															a value that	with an object or	unique signature	provides access to a characteristic of an object or a class / setter and getter /		u sig
															r	men				
	definition define operators	base class (this class inherits from ohter)	derived (other classes can inherit this one)	what members inherits from his base class	inherited members are accesible	type of inheritance	Interfaces (can implement)	instantiated	declaration modifiers	executable code	constitutent types (types used in member declarations)	general	members implicicity made	nested class implicity made	constants	fleids	methods	properties	events	indexers
class	reference type	can't be System.{Array, Delegate, Enum, ValueType}, and have only one	yes	All except instance constr., finalizers, and static constr.	it depends on accesibility modifier	transit	several, and re-implement	yes	PB, PT, INPT, IN, PV (D) *	inside function members (*)	property event	allowed			NW, PB, PT, IN, PV	NW, PB, PT, IN, PV, ST, RO, VL	NW, PB, PT, IN, PV, ST, VR, SD, OV, AT, EX	NW, PB, PT, IN, PV, ST, VR, SD, OV, AT		
abstract class	can't be instantiate d	yes	can be derived to abstract and non abstract				several, re- implement, and map intf. methods onto abstract	NO		yes		abstract members are permitted (but not required)				can have variables and values but they must be nul (*)	yes			ye
static class	NO	NO	NO				NO	NO		yes		only ST and NOT PTIN	not	ning						
sealed class	can't be derived	yes	can't be					yes												
struct	value type	not support definition , but inherits from System.ValueT ype	HOL				several, and re-implement	N.A.	NW, PB, PT, IN, PV						yes	yes	yes	yes	yes	ує
interface	define a contract	Base interfaces!					can inherit multiple base- interfaces		NW (only in nested) PB, PT, INT, PV *	NO		cannot be declared AT, PB, PT, IN, PV, VR, OV, ST					yes, but without implementin g them		yes	ye

Every only one instance storage has a not present in the diagram to reduce the amount of text. I designed this diagram just to help myself in the migral separate unique name / unique name / unique signature signature instances fields Every only one instance instance instance storage has a not present in the diagram to reduce the amount of text. I designed this diagram just to help myself in the migral not present in the diagram to reduce the amount of text. I designed this diagram just to help myself in the migral not present in the diagram to reduce the amount of text. I designed this diagram just to help myself in the migral not present in the diagram to reduce the amount of text. I designed this diagram just to help myself in the migral not present in the diagram to reduce the amount of text. I designed this diagram just to help myself in the migral not present in the diagram to reduce the amount of text. I designed this diagram just to help myself in the migral not present in the diagram to reduce the amount of text. I designed this diagram just to help myself in the migral not present in the diagram to reduce the amount of text. I designed this diagram just to help myself in the migral not present in the diagram to reduce the amount of text. I designed this diagram just to help myself in the migral not present in the diagram to reduce the amount of text. I designed this diagram just to help myself in the migral not present in the diagram to reduce the amount of text. I designed this diagram just to help myself in the migral not present in the diagram to reduce the amount of text. I designed this diagram just to help myself in the migral not present in the diagram to reduce the amount of text. I designed this diagram just to help myself in the migral not present in the diagram to reduce the amount of text. I designed this diagram just to help myself in the migral not present in the diagram to reduce the amount of text. I designed this diagram just to help myself in the migral not present in the diagram to															umn. Some obvious things are f in the migration from C++ and								
operators	instance constructors	constructor	finalizers	static constructors	types (nested)	can declare virtual members (polymorphic behavior)	static member	instance member	constants and nested types	partial modifier allowed	sealed modifier	abstract modifier	public modifier	protected modifier	protected internal	internal modifier	private modifier	static modifier	new modifier / hides inherited member	can supply type- parameter-list	NOTES (things to keep in mind)		
					PB, PTIN, PT, IN, PV (D)					yes									can have (but hidden members will not be removed)			constant or type declaration implicitly declares a static member	
yes	yes		yes	yes		Yes, methods, properties, indexers, and events	i				NO											2. Explicit interface member implementations cannot be abstract, but explicit interface member implementations are of course permitted to call abstract methods	
	doesn't have								implicity static		N	0						must have	impossible	no	implicity inherits from type object	2. shall only have static members, and shall not have protected internal	
												NO									The sealed modifier is used to prevent derivation from a class		
yes	cannot define, but implicitly has a parameterless one	yes, can declare having paramete rs	NO!	yes	PB, IN, PV (D)						nc permi alwa impl	itted, ays		N.A. S inheritar supp	nce isn't						1. the simple types provided by C#, such as int, double, and bool, are, in fact, all are structs types	2. values can't be null	
																					member acces casted to solve ambiguity		
					Only most	important feat	tures are sh	own															