Compilation (before executing) -> Just in time (while executing). C# compiler from source to intermediate, JIT from intermediate to machine

Common Language Runtime (CLR); Структуры – value, а не reference, она не наследуется, но интерфейсы +, а так тоже что и классы. Immutable – все изменения создают новый объект.

*Array* – один блок в памяти, все одного размера, легко по индексу, сложно менять размер (новый). *List* – легко меняет размер (в 2 раза, сначала 4), легко с концом работать, неудобно работать с началом. *Linked List* – не один блок памяти, легко работать со след и раньшим, но неудобно о индексу. *Dictionary* (Associative Container) – key must be immutable, быстрый поиск по ключу. *Queue* – при добавлении добавляет «в конец». *Stack* – добавляет в начало. Queue and stack - в очереди - объекты типа object. *Enqueue()* - помещает элемент в конец очереди. *Dequeue()* - достает из очереди первый элемент. *Contains()* - проверяет имеется ли данный объект в очереди. *Count* - свойство, содержащее количество элементов в очереди. *Peek()* - возвращает первый элемент в очереди, но не удаляет его. От очереди стек отличается лишь тем, что первым "достается" не тот, кто первым пришел, а тот, кто пришел последним. *Push()* - помещает элемент в стек. *Pop()* - достает элемент из стека - последний добавленный. *Peek()* - возвращает верхний элемент стека. *Contains()* - проверяет содержится ли элемент в стеке. *Count* - свойство содержащее количество объектов в очереди.

Delegates – анонимный для каких-то простых(типа лябда-выражений штук). Delegate variable отличается от event тем, что: переменная может быть объявлена в методе и передана как параметр, event вызывается только внутри класса где объявлен, вызывается тоже.

В интерфейсах могут быть свойства (IEnumerable<Employee> Employees {get;}), методы (void SaveAll(string a);), события(event Action<List<Students>> Lol;), indexers(типо массива ключей в словаре).IEnumerable (iterate over container), ICollection(IEnumerable + add,remove, number of items), IList(ICollection + get by index), IDisposable – от которого избавиться можно

Is a – наследование, has a – композиция (типо в классе много полей). Interface is about multiple inheritance, abstract class is about one base class. При наследовании все имена свойств, методов, полей остаются. В абстрактном классе можно поля, свойства (с нормальным гет сетом), абстрактные и виртуальные методы и т.д.

Сначала base конструктор, потом свой; event из base в наследниках не вызывается. Нельзя создавать экземпляр абстрактного класса, надо переписать в наследниках. Sealed classes не наследуются

Singleton – чтобы создавался лишь один экземпляр класса на всю программу.

Finalizers are special methods that are called when the garbage collector cleans the object; They are used to make sure unmanaged resources are released in case Dispose is not called; 

Singleton:   
public class Settings   
{   
private Settings() {} //private Constructor   
private static Settings \_default;   
public static Settings Default // static property   
{   
get   
{   
if (\_default == null)   
\_default = new Settings();   
return \_default;   
}   
}   
}