console.ResetColor();

}

}

class EventHandler2

{

public static void DisplayGreenMessage(string message)

{

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine("Статический метод: " + message);

Console.ResetColor();

}

}

static void Main(string[] args)

{

Tanker myTanker = new Tanker("Победа", 100, new int[] { 50, 30, 70, 90, 100, 100 });

// 3. события

// Подпишем методы обработчиков на событие возможного переворота танкера

EventHandler1 eventHandler1 = new EventHandler1();

myTanker.RolloverMayOccur += eventHandler1.DisplayRedMessage; // экземплярный метод

myTanker.RolloverMayOccur += EventHandler2.DisplayGreenMessage; // статический метод

myTanker.RolloverMayOccur += delegate (string p)

{

Console.ForegroundColor = ConsoleColor.Blue;

Console.WriteLine("Анонимный делегат: " + p);

Console.ResetColor();

};

myTanker.RolloverMayOccur += (k) =>

{

Console.ForegroundColor = ConsoleColor.Yellow;

Console.WriteLine("Лямбда: " + k);

Console.ResetColor();

};

Console.WriteLine("\* \* \*\nспециально уменьшим загрузку правого борта, чтобы вызвать событие возможного опрокидывания танкера\n\* \* \*");

myTanker.RegisterParamDelegate((f,k) => f - k);

myTanker.LoadingStarboard(60);

Console.WriteLine(myTanker.ToString());

Console.ReadLine();

}

