Иллюстрации к курсу лекций по дисциплине «Программирование на C#»

Asynchronous Programming

Использованы материалы пособия Daniel Solis, Illustrated C#

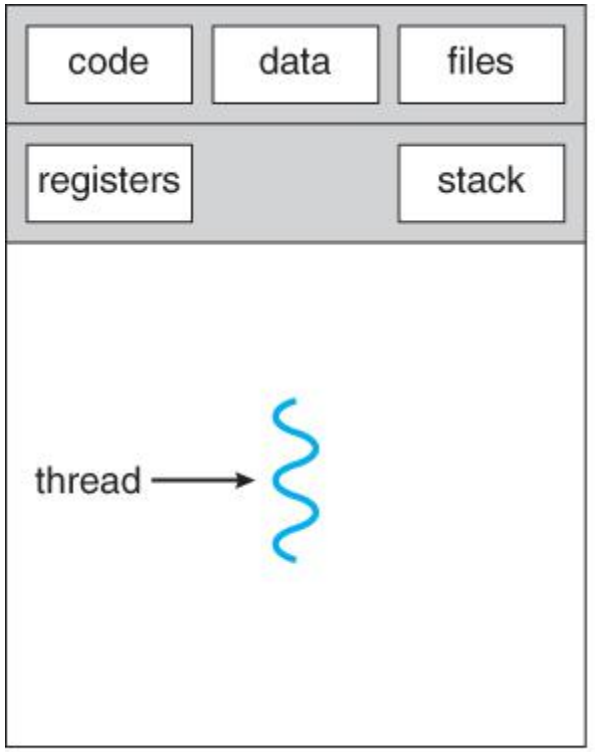
Параллельное программирование

Часть 1

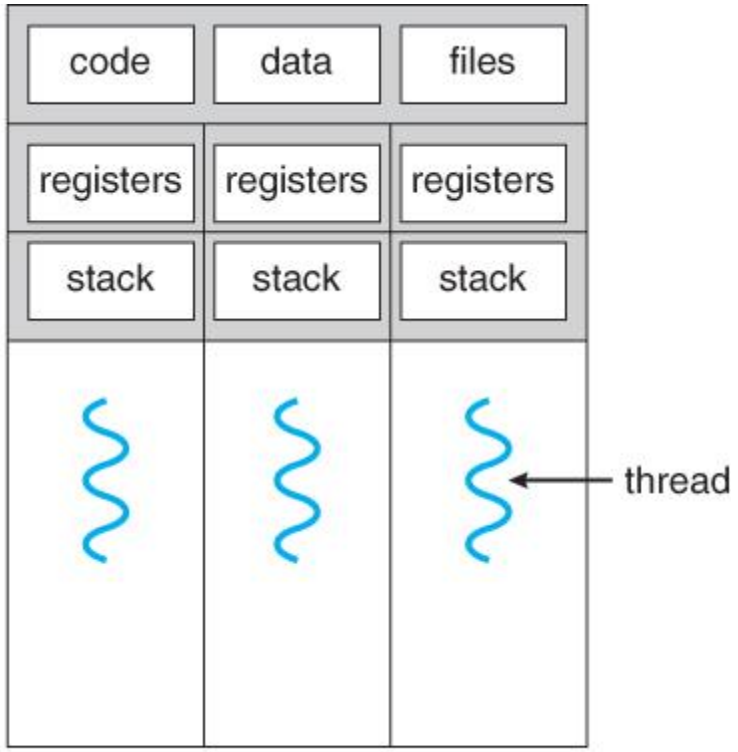
Процессы, потоки и асинхронное программирование

Процесс (Process) представляет собой совокупность ресурсов, которые совместно представляют исполняемую программу.

Поток (Thread) или **поток исполнения** (thread of execution) представляет реально выполняемую часть программы.



single-threaded process



multithreaded process

Синхронное выполнение программы – выполнение в единственном потоке.

Асинхронное выполнение – выполнение программы в нескольких потоках.

Потоки

Основные конструкторы :

Thread(ThreadStart start)
Thread(ParameterizedThreadStart start)

Типы параметров:

delegate void ThreadStart()
delegate void ParameterizedThreadStart(Object obj)

Создание и запуск потока Thread

```
static void Print()
{
    for (int k = 0; k < 5; k++) {
        Console.WriteLine("\t\t\tПоток_2");
    }
}

/// <summary>
/// запустить несколько раз, чтобы увидеть отличия
/// </summary>
public static void Main()
{
    Thread tr = new Thread(Print);
    tr.Start();
    for (int k = 0; k < 5; k++) {
        Console.WriteLine("Поток_1");
        Thread.Sleep(0);    // Внимание, МОЖЕТ передаться управление
    }
    Console.WriteLine("Нажмите любую клавишу!");
    Console.ReadKey(true); // можно убрать
}
```

Результаты при использовании двух потоков

Поток_1
Поток_1
Поток_1
Поток_1
Поток_1
Нажмите любую клавишу!
Поток_2
Поток_2
Поток_2
Поток_2
Поток_2

Поток_1
Поток_2
Поток_2
Поток_2
Поток_2
Поток_2
Поток_1
Поток_1
Поток_1
Поток_1
Нажмите любую клавишу!

Задержки в потоках (метод Sleep)

```
static void Print2()
{
    for (int k = 0; k < 5; k++)
    {
        Thread.Sleep(100);
        Console.WriteLine("\t\t\tПоток_2");
    }
}
public static void Main()
{
    Thread tr = new Thread(Print2);
    tr.Start();
    for (int k = 0; k < 5; k++)
    {
        Thread.Sleep(100);
        Console.WriteLine("Поток_1");
    }
    Console.WriteLine("Нажмите любую клавишу!");
    Console.ReadKey(true); // можно убрать
}
```

Результаты выполнения при задержках в потоках

Добавили операторы Thread.Sleep(100):

Поток_1

Поток_2

Поток_1

Поток_2

Поток_1

Поток_2

Поток_1

Поток_2

Поток_1

Нажмите любую клавишу!

Поток_2

Еще вариант работы:

Поток_1

Поток_2

Поток_2

Поток_1

Поток_1

Поток_2

Поток_1

Поток_2

Поток_2

Поток_1

Нажмите любую клавишу!

Синхронизация потоков и использование Join

```
static void Print3()
{
    for (int k = 0; k < 5; k++)
        Console.WriteLine("\t\t\tПоток_2");
}
/// <summary>
/// Синхронизация потоков
/// </summary>
static void Main()
{
    Thread tr = new Thread(Print3);
    tr.Start();
    tr.Join();  // Ожидание окончания потока tr
    for (int k = 0; k < 5; k++)
        Console.WriteLine("Поток_1");
    Console.WriteLine("Нажмите любую клавишу!");
    Console.ReadKey(true);
}
```

Добавили оператор tr.Join():

Поток_2
Поток_2
Поток_2
Поток_2
Поток_2

Поток_1
Поток_1
Поток_1
Поток_1
Поток_1

Нажмите любую клавишу!

Некоторые члены класса Thread

Thread.CurrentThread – статическое свойство (тип Thread)

IsAlive – свойство (тип bool, поток запущен и не завешен)
Name – свойство (тип string, имя потока, write-once)

Start() – запуск потока (перевод в состояние *running*)
Start(аргумент) – запуск потока с передачей аргумента

Join() – синхронизация (блокирует выполнение вызывающего потока до завершения потока, представленного экземпляром)
Join(аргумент) – синхронизация с таймаутом

Sleep(time) – приостановить исполнение потока на *time* мс
Thread.Sleep(0) – возможность переключиться на исполнение другого потока.

Именование потоков и передача данных в поток

```
static void Print4(object par) {
    Console.WriteLine("\t\t" + Thread.CurrentThread.Name + " запущен из " + (string)par);
}

public static void Main() {
    Thread.CurrentThread.Name = "Main";
    // Thread.CurrentThread.Name = "Main2"; // System.InvalidOperationException: Это
свойство уже назначено и не может изменяться
    Thread tr = new Thread(Print4);
    tr.Name = "Вторичный";
    tr.Start("Main"); // передаем в поток данные
    Console.WriteLine("Поток_1 = " + Thread.CurrentThread.Name);
    Console.WriteLine("Нажмите любую клавишу!");
    Console.ReadKey(true);
}
```

Результаты выполнения программы:

Поток_1 = Main

Нажмите любую клавишу!

Вторичный запущен из Main

Передача результатов работы из потока

```
static void Fib(object par) {
    int[] ar = (int[])par;
    ar[0] = ar[1] = 1;
    for (int j = 2; j < ar.Length; j++)
        ar[j] = ar[j - 1] + ar[j - 2];
}

public static void Main05() {
    Thread tr = new Thread(Fib);
    int[] row = new int[8];
    tr.Start(row);
    tr.Join();
    foreach (int e in row)
        Console.Write(e + "  ");
    Console.WriteLine("\r\nНажмите любую клавишу!");
    Console.ReadKey(true);
}
```

Результаты выполнения программы:

1 1 2 3 5 8 13 21

Нажмите любую клавишу!

Передача данных между потоками

```
static void Fib6(int n, out int[] par) {
    par = new int[n];
    par[0] = par[1] = 1;
    for (int j = 2; j < n; j++)
        par[j] = par[j - 1] + par[j - 2];
}

public static void Main06() {
    int[] row = null;
    Thread tr = new Thread(() => Fib6(6, out row));
    tr.Start();
    tr.Join();
    foreach (int e in row)
        Console.Write(e + "  ");
    Console.WriteLine("\r\nНажмите любую клавишу!");
    Console.ReadKey(true);
}
```

1 1 2 3 5 8
Нажмите любую клавишу!

Локальная переменная передается в два потока

```
static void Print7(string mes) {
    Console.WriteLine(mes);
}

public static void Main07() {
    string line = "Tr_1";
    Thread tr1 = new Thread(() => { Print7(line); });
    tr1.Start();
    // tr1.Join();
    line = "Tr_2";
    Thread tr2 = new Thread(() => { Print7(line); });
    tr2.Start();
    Console.WriteLine("\r\nНажмите любую клавишу!");
    Console.ReadKey(true);
}
```

Результаты выполнения программы:

Нажмите любую клавишу!

Tr_2

Tr_2

Результаты работы и tr1.Join();

Tr_2
Tr_2
Нажмите любую клавишу!
==========
Нажмите любую клавишу!
Tr_2
Tr_2
==========
Tr_2
Нажмите любую клавишу!
Tr_2
==========
С оператором tr1.Join(), всегда:
Tr_1
Нажмите любую клавишу!
Tr_2

Локальные переменные метода потока

```
static void Factorial()
{
    int r = 1;
    for (int i = 0; i < 5; i++)
        r *= i + 1;
    Console.WriteLine(r);
}

public static void Main08()
{
    Thread tr = new Thread(Factorial);
    tr.Start();
    Factorial();
    Console.WriteLine("\r\nНажмите любую клавишу!");
    Console.ReadKey(true);
}
```

Результаты:

120
120
Нажмите любую клавишу!

Parallel Loops

Пространство имен: **System.Threading.Tasks**

Parallel.For loop **Parallel.ForEach loop**

Прототип метода:
void Parallel.For(int *fromInclusive*, int *toExclusive*, Action *body*);

Параметры Parallel.For:
int *fromInclusive* – начальное значение параметра (индекса);

int *toExclusive* – верхняя граница параметра (индекса) серии итераций;

Action *body* – **библиотечный** делегат, представляющий исполняемый на каждой итерации код:

public delegate void Action()

Пример с методом Parallel.For

```
public static void MainFor()
{
      Parallel.For(0, 15, i =>
         Console.WriteLine("The square of {0} is {1}", i, i * i)
      );
}
```

Результаты выполнения:

The square of 0 is 0
The square of 7 is 49
The square of 8 is 64
The square of 9 is 81
……

https://docs.microsoft.com/ru-ru/dotnet/api/system.threading.tasks.parallel.for?view=net-6.0

Parallel.ForEach

```
static  ParallelLoopResult  ForEach<TSource>(
        IEnumerable<TSource> source,
        Action<TSource>  body)
```

Параметры Parallel.ForEach:
TSource – тип элементов - объектов в коллекции;

source – коллекция объектов типа TSource.

body – лямбда-выражение.

Библиотечный обобщенный делегат:
public delegate void Action<T>(T obj)

https://docs.microsoft.com/ru-ru/dotnet/api/system.threading.tasks.parallel.foreach?view=net-6.0

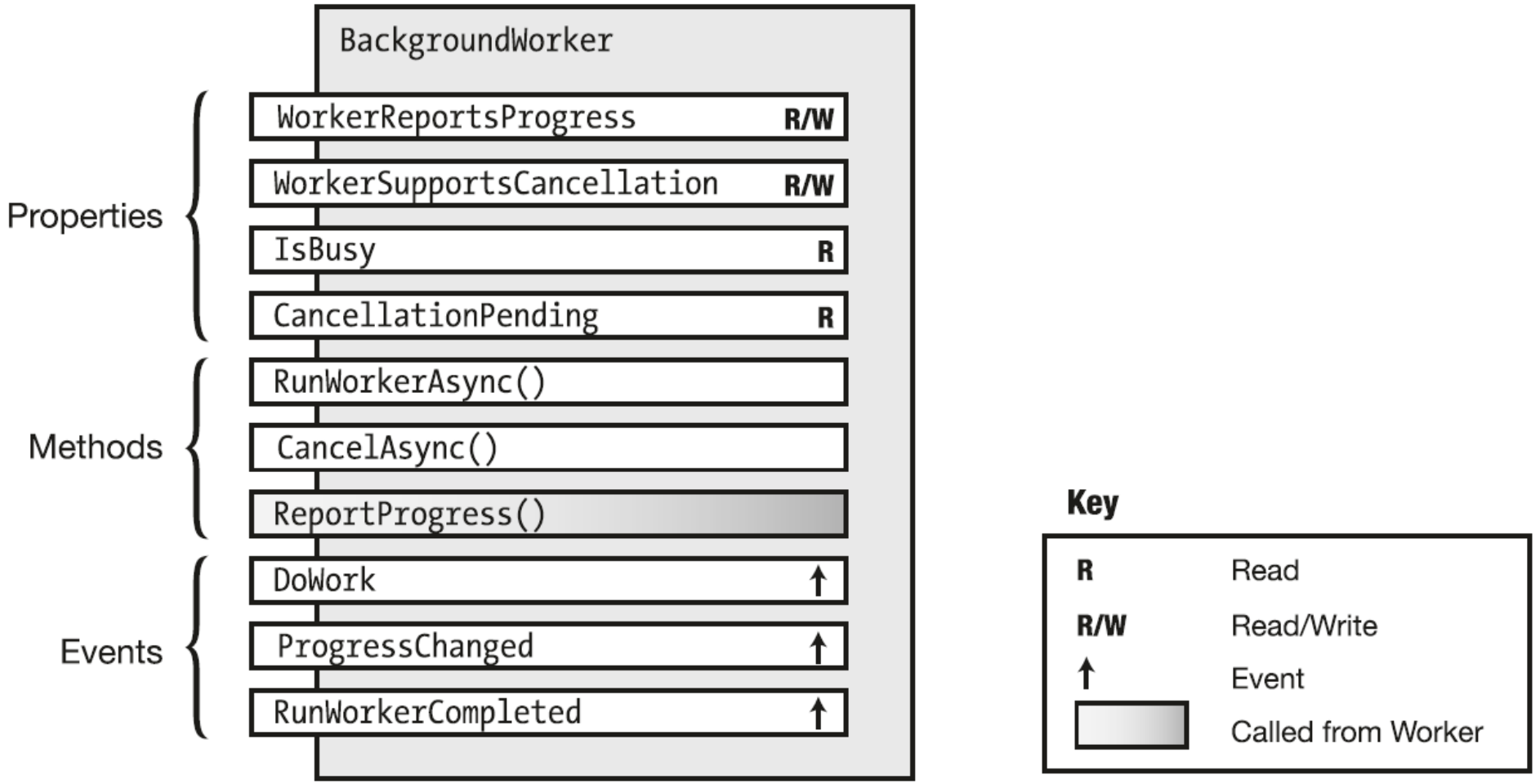
Parallel.ForEach

```
public static void Main02ForEach()
{
    string[] squares = new string[]
    { "We", "hold", "these", "truths", "to", "be",
"self-evident", "that", "all", "men", "are",
"created", "equal" };
    Parallel.ForEach(squares,
        i => Console.WriteLine(string.Format("{0}
has {1} letters", i, i.Length)));
}
```

Результат:
"We" has 2 letters
"equal" has 5 letters
"truths" has 6 letters
"to" has 2 letters
"be" has 2 letters……

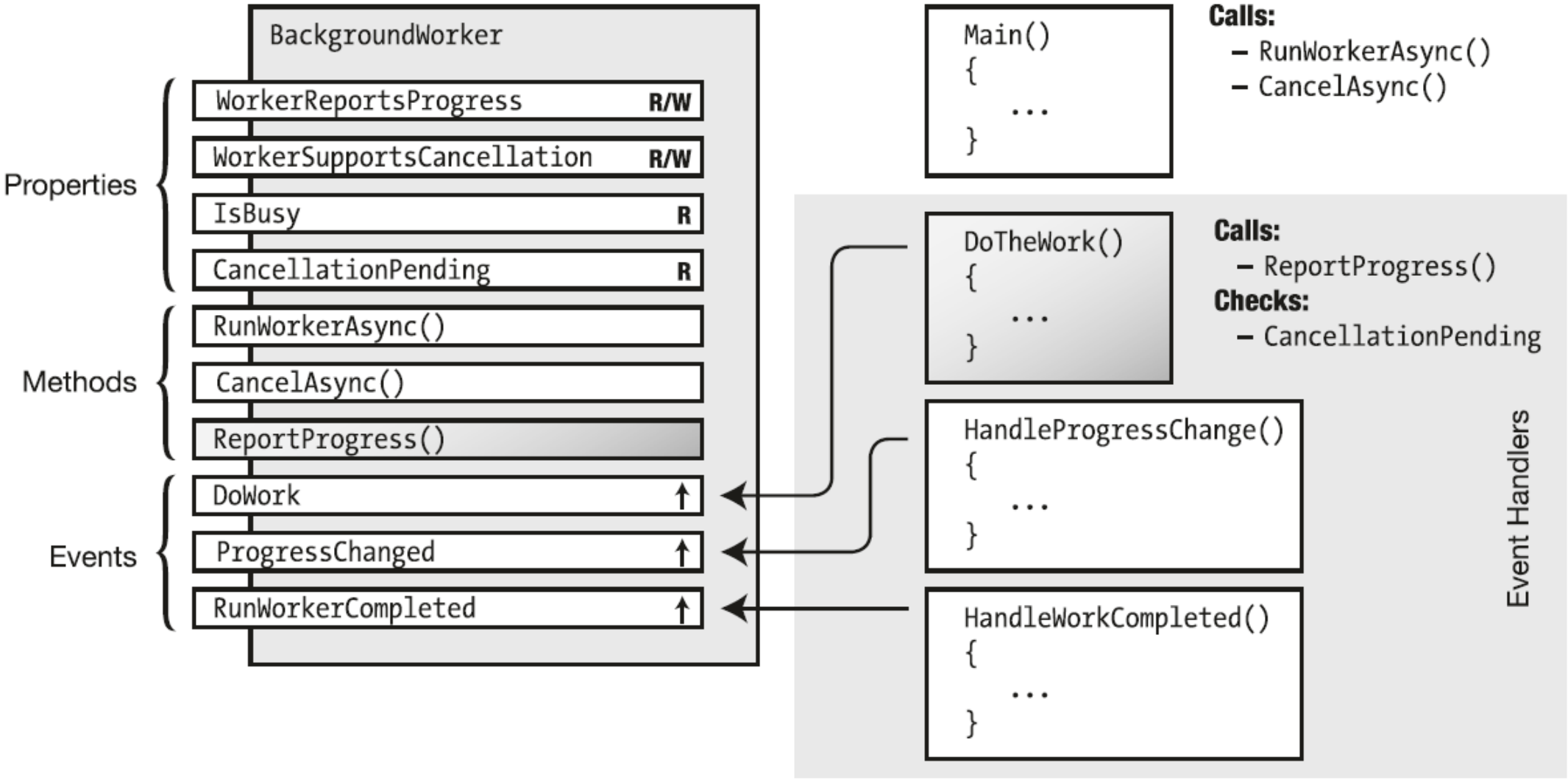
Класс BackgroundWorker

Содержится в пространстве имен System.ComponentModel



https://docs.microsoft.com/ru-ru/dotnet/api/system.componentmodel.backgroundworker?view=net-6.0

Программа с объектом класса...

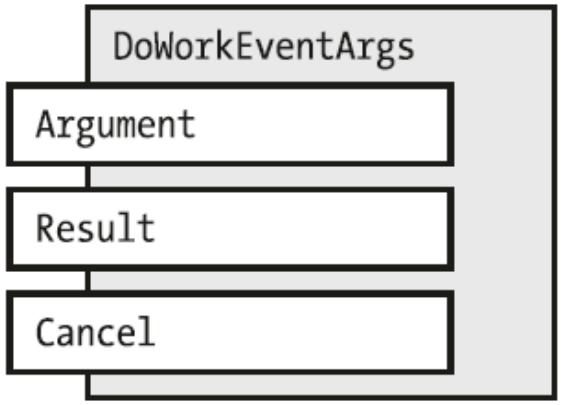


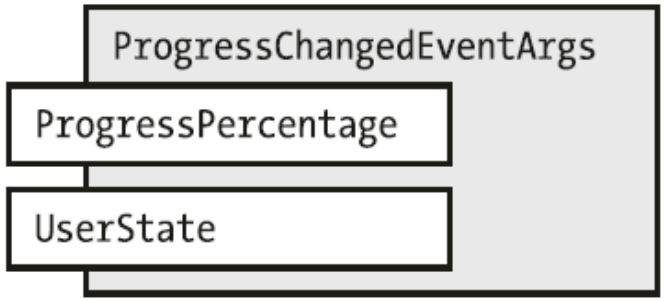
Делегаты событий для класса BackgroundWorker

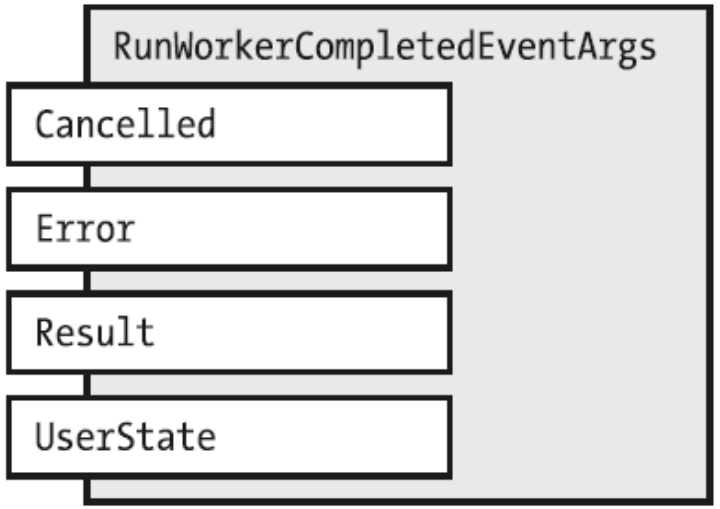
```
void DoWorkEventHandler ( object sender,
        DoWorkEventArgs e )

void ProgressChangedEventHandler ( object sender,
        ProgressChangedEventArgs e )

void RunWorkerCompletedEventHandler ( object sender,
        RunWorkerCompletedEventArgs e)
```

Классы, производные от EventArgs

- DoWorkEventArgs;
- ProgressChangedEventArgs;
- RunWorkerCompletedEventArgs.







Консольный пример BackgroundWorker

```
using System;
using System.ComponentModel; // BackgroundWorker
using System.Threading;

namespace DemoLect_MultiThreading {
    public class BackgroundWorkerDemo {

        public static void Main() { // Основной поток
            BackgroundWorker bgw = new BackgroundWorker();
            bgw.WorkerReportsProgress = true;
            bgw.DoWork += (sender, args) => {
                for (int n = 0; n++ < 50;) {
                    Thread.Sleep(50);  // Задержка потока
                    Console.Write("w");
                    if (n % 5 == 0)
                        bgw.ReportProgress(2 * n);
                }
            };
            bgw.ProgressChanged += (sender, args) => {
                Console.Write(args.ProgressPercentage + "%");
            };
```

Консольный пример BackgroundWorker (продолжение)

```
            bgw.RunWorkerCompleted += (sender, args) => {
                Console.WriteLine("\r\nCancelled!");
            };

            Console.WriteLine("Основной поток выводит точки!");
            bgw.RunWorkerAsync();   // Фоновый поток запущен

            while (true) {   // Цикл основного потока, Ctrl + C для прерывания
                Console.Write(".");
                // Искусственная задержка основного потока:
                for (int k = 0; k < int.MaxValue / 300; k++) ;
            }
        }
    }
}
```

Консольный пример BackgroundWorker (результат работы)

1) пример

Основной поток выводит точки!
..w.w.w.w.w10%.w..w.w.w.w20%.w.w.w..w.w30%.w.w.w.w.w40%
..w.w.w.w.w50%.w.w..w.w.w60%.w.w.w.w..w70%.w.w.w.w.w80%
.w..w.w.w.w90%.w.w.w..w.w100%
Cancelled!
.....^C

2) пример

Основной поток выводит точки!
...w...w..w...w..w10%...w..w...w...w.w20%..w..w..w...w..w30%...w..
w...w..w...w40%..w...w..w...w...w50%..w...w..w...w..w60%...w..w..
w..w...w70%..w...w..w...w..w80%...w..w...w..w...w90%...w..w...w..
w..w100%
Cancelled!
.....^C