**54. Генетични (пораждащи )типове в обектното програмиране. Синтаксис. Начин на обработка в .NET среда. Разлика с шаблонизирани типове.**

**1.Същност на генетичните типове**

-Целта е сходна на целите на ООП – ***algorithm reusing*** . Механизмът е въведен в CLR на .NET

-Реализациите да се отнасят за обекти от различен тип;

-Може да се създаде :

***‘генетичен референтен тип’*** ;

***‘генетичен стойностен тип,***

***‘генетичен интерфейс’***

***‘генетичен делегат’;***

***‘генетичен метод’***.

**2.Синтаксис на генетичен тип:**

*public class List<T> : IList<T>, ICollection<T>, IEnumerable<T>, IList, ICollection, IEnumerable*

*{ public List();*

*public void Add(T item);*

*public void Sort( IComparer<T> comparer);*

*public T[] ToArray(0);*

*….*

*public Int32 Count {get;}*

*…*

*}*

**3.Начин на обработка:**

За да се поддържат генетични имплементации, към .NET се добавиха:

1.Нови IL инструкции, четящи конкретния тип на аргуманта;

2.Метаданновото описание се обогатява с описание на типа на параметрите;

3.Променя се синтаксисът на C#, Visual Basic и т.н.

4.Променят се компилаторите;

5.Променя се JIT компилаторът, така че да генерира ‘native code’ за всяко повикване с конкретен тип на аргумент.

***4.Разлика с шаблонизирани типове***

-Разработчикът не е нужно да притежава сорса на генеричния алгоритъм ( за разлика от C++ templates или Java generics) за да прекомпилира.CLR средата генерира ‘native code’ за всеки метод, първият път когато методът

се повика с указан тип данни. Това разбира се, увеличава размера на кода

(при генерични реализации), но не и производителността

При шаблоните, компилаторът генерира separate source-code functions (named specializations) при всяко отделно повикване на ф-ия шаблон или инстанция на шаблонизиран клас.

-ясен код: рядко се налагат tape casts;

-Подобрена производителност: преди генетиците, същото се постигаше с използване на Object типа. Това налага непрекъснато пакетиране (boxing), което изисква памет и ресурс, форсира често включване на с-мата за garbage collection. При генетичните алгоритми няма пакетиране.Това подобрява десетки пъти производителността.