Практикум: Създаване на нишки

В настоящия практикум ще създадем две приложения работещи с нишки.

Първо приложение: Два брояча наобратно

Ще създадем конзолно приложение, което работи с пускане на две нишки едновременно, които броят от 100 до 0 наобратно.

1. Създайте конзолно приложение ThreadExample
2. Кодът, който ще представлява нишката ще бъде изведен в отделен метод в рамките на Program.cs. Този метод ще носи името CountBackwards(). Кодът на метода е както следва:

```
static Random random = new Random();
2 references
static void CountBackwards()
{
    int waitTime = random.Next(1, 10) * 1000;
    Console.WriteLine($"Wait time: {waitTime}");
    for (int i = 10; i >= 0; i--) //shortcut: forr + 2 times TAB
    {
        Console.WriteLine(i);
        System.Threading.Thread.Sleep(waitTime);
    }
}
```

Кодът поставете между **class Program {** и **static void Main(string[] args) {**

Обяснение на кода:
Първо създаваме статичен обект от класа Random. Този обект ще ни бъде полезен за генерирането на случайни числа.

Вътре в метода започваме с генериране на случайно число от 1 до 10 - това ще бъде броят секунди, които програмата ще изчаква преди да отброи следващото число. Например, ако случайното число е 4, то програмата ще изведе 10, след което ще изчака 4 секунди и едва тогава ще изведе 9 и т.н.

Генерираното число се умножава по 1000, понеже времето на изчакване се измерва в милисекунди (1 секунда = 1000 милисекунди).

След изчислението извеждаме времето за изчакване на екран.

След това започваме да броим, използвайки for цикъл с обратно зададени стойности (началната стойност е 10 и броячът намалява тази стойност до достигане на 0). На всяко повторение от for цикъла, извеждаме текущата стойност на i. По този начин ни излиза на екран поредното отброяване. След това оставяме нишката да „спи" за съответния брой милисекунди (waitTime). Когато нишката „спи" тя е в режим на изчакване и ще продължи своето време след времето за заспиване.

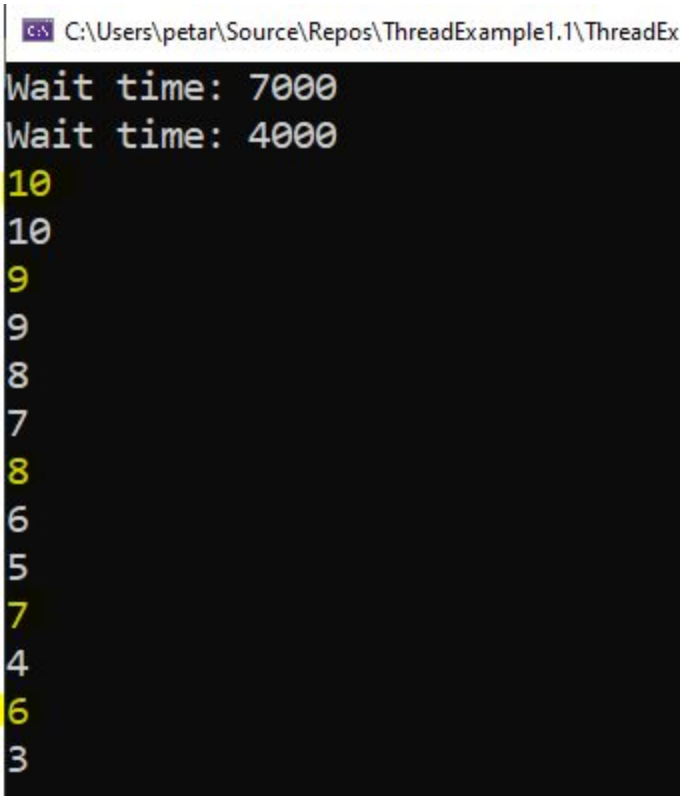
3. Код в Main метода.
   В главния метод трябва да създадем обекти за две нишки и да ги стартираме:

```
static void Main(string[] args)
{
    Thread thread1 = new Thread(CountBackwards);
    Thread thread2 = new Thread(CountBackwards);
    thread1.Start(); thread2.Start();
}
```

Тук създаваме обект от класа Thread (нишка). В конструктора подаваме желания метод, който нишката ще изпълнява. След това стартираме едната нишка и другата.

4. Изпълнение на програмата

В случая и двете нишки ще изпълняват един и същ метод - метода за броене на обратно **CountBackwards()**, но всяка нишка ще генерира свое случайно число, защото това са отделни извиквания на метода и в повечето случаи нишките ще имат различно време на изчакване. По този начин ще видите как числата от едното броене и от другото ще се преплитат.



Първата нишка изчаква 7 секунди между изхода на отделните числа, а втората - 4.
С жълто е маркиран изхода от първата нишка.

Второ приложение: Две състезаващи се нишки

Ще създадем конзолно приложение, в което две нищки ще трябва да натрупат поне 100 точки. Ще направим така, че в рамките на секунда всяка от нишките да увеличава точките си със случайно число. Победител е първата нишка, която е достигнала до 100 точки.

1. Ще създадем генератора на случайни числа - той ще ни потрябва по-късно:
   ```
   static Random random = new Random();
   ```
2. Създайте метод Player1(), който ще представлява първата нишка:

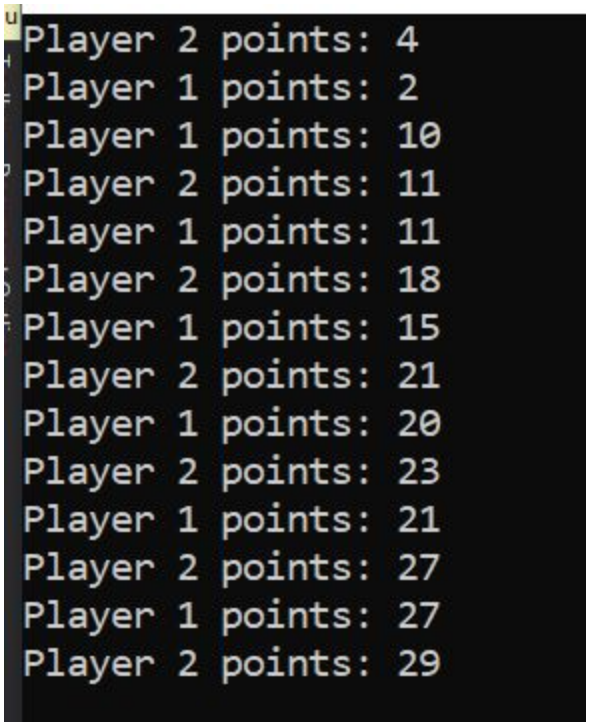
```
static void Player1()
{
    int currentPoints = 0;
    int endPoints = 100;
    while (currentPoints <= endPoints)
    {
        currentPoints += random.Next(1, 10);
        Console.WriteLine($"Player 1 points: {currentPoints}");
        System.Threading.Thread.Sleep(1000);
    }
    Console.WriteLine("Player 1 Finished!");
}
```

Обяснение на кода:
Играчът започва с 0 точки. Докато неговите точки са по-малки от задаения краен брой точки, неговите точки се увеличават със случайна стойност между 1 и 10, след което изчакваме секунда преди следващото повторение. Когато цикълът приключи, знаем, че и играчът финишира, съответно нишката приключва.

3. Създайте аналогичен метод Player2() - може да копирате по-горния метод и да направите промени по него.
4. В рамките на главния метод, създайте обектите от класа Thread и им посочете съответните методи, които трябва да се изпълняват от нишките. Не забравяйте да стартирате нишките след това.

```
static void Main(string[] args)
{
    Thread player1 = new Thread(Player1);
    Thread player2 = new Thread(Player2);
    player1.Start(); player2.Start();
}
```

5. Нека състезанието да започне :)



Автор: Петър Р. Петров, учител по професионална подготовка в ПГЕЕ „Константин Фотинов”