

## C# Programming Language

ان لغة السي شارب تتبع مجموعة لغات .Net Frame Work وهذه البيئه  
 تحتوي على عدة لغات لكن كل منهم له الكود الخاص به وتوجد بها لغه وسيطيه  
Microsoft Intermediate Language (MSIL) وبالتالي هي لغه لا تعتمد  
 على الاله فهي تعمل على اى جهاز

ان C# تضم مميزات VB - C++ بما لها من سهولة التعامل فى الواجهات  
 الرسوميه بالاضافه الى قوة البرمجه .

ان الفارق الجوهرى بين C# و C++ هو كالتالى :

C++	C#
هى لغة تستطيع بها ان تستخدم مفهوم البرمجه كائنية التوجه	هى لغة مبنيه بالاساس على مفهوم البرمجه كائنية التوجه او <b>Object Oriented</b>

الهيكـل التـنظيـمـي لبرنامـج C# :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

**namespace**  
الاساسيه والتي تستدعي عند كتابة اي برنامج

```
namespace ConsoleApplication2
{
    class Program
    {
        static void Main(string[] args)
        {
            Main()
        }
    }
}
```

**class**  
الرئيسي للبرنامج ويوجد بداخله الداله الرئيسيـه  
يكتب هنا داخل الداله الرئيسيـه الكود الذي سوف ينفذ فهو يبدء التنفيذ من الداله  
**Main()**

نأتى لتفسير التكوين الرئيسي لبرنامج C#

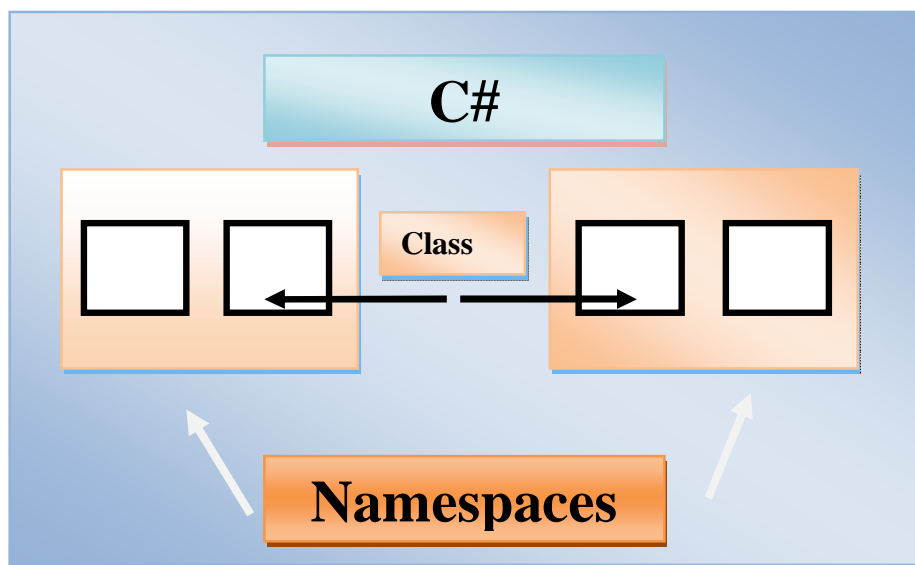
Using هي كلمه محجوزه داخل اللغه وتستخدم للنداء على اسم معين ليصبح

جزء من البرنامج

Namespace ويعرف مجازا بفضاء الاسماء وهو عباره عن مجلد يحتوى

بداخله على Classes جاهزه للعمل عليها ولها اسماء محدده

ونستطيع انه نمثله بالشكل التالي :



**Classes** هي وحدات محجوزه داخل اللغة وتحتوى على جزئين رئيسيين:

1- **Data Member** وهي المتغيرات والتي تحتوى على بيانات

2- **Function Member** دوال تقوم بتنفيذ عمليات على الدوال المخزنه

✚ ان اى برنامج داخل C# قد يحتوى على اكثر من namespace على الاقل

لابد من وجود واحد فقط وقد يحتوى على اكثر من **Class** لكن على الاقل

يجب ان يحتوى على **Class** واحد

✚ وبعد **System** هو namespace الرئيسى بصفته يحتوى على classes

تحتوى على دوال التوصيف للدخل والخرج داخل اى برنامج

ان وظيفة الداله الرئيسييه Main() ان Compiler يبدء دائما بتنفيذ خطوات

البرنامج من عندها ايا كان موضعها ولا بد ان تكون موجوده داخل class

الرئيسي للبرنامج والذي يعطى اسم الافتراضى Program ومن الممكن

وضع تلك الداله داخل اى class وسعتبره انه الرئيسى الذى يحتوى على

تلك الداله

ان لغة C# حساسه لحالة الاحرف بمعنى انه يجب ان يراعى فيها capital

او small

شرح الاوامر البرمجييه داخل لغة C# :-

اوامر الطباعه :

هناك امران اساسيان داخل C# وهما كيفية التعامل مع الدخول والخرج والذي

لاغنى عنهم داخل اى برنامج وهما ممثلان فى دالتين يتبعان نفس class

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication2
{
    class Program
    {
        static void Main(string[] args)
        {
            string myname;
            myname = Console.ReadLine();
            Console.WriteLine("hello", myname);
            Console.ReadLine();
        }
    }
}
```

الداله ReadLine()  
خاصه باستلام المدخلات داخل اى برنامج

الداله WriteLine()  
خاصه باظهار المخرجات

دوال الدخل والخرج والتي تتبع جميعها التصنيف Console	
هذه الدالة تقوم بطباعة الخرج ويظل المؤشر في مكانه	<b>Write()</b>
هذه الدالة تقوم بطباعة الخرج وينتقل المؤشر الى سطر جديد اسفل الخرج	<b>WriteLine()</b>
تقوم هذه الدالة بقراءة المدخلات ودائما ما تعتبر لغة السي شارب ان جميع المدخلات على انها نصوص لذلك يجب تحويل المدخلات بعد قراتها	<b>ReadLine()</b>

معامل تنظيم ظهور الخرج على الشاشة وما يعرف Replacement Operator

```
string myname;
string hellomessage;
myname = Console.ReadLine();
hellomessage = "hello mr";
Console.WriteLine("hello{0}, {1}", myname, hellomessage);
Console.ReadLine();
```

**Replacement Operator**

والذي يشير الى ترتيب ظهور قيم الخرج على الشاشة حيث يبدأ بالرقم 0 ثم 1 وهكذا

انواع البيانات داخل لغة C# :

انواع البيانات داخل السي شارب وسعة كلا منها		
النوع	فيما يستخدم	سعته
Byte	للارقام	0 الى 8 bits
Short	للارقام	16 bits الى 2byte
Int	للارقام	32byte to 4byte
Char	للحروف	8byte for one char
Float	للكسور	4byte
Boolean	للقيم المنطقية	True or false
String	للتنصوص	Stream open of char
Double	للارقام التي تحتوى علامات عشرية	8byte

القاعده العامه لتعريف المتغيرات داخل C# :

```

1 Variable declaration
2 syntax : Var_type var_name = initial_value;
3
4 example : int x = 5;
5           string y = "ali";
6           char c = "m";
7           double s = 34.23;
8           float t = 2.3;
9           and so on

```

قيمته ابتدائيه

اسم المتغير نوع المتغير

داخل لغة السي شارب لابد من الاعلان عن المتغيرات صراحة وتعريفها قبل استخدامها او اعطاؤها اي قيمه

```

using System.Text;

namespace ConsoleApplication2
{
    class Program
    {
        static void Main(string[] args)
        {
            string myname;
            string hellomessage;
            myname = Console.ReadLine();
            hellomessage = "hello mr";
            Console.WriteLine("hello{0}, {1}", myname, hellomessage);
            Console.ReadLine();
        }
    }
}

```

تم تعريف المتغيرات قبل استخدامها او اعطاؤها قيم  
يأخذ المتغير القيمة من المستخدم اثناء تنفيذ البرنامج  
اعطاؤه قيمة ابتدائية

دوال التحويل داخل لغة السي شارب :

قلنا سابقا ان السي شارب تعتبر جميع المدخلات على انها نصوص فكيف اذا ما تم

ادخال ارقام او انواع اخرى فهناك دوال للتحويل بين انواع البيانات

دوال التحويل داخل السي شارب

Parse

Convert

ولدينا الامثلة التاليه :

```

string myname;
string hellomessage;
int myage;
myname = Console.ReadLine();
myage = Int32.Parse(Console.ReadLine(myage));
myage = Convert.ToInt32(Console.ReadLine(myage));

```

تحويل القيم قبل استخدامها بالطريقتين

تستطيع الان استخدام نفس الدالتين لتحويل باقى انواع البيانات حيث انك مضطر

لاستخدامها فى جميع الاحوال

انواع المعاملات داخل السى شارب :

Arithmetic operators	المعاملات الرياضيه
Multiplication	*
Addition	+
Subtraction	-
Module	%
Dividing	/
Increment by 1	++
Decrement by 1	--

## Competitive Operators

معاملات المقارنه

Smaller than	<
--------------	---



Greater than	>
Equal	==
Smaller than or equal	<=
Greater than or equal	=>
Not equal	!=

### Logical operator

### المعاملات المنطقية

And	&&
Or	
not	!

### معاملات المساواة

### Assignment

Assignment plus	+=
Assignment minus	-=

ان جميع المعاملات التي تم سردها في الجداول السابقة لها الاستخدام الاوسع

داخل السي شارب وتدخل في معظم العمليات الحسابية والمنطقية

# Flow Control Statments

استخدام الدوال الشرطيه داخل اللغه **Flow Control – statements** :

الدوال الشرطيه من الدوال الواسعة الانتشار داخل لغات البرمجه بصفه عامه

وذلك لاهميتها الكبيره فى التحكم فى سير البرنامج ولدينا العديد منها داخل

السى شارب .

القاعده الشرطيه IF

القاعده العامه لها :

```

1 IF Statment : syntax : if (condition)
2                               {
3                               statments ;
4                               }
5

```

بمعنى لو تحقق الشرط سوف يتم تنفيذ الجمل البرمجيه التى تتبع القاعده الشرطيه

```

static void Main(string[] args)
{
    int mynumber;
    mynumber = int.Parse(Console.ReadLine());
    if (mynumber == 10)
    {
        Console.WriteLine("we are win the race");
    }
}

```

الشرط المطلوب تحقيقه

الجمله التى ستنفذ اذا تحقق الشرط

القاعده الشرطيه IF ELSE :

القاعده العامه لها :

```

1 IF Elase Statment : syntax : if (condition)
2                               {
3                               statments ;
4                               }
5                               Else
6                               {
7                               statments;
8                               }

```

بمعنى انه اذا تحقق الشرط سيتم تنفيذ الجمل التابعه IF واذا لم يتم تحقيق الشرط سيتم

تنفيذ الجمل التابعه ELSE

```

static void Main(string[] args)
{
    int mynumber;
    mynumber = int.Parse(Console.ReadLine());
    if (mynumber == 10)
    {
        Console.WriteLine("we are win the race");
    }
    else
    {
        Console.WriteLine("we are lose");
    }
}

```

الجمله الاخرى في حالة عدم تحقيق الشرط

القاعده الشرطيه NESTED IF : 

القاعده الشرطيه المتداخله والصوره العامه لها :

```

1  Nasted IF Statment : syntax
2
3  If (condition1)
4      If (condition2)
5          {
6              Statements1;
7          }
8  Else if (condition3)
9      IF (condition4)
10         {
11             Statments2;
12         }
13 Else
14     Statments3;
15 Else
16     Statments4;

```

لاحظ دائما فى حالة if المتداخله فان else تتبع اقرب if لها

```

{
static void Main(string[] args)
{
    int mynumber;
    mynumber = int.Parse(Console.ReadLine());
    if (mynumber <= 10)
        if (mynumber >= 5) فى حالة تحقيق الشرطين تطبع الجمله الاولى
        {
            Console.WriteLine("we win the race");
        }
        else فى حالة عدم تطبيق الشرط الثانى بعد الاول تطبع الجمله الثانيه
        {
            Console.WriteLine("we are lose");
        }
    else فى حالة عدم تطبيق اى من الشرطين تطبع الجمله الثالثه
    {
        Console.WriteLine("wrong value");
    }
    Console.ReadLine();
}

```

## القاعده الشرطيه SWITCH CASE :

القاعده العامه لها :

```

1 Switch Case statment : syntax
2 Switch (variable or expertion)
3 {
4 Case 1: {statements};
5 Break;
6 .
7 .
8 Case n :{ statements};
9 Break;
10
11 Default :{ statements};
12 Break;
13 }

```

يتم تعريف المتغير ثم في كل مره اعطاؤه قيمه معين مع كل جمله Case وتنفيذ الجمله المطلوبه اذا ما تساوت هذه القيمه مع اى حاله من تلك الحالات ولا حظ انه بعد كل حاله لابد من وضع الكلمه المحجوزه Break لتفصل كل حاله عن الاخرى ثم فى النهايه وضع الحاله Default والتي سوف تنفذ جملتها اذا لم تتحقق اى من الحالات السابقه

```

string thename;
thename = Console.ReadLine();
switch (thename)
{
    case "ali":
        Console.WriteLine("this is my name");
        break;
    case "mohamed":
        Console.WriteLine("this is my brother name");
        break;
    case "basem":
        Console.WriteLine ("this is my frend name");
        break ;
    default :
        Console.WriteLine("i dont know that name");
        break;
}
Console.ReadLine();

```

كل حاله منفصله عن الاخرى

ستنفذ هذه الجملة اذا لم يتوفر اى شرط من الشروط السابقة

## Loops statments

الجمل التكراريه : LOOPS

تستخدم هذه الجمل فى تكرار جزء معين من البرنامج اذا ما تحققت شروط معينه

حيث يكون التكرار محدود وله ضوابط وليس الى مالانهايه ولها انواع داخل لغات

البرمجه وسنوضح الان كل جملة وطريقة عمله

الجمله التكراريه : For Loop

القاعده العامه لها :

```

1 For Statment :syntax
2 For (initialize; condition; update);
3 {Statements};

```

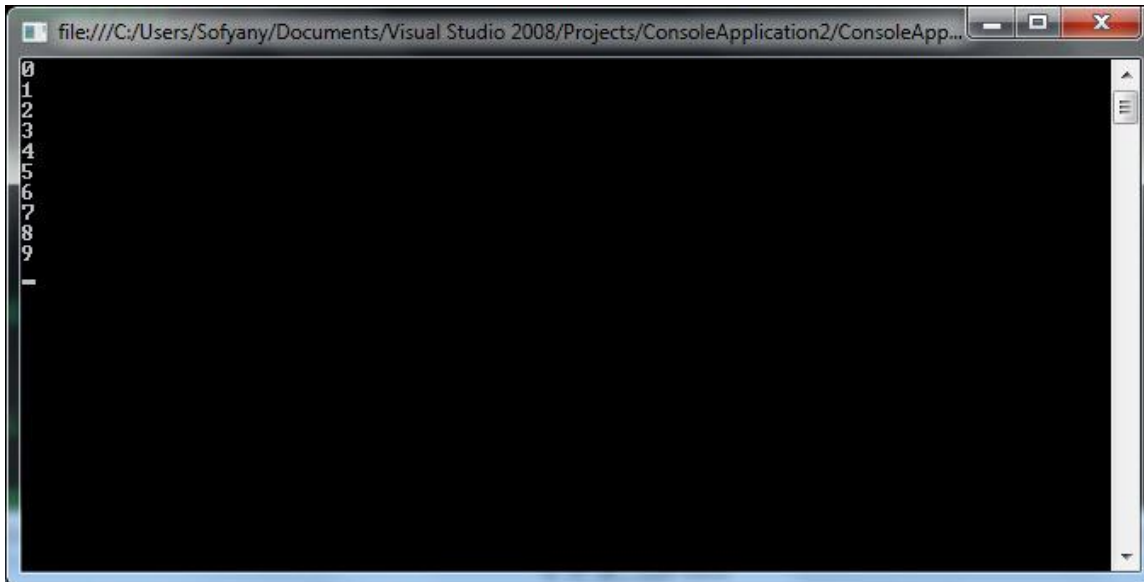
```
using System.Text;
namespace ConsoleApplication2
{
    class Program
    {
        static void Main(string[] args)
        {
            for (int i = 0; i < 10; i++) ;
            int k;
            for (k = 0; k < 10; k++)
            {
                Console.WriteLine(i);
            }
        }
    }
}
```

اولا يوضع القيمة الابتدائية ثم الشرط ثم التحديث بحيث لا يزيد التحديث عن الشرط المحدد ولاحظ انه يمكن تعريف المتغير داخل جملة التكرار

الحاله الاخرى تعريف المتغير مسبقا ثم بناء الجملة عليه

```
static void Main(string[] args)
{
    int i;
    for (i = 0; i < 10; i++)
    {
        Console.WriteLine(i);
    }
    Console.ReadLine();
}
```

الجملة التي ستنفذ في كل مره يتحقق فيها الشرط وهي طباعة قيمة المتغير



## القاعده الشرطيه While Loop :

وهذه القاعده تؤكد على استمرار تنفيذ جزء معين من الكود طالما ان الشرط مازال

متحققا والقاعده العامه لها كالتالي

```

1 While Loop statment :syntax
2
3     Initialize;
4 While (condition)
5 {
6     Statements;
7     Update;
8 }
```

```

static void Main(string[] args)
{
    //القيمه الابتدائيه
    int i = 0;
    //الشرط
    while (i < 10)
    {
        Console.WriteLine(i); //الجملة
        i++; //التحديث وبعد كل مره
    } //يعود ليختبر الشرط مع
    Console.ReadLine(); while
}
}
```

النتاج



## القاعده الشرطيه DO While :

وهذه القاعده لها خصوصيه حيث انه يتم تنفيذ الكود اول مره قبل اختبار الشرط  
بمعنى انه اول مره سيتم التنفيذ حتى لو يتحقق الشرط وبعد ذلك يختبر الشرط كل  
مره . القاعده العامه لها :

```

1 do while statment : syntax
2 Initialize;
3 do
4 {
5 Statements;
6 Update;
7 }
8 While (condition);

```

لاحظ ان الشرط وضع فى نهاية الجملة حتى يتم التنفيذ فى المره الاولى اذا لم يتحقق

لكن بعد ذلك فانه لن يتم النفيذ الا اذا تحقق الشرط

```

static void Main(string[] args)
{
    int i = 0;
    do
    {
        Console.WriteLine(i);
        i++;
    }
    while (i < 10);

    Console.ReadLine();
}

```

المثال الاول سوف يطبع  
القيمه الاولى ثم يختبر  
الشرط بعدها

0  
1  
2  
3  
4  
5  
6  
7  
8  
9

الخرج

```
static void Main(string[] args)
{
    int i = 11; لاحظ هنا ان القيمة الابتدائية
    do          تخالف الشرط
    {
        Console.WriteLine(i);
        i++;
    }
    while (i < 10);

    Console.ReadLine();
}
}
```

11

ومع ذلك قام بطباعة  
القيمة الاولى وتوقف عند  
ذلك لانه اختبر الشرط  
في المره الثانيه ووجده

✚ جملة الخروج عن الحلقة التكرارية عند الوصول الى نقطه معينه break :

والتي تقوم بانهاء الحلقه عند الوصول الى نقطه معينه او تحقيق شرط معين

القاعده العامه لها : تستخدم داخل اى من الجمل التكراريه السابقه ودائما توضع بعد اختبار الشرط كل مره ولدينا المثال التالي :

```
static void Main(string[] args)
{
    int i;
    for (i = 0; i < 10; i++)
    {
        if (i == 5) سيتم كسر الحلقه عند الوصول الى العدد 5
        break;
        Console.WriteLine(i);
    }

    Console.ReadLine();
}
```

```
0
1
2
3
4
```

تم الخروج من الحلقه عند الوصول الى الرقم 5 حيث قام بالعد حتى الرقم 4 فقط حسب شرط الخروج من الحلقه

🚩 امر الاستمرار داخل الحلقة عند تحقيق شرط معين Continue :

هذا الامر يخرج من الحلقة ثم يعود اليها مره اخرى عند شرط معين بمعنى انه

يتخطى تنفيذ جزء معين من الحلقة ثم يعود لينفذ الباقي

ولدينا المثال السابق حيث سيتجاهل البرنامج طباعة الرقم 5 ثم يطبع باقى الارقام

القاعده العامه لها : انها توضع فى نفس مكان break

```
static void Main(string[] args)
{
    int i;
    for (i = 0; i < 10; i++)
    {
        if (i == 5)           عند الوصول الى 5 لن ينفذ
            continue;       الامر ويقوم بتنفيذ الباقي
        Console.WriteLine(i);
    }

    Console.ReadLine();
}
}
```

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2
```

# Compound Data: Arrays

1 – المصفوفات او Array :

يجب ان يكون كل عناصر المصفوفه لها نفس نوع البيانات وممكن ان تختلف فى القيم  
ويوجد لها نوعان المصفوفه احادية البعد او one –D والمصفوفه الثنائية البعد او

Tow – D

القاعده العامه لتعريف المصفوفه الاحديه :

- 1 Array Definition : syntax
- 2 Data type [] reference name = new data type [size];

```

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            int [] onearray = new int [5];
            Console.ReadKey();
        }
    }
}

```

نوع بيانات المصفوفه

اسم المصفوفه

حجم المصفوفه

ومن الممكن ان تعطيهها قيما ابتدائيه عند التعريف لكن يجب الاتزيد القيم عن حجم المصفوفه

```
class Program
{
    static void Main(string[] args)
    {
        int [] onearray = new int [5]{10,15,20,25,30};

        Console.ReadKey();
    }
}
```

القيم الابتدائيه من نفس نوع البيانات ولا تزيد عن حجم المصفوفه

ومن الممكن ايضا تحديد حجم المصفوفه عن طريق القيم الابتدائيه لكن ليست مستحبه

ومن الممكن اعطاء قيما ابتدائيه للمصفوفه من المستخدم وقت تنفيذ البرنامج

طريقة الاتصال مع المصفوفات الاحادية :

```
1 Write data to element in array :
2 Direct in code: array name [index of element] = value;
3 X [2] = 10;
4
5 write data from user on runtime:
6 X [2] = int.parse (Console.ReadLine ());
7
8 print element data :
9 Console.WriteLine(x [2]);
10
11 modifi on element data:
12 X [3] = x [3]*3;
```

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] x = new int[10];
            for (int i = 0; i < 10; i++)
            {
                Console.WriteLine("enter no :{0}", i);
                x[i] = int.Parse(Console.ReadLine());
            }
            Console.ReadKey();
        }
    }
}

```

سعة المصفوفة 10 لذلك لن يزيد الحلقة التكراريه عن ذلك

سوف يتم استلام بيانات المصفوفه من المستخدم وقت تنفيذ البرنامج

ومن المثال السابق سوف بتكملة البرنامج لحساب اكبر قيمه و اقل قيمه تم ادخالها من المستخدم كما يلي :



```
static void Main(string[] args)
```

```
{
```

```
    int i;
```

```
    int[] x = new int[10];
```

```
    for ( i = 0; i < 10; i++)
```

```
    {
```

```
        Console.WriteLine("enter no :{0}", i);
```

```
        x[i] = int.Parse(Console.ReadLine());
```

```
    }
```

```
    int max = x[0];
```

هنا تم تعريف متغيرين بالقيمة الصغرى والعظمى

```
    int min = x[0];
```

واعطائهم القيمة الموجوده بالعنصر الاول من

```
    for (int j = 0; j < 10; j++)
```

المصفوفه

```
    {
```

```
        if (x[j] < min)
```

تتم المقارنه العنصر الاول من المصفوفه بجميع العناصر الاخرى

```
            min = x[j];
```

ولرى من فيهم الاصغر من الاخر حتى يتم ايجاد الاصغر فيهم

```
        if (x[j] > max)
```

جميعا ثم بعدها نفس العمليه تتكرر من فيهم الاكبر من الاخر حتى

```
            max = x[j];
```

نجد الاكبر فيهم ثم نقوم بطباعة القيمتين على الشاشة

```
    }
```

```
    Console.WriteLine("the max no :{0}", max);
```

```
    Console.WriteLine("the min no :{0}", min);
```

```
    Console.ReadKey();
```

```
}
```

```
}
```



## : For Each .....Loop 🚩

هناك نوع اخر مهم جدا من الحلقات التكراريه وغالبا ما يستخدم مع المصفوفات

حيث يستخدم فى قراءة بيانات المصفوفه ولا يمكن استخدامه فى الكتابه اليها او

تغير قيمها

القاعده العامه لها :

```

1  Foreach statement : syntax
2  For each (<data type> <identifier> in array)
3  {Stmts};
4

```

```

static void Main(string[] args)
{
    int i;
    int[] x = new int[10];
    for ( i = 0; i < 10; i++)
    {
        Console.WriteLine("enter no :{0}", i);
        x[i] = int.Parse(Console.ReadLine());
    }
    foreach (int m in x) استخدام الحلقه التكراريه
        Console.WriteLine(m); foreach
    Console.ReadKey(); فى قراءة محتويات المصفوفه وطباعتها
}
}
}

```

: Tow – D Array 🇵🇸

المصفوفات ثنائية الابعاد هي عباره عن نوع من المصفوفات يشبه الجداول

حيث يتكون من صفوف واعمده ويكون الخلايا وهذه الخلايا هي عناصر

المصفوفه

التوصيف العام لها :

```

1 2-D array : syntax
2 Data type [,] arrayreferance = new data type [rows_no, columns_no];
3

```

For example:

```

1 int [,]A = new int [4,5] \\ decleration
2
3 intialize array
4
5 int [,] A = new int [1,2]
6 A[0,0] = 1
7 A[0,1] = 2; اعطانها قيم ابتدائيه
8
9 int [,] A = new int [3,4]
10 A[0,0] = 1;
11 A[0,1] = 2;
12 A[0,2] = 3;
13 A[0,3] = 4;
14 ..
15 ..
16 ..
17 ..
18 A[2,3] = 12;

```

تعريف المصفوفه

رقم الصف

رقم العمود

اعطانها قيم ابتدائيه

عدد الاعمده X اذا تم ضرب عدد الصفوف  
تحصل على عدد عناصر المصفوفه

اعطاء بيانات للمصفوفه من المستخدم اثناء تنفيذ البرنامج فى هذا المثال :

```

static void Main(string[] args)
{
    int[,] x = new int[4, 5]; حلقة تكراريه للمصفوف
    for (int i = 0; i < 4; i++)
    {
        for (int j = 0; j < 5; j++) حلقة تكراريه للاعمده
        {
            Console.WriteLine("enter value of element:({0},{1})", i, j); كتابة عنوان العنصر
            x[i, j] = int.Parse(Console.ReadLine());
        }
    }
    Console.ReadKey();
}

```

1 enter value of element:<0,0>  
 2 enter value of element:<0,1>  
 3 enter value of element:<0,2>  
 4 enter value of element:<0,3>  
 5 enter value of element:<0,4> *تم اعطاء قيم للعناصر حتى 20 عنصر*  
 6 enter value of element:<1,0>  
 7 enter value of element:<1,1>  
 8 enter value of element:<1,2>  
 9 enter value of element:<1,3>  
 10 enter value of element:<1,4>  
 11 enter value of element:<2,0>  
 12 enter value of element:<2,1>  
 13 enter value of element:<2,2>  
 14 enter value of element:<2,3>  
 15 enter value of element:<2,4>  
 16 enter value of element:<3,0>  
 17 enter value of element:<3,1>  
 18 enter value of element:<3,2>  
 19 enter value of element:<3,3>

ولقراءة عناصر المصفوفه :

```
1 Read 2d array values
2
3 Console.WriteLine(A[2,3]);
```

```
20
(0,0) value is == 1
(0,1) value is == 2
(0,2) value is == 3
(0,3) value is == 4
(0,4) value is == 5
(1,0) value is == 6
(1,1) value is == 7
(1,2) value is == 8
(1,3) value is == 9
(1,4) value is == 10
(2,0) value is == 11
(2,1) value is == 12
(2,2) value is == 13
(2,3) value is == 14
(2,4) value is == 15
(3,0) value is == 16
(3,1) value is == 17
(3,2) value is == 18
(3,3) value is == 19
(3,4) value is == 20
```

طباعة عناصر المصفوفه  
الثنائيه

وللتعديل على بيانات المصفوفه الثنائيه :

```
1 Modifi 2-d array
2
3 A[0,0] = A[0,0] * 2
```

## : Array Class Properties And Methods 🚩

الخصائص والدوال التي يتضمنها الفصيل Array

## Array Length – 1

ومنه تعرف طول المصفوفه ةتفيدك فى حالة اذا كان عدد العناصر مجهولا لك

بمعنى انه عند تعريف المصفوفه فلن يتم تحديد طول معين بل سيقوم المستخدم

بأدخاله وقت التنفيذ

```

1 syntax : Arrayname.lenght
2 example : A.lenght
3
4 example in programming :
5 int arraylength;
6 arraylength = int.parse(console.ReadLine());
7 int[] A = new int[arraylength];
8 for(i=0;i<A.lenght;i++)
9 {
10     A[i] = int.Parse(Console.ReadLine());
11 }
12 Console.ReadKey();
13

```

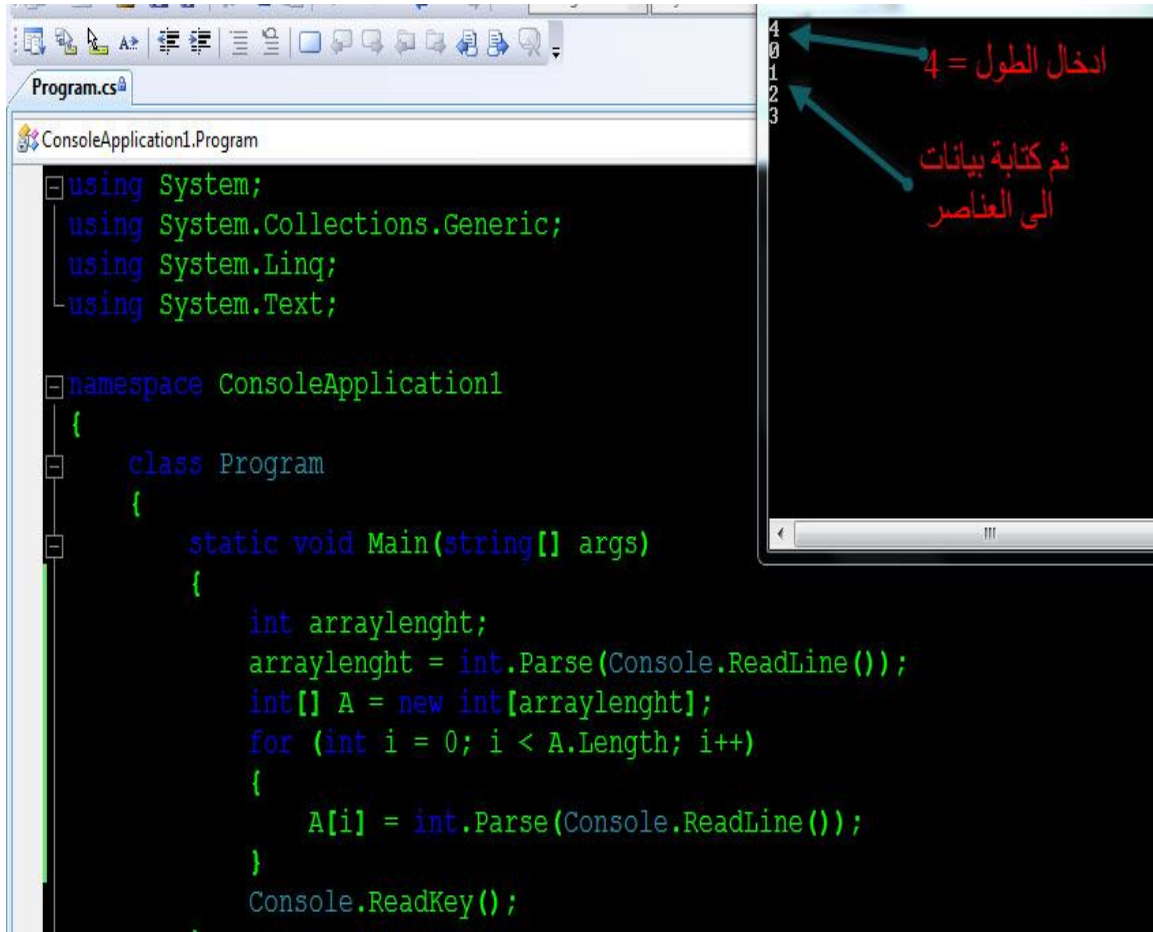
تعريف متغير بطول المصفوفه

جعل المستخدم هو من يحدد الطول المناسب

تحديد طول المصفوفه بالطول الذى ادخله المستخدم

ادخال بيانات الى العناصر

استدعاء خاصية الطول من الفصيل الذى يتبعه وهو المصفوفه وبالتالي سوف تأخذ القيمة التى ادخلها



2 - الخاصية Rank :

والتي منها تعرف نوع المصفوفه اذا كانت ثنائية الابعاد ام احادية البعد

واستدعائها يكون كالتالى :

```

1 Rank Properties
2
3 syntax : arrayname.Rank

```



### 3 – الخاصية التي تسمح بتغيير حجم المصفوفة Resize :

تغيير الحجم مع الاحتفاظ بالقيم القديمه لها والقاعده العامه لاستدعائها

```

1 Array Resize Properties
2 syntax : Array.Resize(ref  refname,newSize);
3
4
5 example : Array.Resize(ref A,20);

```

### 4 – خاصية البحث داخل عناصر المصفوفه Binary Search :

وتقوم هذه الخاصيه بالبحث داخل عناصر المصفوفه عن عنصر يحتوى على قيمه معينه

لاستخدامه ولكن شرط ان تكون عناصر هذه المصفوفه مرتبه ترتيبا تنازليا او تصاعديا

القاعده العامه لاستخدام الخاصيه :

```

1 serach item properties
2
3 syntax : var = Array.BinarySerach(A,8);
4
5 example : x = Array.BinarySearch(A,8);
6

```

5 – الخاصية التي تسمح بترتيب عناصر المصفوفة Sort :

القاعده العامه لاستخدامها :

```

1  Sorting array elements
2
3  syntax : Array.Sort (refname) ;
4
5  example : Array.Sort (A) ;
6

```

الخاصية ترتيب العناصر  
اسم المصفوفه المراد ترتيبها

6 – الداله Getting لاحضار قيمة العناصر والداله Setting لوضع قيم لعناصر

المصفوفه :

القاعده العامه لاستخدامهم :

```

1  Getting and Setting method
2  Getting method : syntax : var = refname.GetValue(i);
3  example : z = A.GetValue(3);
4  here if you set the value in avriable
5  you must convert it like this : z = int.Parse(A.GetValue(3));
6
7  متغير يحمل قيمة العنصر
8  تحويل القيمه
9
10 Setting method :syntax : refname.Setvalue(data,index);
11 A.Setvalue (20,2);

```

استخدام الداله مع رقم العنصر  
اسم المصفوفه  
تحويل القيمه  
متغير يحمل قيمة العنصر  
رقم العنصر  
القيمه المراد وضعها  
استخدام الداله لوضع قيمه



## 7 – دوال نسخ المصفوفات الى اخرى وهما نوعان

Copy – 1 وتقوم بنسخ جزء من مصفوفه الى اخرى

Copy To – 2 تقوم بنسخ كامل مصفوفه الى اخرى

قاعدة استخدامهم :

```

1 Copy and Copy to
2
3 Copy syntax : Array.Copy(Arrayref1,start1,Arrayref2,start2,no of element);
4
5 example : Array.Copy (A,5,B,6,3);
6
7
8
9 Copy To syntax : Arrayref1.CopyTo(Arrayref2,start);
10
11 example : A.CopyTo(B,0);
12

```

من اليسار اسم المصفوفه المراد نسخ العناصر منها - بداية النسخ - اسم المصفوفه المراد نسخ العناصر اليها - بداية وضع العناصر الاتيه - عدد العناصر المراد نسخها

كلمه محجوزه

الداله

بداية النسخ

اسم المصفوفه المراد النسخ اليها

اسم المصفوفه المراد نسخها بالكامل

الداله

## 8 – الداله Reverse :

والتي تقوم بعكس عناصر المصفوفه اى عكس وضعية العناصر

طريقة استخدامها :

```

1 Reverse Function
2
3 syntax : Array . Reverse (A);
4

```

كلمه محجوزه

الداله

اسم المصفوفه المراد تنفيذ الامر عليها

# C# Methods

الدوال واستخدامها داخل الـ سي شارب :

القاعده العامه لاستخدام الدوال داخل لغة C# :

```

1 Method decleration in c#
2 syntax :
3 [Access_modifier] Return_type method_name (arguments)
4 {
5 //method body
6 }

```

اهم ما يميز الداله عن المتغيرات العاديه  
وهو تحديد بارمترات تستخدم لتحديد  
نوع الدخول للداله والتي تؤدي عليه

هي عبارته عن  
معاملات  
تستخدم لتحديد  
سماحيه  
استخدام الداله  
داخل البرنامج  
والنوع  
الافتراضي لها  
هو خاص

نوع  
البيانات  
التي تعود  
بها الداله  
اسم يميز الداله  
والافضل ان  
يكون مرتبط  
بوظيفة الداله

انواع المعاملات او Access Modifiers :

1	Private :	لها فقط class يسمح هذا النوع بالتعامل مع الداله او المتغير داخل
2		
3		
4	Public :	class يسمح هذا النوع بالتعامل مع الداله او المتغير خارج
5		
6		
7	Protected :	يسمح هذا النوع بالتعامل مع لاداله او المتغير داخل المشروع فقط namespace لديه او
8		

- بعض الامثله على بناء الدوال وتعريفها داخل البرنامج :

1	examples :	
2		
3	int fact (int number);	داله لكن لها بارامتر واحد فقط
4	{method code}	
5		
6		
7	int bar ();	داله ليس لها اي بارامترات
8	{method code}	
9		
10		داله لها بارامتران لكن ليس لها قيمه عائده منها
11	void sum (int x, int y)	بسبب المعرف
12	{method code}	

void

حيث انه لا يسمح باى قيمه عائده من الداله

هذه هي الثلاثة احتمالات التي تبني عليها الداله داخل السي شارپ وهناك احتمال رابع وهو نادر وهو ان الداله لاتعود بقيمه وليس لها بارامترات ومن امثله ذلك الداله التي تستخدم في طباعة الخرج على الشاشه

WriteLine()

بعض الامثلة :

```

class Program
{
    void showdate ()
    {
        DateTime thedate;
        thedate = DateTime.Now;
        Console.WriteLine (thedata);
    }

    static void Main(string[] args)
    {
        Program pro = new Program ();
        pro.showdate ();
        Console.ReadKey ();
    }
}

```

داله لاظهار تاريخ اليوم ليس لها بارامترات او قيمه عانده

ولاستدعائها يجب تخليق كائن جديد من الفصيل التابعه له ثم استدعائها بهذا الكائن



```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static int getarea(int height, int width)
        {
            return height * width;
        }

        static void Main(string[] args)
        {
            Console.WriteLine (getarea (4, 4));
            Console.ReadKey ();
        }
    }
}

```

داله لها بارامتران وتعود بقيمة المساحه

استدعاء الداله واعطائها قيمة البارامترات



ومن المثالين السابقين نريد ان نسلط الضوء على بعض الاساسيات الهامه جدا فى استخدام الدوال :

1 – ان استدعاء الدوال يكون على حسب المعرف لها او Access Modifier وقد قمنا بشرح انواعه .

2 – ان كتابة الداله داخل الفصيل او Class قبل الداله الرئيسيه Main او بعدها ويكون استدعائها من داخل الداله Main او من اى داله اخرى .

3 – ان استدعاء اى داله لا يكون الا عن طريق انشاء كائن جديد من الفصيل او Class والذي تتبعه الداله او الذى تم انشائها بداخل كواحد من عناصره الاساسيه ولا يكون غير ذلك الا فى حاله واحده وهى ان تعطى المعرف Static وهو الوحيد الذى يسمح بأستدعاء الداله بشكل منفرد دون الحاجه الى كائن المخلق من الفصيل ويفسر هذا وضع هذا المعرف دائما قبل الداله الرئيسيه Main داخل اى برنامج فى السى شارب حتى يتسنى استدعائها من داخل البرنامج مباشرة لان التنفيذ يبدأ من عندها

4 – ان الداله او قد يكون لها اكثر من بارامتر او لا يكون لديها اى بارامترات وقد تعود بقيمه واحده او اكثر من قيمه او لاتعود بقيم على الاطلاق لكن يجب ان تلاحظ الفرق بين الحالتين فى المثال التالى :



```

namespace ConsoleApplication1
{
    class Program
    {
        void showdate()
        {
            DateTime thedate;
            thedate = DateTime.Now;
            Console.WriteLine(thedate);
        }
    }

    static int getarea(int height, int width)
    {
        return height * width;
    }
}

```

void ان استخدام كلمة التعريف تمنعك من ان تكون الداله لها قيمه عائده وتلاحظ هذا في عدم وجود الكلمه المحجوزه return

ان وجود نوع من انواع البيانات للداله اذا لا بد وان يكون لها قيمه عائده اى لا بد من استخدام الكلمه المحجوزه return

ان الكلمتان void و return لا يجتمعان في تركيب داله واحده

5 - من الممكن اعطاء القيم التي تعود بها الدوال لمتغيرات لكن من نفس نوع البيانات

التي تعود به الداله

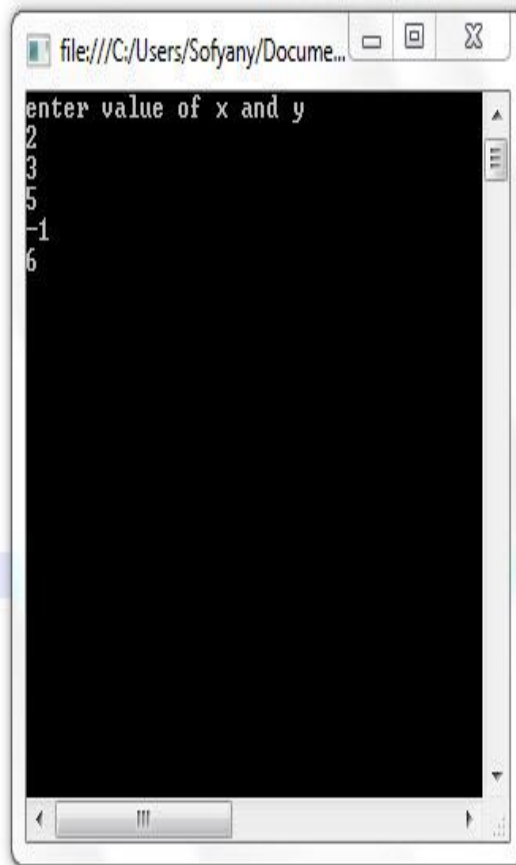
6 - لا بد من اعطاء كافة القيم للبارامترات التي تحملها الداله داخل الكود ولا بد ان تكون

القيم من نفس نوع تلك البارامترات

- في المثال التالي نقوم بعمل برنامج يقوم ببعض العمليات الرياضيه لكن

باستخدام الدوال :

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 namespace ConsoleApplication1
6 { class Program
7     { static double sum(int x, int y)
8         {
9             return x + y;
10        }
11        static double sub(int x, int y)
12        {
13            return x - y;
14        }
15        static double mul(int x, int y)
16        {
17            return x * y;
18        }
19        static void Main(string[] args)
20        {
21            int x, y;
22            Console.WriteLine("enter value of x and y");
23            x = int.Parse(Console.ReadLine());
24            y = int.Parse(Console.ReadLine());
25            Console.WriteLine(sum(x, y));
26            Console.WriteLine(sub(x, y));
27            Console.WriteLine(mul(x, y));
28            Console.ReadKey();
29        }
30    }
```



```
file:///C:/Users/Sofyany/Docume...
enter value of x and y
2
3
5
-1
6
```

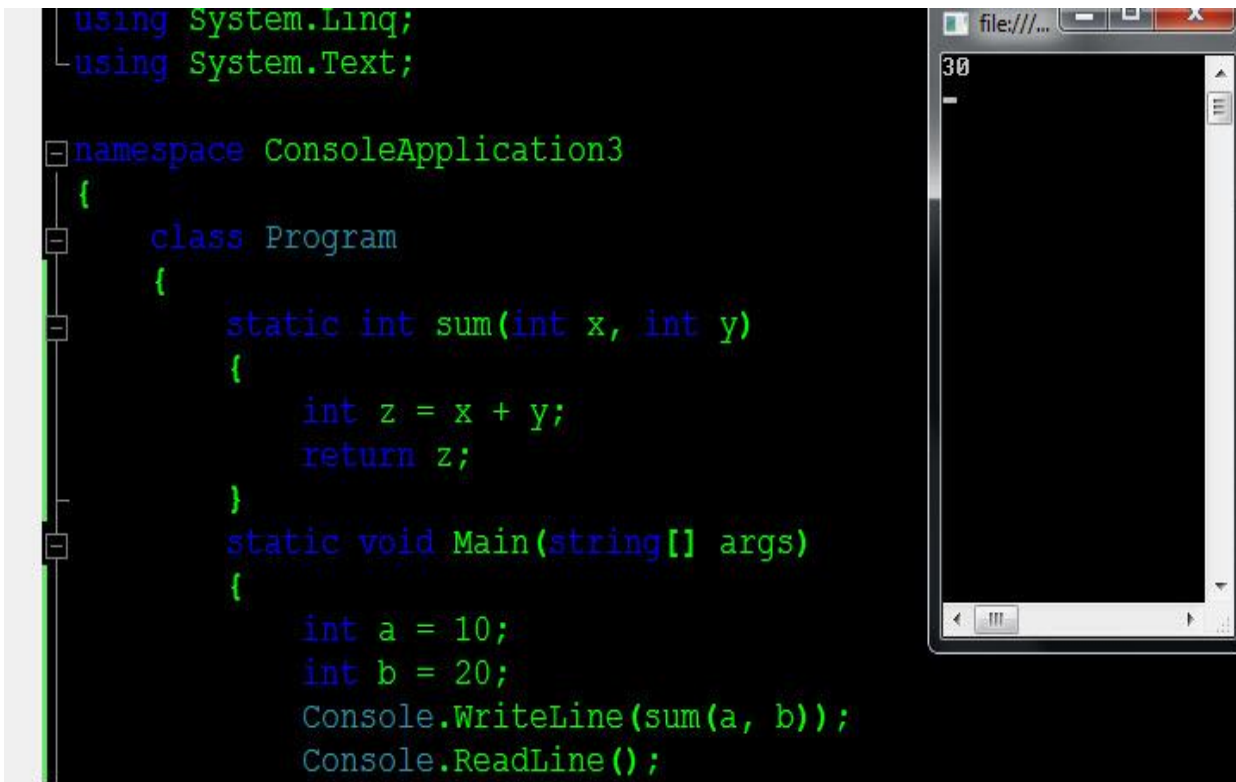
طرق اعطاء البارامترات الى الدوال : Passing Parameters to method

by value – 1

وهي ارسال القيمه الى البارامتر مباشرتا وتكون عباره عن نسخ Data الاساسيه

ونقلها الى مكان في الذاكره وبالتالي هذا استغلا سييء للذاكره لانه ياخذ مساحه

اكبر بدون داعى ومثال عليه :



```

using System.Linq;
using System.Text;

namespace ConsoleApplication3
{
    class Program
    {
        static int sum(int x, int y)
        {
            int z = x + y;
            return z;
        }
        static void Main(string[] args)
        {
            int a = 10;
            int b = 20;
            Console.WriteLine(sum(a, b));
            Console.ReadLine();
        }
    }
}

```

by reference – 2

الطريقه الثانيه وهي اعطاء البارامترات عن طريق الاشاره الى مكان معين في الذاكره

واليك المثال التالي :



```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

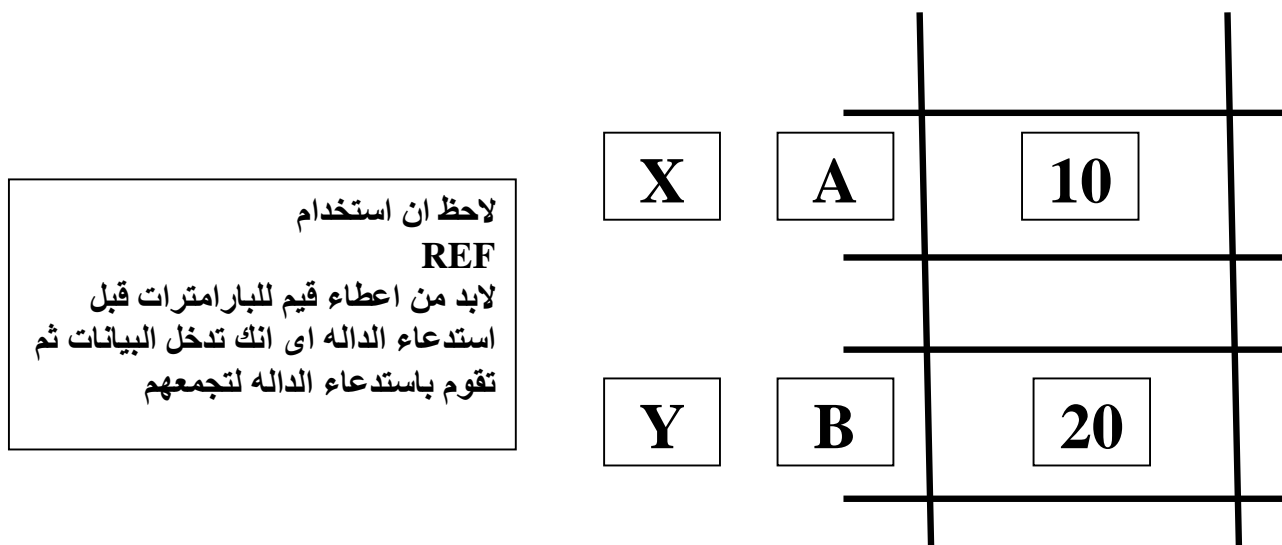
namespace ConsoleApplication3
{
    class Program
    {
        static int sum(ref int x,ref int y)
        {
            int z = x + y;
            return z;
        }
        static void Main(string[] args)
        {
            int a = 10;
            int b = 20;
            int z = sum(ref a,ref b);
            Console.WriteLine(z);
            Console.ReadLine();
        }
    }
}

```

استخدام الكلمه الحجزه قبل البارامترات  
مره عند التعريف ref

مره اخرى عن استدعاء الداله واعطاء  
قيم للبارامترات

ان استخدام هذه الطريقه هو الاستخدام الامثل للذاكره لانها كلا من البارامترات والمتغيرات التى تمثلها تشير الى مكان واحد فى الذاكره



by out – 3

هى نفس استخدام ref لكن الاختلاف فى انك تستدعى الداله اولاً ثم تعطيهها قيما

ابتدائيه ولنرى فى المثال التالى :

```

static int GetArea(int width, int height, out int prem)
{
    prem = (width * 2) + (height * 2);
    return width * height;
}
static void Main(string[] args)
{
    int area;
    int prem;
    area = GetArea(5, 10, out prem);
    Console.WriteLine("the area is :{0}&the prem is :{1}", area, prem);
    Console.ReadLine();
}

```

الطول والعرض اخذوا قيما مباشره لانهم  
وتم حساب المساحه بهم byvalue معرفين  
لكن المحيط لم يأخذ قيم مباشره بل تم حسابه  
عن طريق القيم التى اعطيت له من الطول  
والعرض

## : C# Variable Scopes

النطاق الذي تستخدم فيه المتغيرات داخل اللغة :

وله ثلاثة انواع :

class level – 1

متغيرات يتم تعريفها داخل class ويتم استخدامها مع اي method اخرى

method level – 2

متغيرات خاصه ب method ولا تستطيع استخدامها مع اي method اخرى

nested level – 3

متغيرات تعرف داخل اي block من data ولا تستخدم خارج هذا block مثل الحلقات

مثلا او الجمل الشرطيه

## : Passing Arrays as Parameters 🚩

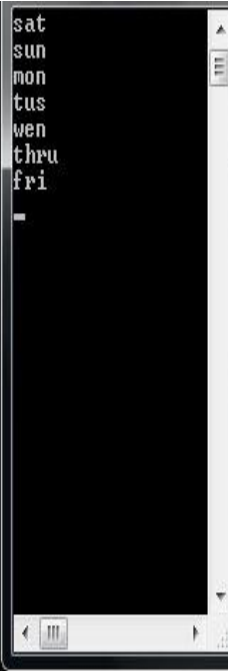
## Passing by value – 1

برنامج عمل array ويضع قيمه ابتدائيه ويعطيها الى method لكي تطبعها

على الشاشة وهنا سوف نعطي المصفوفه للداله كبارامتر

```
using System.Text;

namespace ConsoleApplication3
{
    class Program
    {
        static void printarray(string[] w)
        {
            for (int i = 0; i < w.Length; i++)
            {
                Console.Write(w[i]);
                Console.WriteLine();
            }
        }
        static void Main(string[] args)
        {
            string[] weekdays = new string[]
            { "sat", "sun", "mon", "tus", "wen", "thru", "fri" };
            printarray(weekdays);
            Console.ReadLine();
        }
    }
}
```



## : Method overloading 🇸🇦

من الممكن بناء اكثر من داله بنفس الاسم لعمل عدة وظائف تحت مسمى واحد بشرط ان تختلف هذه الدوال فى عدد البارامترات او ان البارامترات تختلف نوع البيانات المعرفه بها وتسمى هذه العمليه Overloading او التحميل الذايد

```

{
class Program
{
    public int add(int a, int b)
    {
        return a + b;
    }
    public float add(float a, float b)
    {
        return a + b;
    }
    public int add(int a, int b, int c)
    {
        return a + b + c;
    }
    static void Main(string[] args)
    {
        Console.WriteLine(add(2, 3));
        Console.WriteLine(add(2.2, 3.3));
        Console.WriteLine(add(2, 3,4));
        Console.ReadLine();
    }
}
}

```

جميع الدوال بنفس الاسم لكنها تختلف من ناحية تعريف البارامترات ونوع بياناتها

# Object Oriented Programming

- رؤيه فلسفيه حول البرمجه الكائنية التوجه :

ان مفهوم البرمجه الكائنية التوجه لهو بالاساس يعتمد على النظرة المجرده للحياه التى نعيشها ذلك بأن كل شىء فى هذه الحياه عباره عن مجموعه من الكائنات ولها صفاتها الخاصه ومميزاتها والشكل التى تتصرف به وتجد ان الطبيعه قد ربطت بين كل الاشكال المختلفه للحياه كل منهم له اسلوبه الخاص بالتعامل واسلوبه الخاص فى الطريقه التى يتعامل بها مع الاخر وذلك الربط ادى فى النهايه الى تكوين فضاء يحتوى هذه الكائنات جميعا لكى توجه الى فكرة العوالم المتوازيه مبرمجه طبيعيا لان تتعامل مع بعضها فى تعاون برمجى كامل وبالتالي ارادو تطبيق تلك الفكره والسير على خطاها فى مفهوم البرمجه الكائنية التوجه فهو يعطى فكره مجردة للنظام ككل ثم يقسمه الى مجموعه من الكائنات التى لها خصائص تميزها عن غيرها ويوضع مع كل كائن الاساليب التى يتعامل بها مع غيره .

مزايا البرمجه كائنية التوجه :

1 – سهولة متابعة وصيانة السوفت وير

2 – سرعة تطوير البرمجيات ومتازتها

## Object Oriented Programming Concepts

### Encapsulation – 1

ان مجموعه من العناصر لها نفس الخصائص ولها نفس السمات من الممكن ان تجتمع داخل فصيل واحد ويسمى Class وهذا الفصيل هو الذى يمثلها بوجه عام لكن لا يتعامل

معها عن طريقه بل عن طريق كائن مخلق منه وهو Object

- اذا ما اردنا ايجاد امثله من الحياه لدينا Class يمثل البشريه وهذا الفصيل له

جميع صفات البشريه واسلوب تعاملها مع غيرها والذى هو فصيل رئيسى من

منظومه اشمل وسوف يطلق عليها namespace وهذه المنظومه ممثله فى الكون

اذا اردنا انشاء كائن من فصيل البشريه ليكن الانسان الذى يحيا على الارض واى

انسان اذا يحمل جميع صفات وسلوكيات هذا الفصيل

ومن الممكن ان ننشئ فصيل اخر من فضاء الكون وليكن فصيل للحيوان

ومن الممكن ان يكون هناك فصيل للنباتات واخر للجماذ وهكذا



: Class Builder Features 

مالذي توفره لك Classes من مزايا داخل البرمجة كائنية التوجه ؟

1 – استطعت بذلك ان تقوم بتمثيل كل Object او عنصر داخل System بمجموعه من Classes والتي تحتوى على كل التعاريف لهذا Object من سمات وتصرفات

2 – استطعت حماية بياناتك الاساسيه لهذا Object من محاولة العبث بها او حتى تغييرها والتي تتمثل فى المفاهيم الاتيه :

## : 1 – Encapsulation Features مزايا التغليف

```

1 Encapsulation Builder :
2 بينى الفصيل بواسطة الكلمة المحجوزه التى تعرفه ثم يعطى اسم ثابت له
3 class class_name
4 {
5 //attributes// تكتب هنا المتغيرات او
6 int var1; ما يسمى بالمعلومات
7 int var2; التى تخص هذا الفصيل
8 string var3;
9
10 public string function_name(parameters)
11 {
12 //method body// توضع هنا الدوال التى تعبر عن تصرفات الفصيل والتى تكون ثابتة خلال حياة النظام
13 }
14
15 public data_type property_name
16 {
17 get
18 {
19 return attributes_value; توضع هنا الخواص التى يعتمد عليها
20 } الفصيل والتى تعطى قيما متغيره وهذه
21 set الخواص هى التى تعطيك التحكم فى
22 { قيم لمعلومات الفصيل عند بعد
23 attributes = value;
24 }
25 }
26 }

```

مزايا التغليف :

### State Retention – 1

الابقاء على حاله وتصرفات العناصر او Object داخل System وذلك يمثل

فى الجزء الذى يعطى له وهو Methods فقط انا اعطى لها القيم وهى تقوم

بالتصرف الثابت لها

### Data Hiding – 2

يتم عزل البيانات والتي تكون ممثله فى Attributes للعنصر Object داخل

النظام بعيدا عن اى تغيير

ويبقى هنا السؤال كيف لى ان استخدام هذا الفصيل الضخم والملبىء بعدة

عناصر وكيف لى ان استخدام تلك العناصر من خارج هذا الفصيل ؟

وتأتى الاجابه بأن بناء الفصيل شىء واستخدام محتوياته شىء اخر بمعنى ان

الفصيل الذى يسمى البشريه هو فصيل عام يطلق على بنى البشر لكن يجب ان يجسد

فى عنصر معين لكى تستطيع ان تتعامل معه وهذا العنصر الذى يمثله هو الانسان

وبالتالى سوف يتم انشاء عنصر من كل فصيل لكى تستطيع ان تستخدم كافة عناصره

انتهت النظره الفلسفيه لى الى تدعيمها بأمثله عمليه :

🚩 سماحية استخدام عناصر Class داخل البرنامج :

تحتوى لغة C# على خمسة انواع منهم :

```
1 private : وتعطى سماحية استخدام عناصر الفصيل داخل الفصيل فقط :  
2  
3  
4 protected: داخل الفصيل او فصيل مشتق منه  
5  
6  
7 internal: يستخدم فى اى مكان داخل المشروع عامة  
8  
9  
10 protected internal : استخدامه داخل المشروع او مشروع مشتق منه  
11  
12  
13 public: يتعامل مع عناصر الفصيل من اى مكان داخل نفس المشروع او داخل  
فضاء الاسماء المضمن بداخله
```

بعض الامثله على بناء Classes داخل C# :

```

using System.Text;

namespace ConsoleApplication3
{
    class EX1      تعريف الفصيل بأسم معين
    {
        private int x;      تعريف المتغيرات من نوع خاص اي لا
        private int y;      يمكن استخدامها الا داخل الفصيل فقط

        public void setdata(int a, int b)
        {
            x = a;      تعريف داله من نوع عام والتي يمكن استدعاؤها
            y = b;      خارج الفصيل والتي تعطى قيما للمتغيرات
        }

        public int sum()
        {
            return x + y;      تعريف داله للجمع
        }

        public float avg()      تعريف داله للمتوسط الحسابي
        {
            return sum() / 2;
        }
    }
}
class Program

```

والان كيف لنا ان نستخدم عناصر هذا الفصيل داخل البرمجه ؟

والجواب هو ان نقوم بانشاء كائن يتبع هذا الفصيل وبالتالي فهو يستطيع استدعاء كافة

عناصره كالتالي : القاعده البرمجيّه

```

1 create object from class
2
3 class_name obj_name = new class_name;
4

```

اسم الفصيل مره اخرى ←

كلمه محجوزه تدل على انشاء الكائن

اسم الكائن الذي تريد انشاؤه

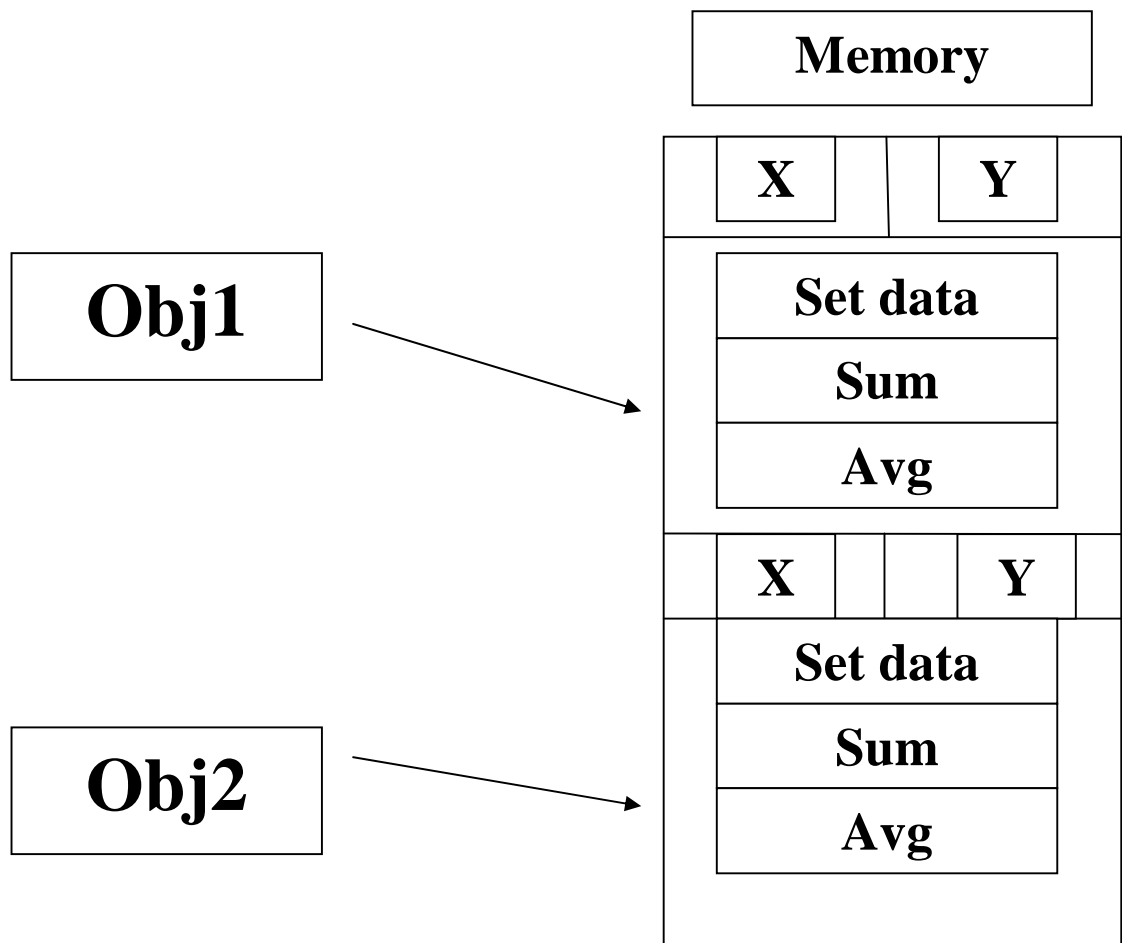
اسم الفصيل الذي تريد انشاء كائن منه

```

class Program
{
    static void Main(string[] args)
    {
        EX1 obj1 = new EX1 ();
        EX1 obj2 = new EX1 ();
        Console.ReadLine ();
    }
}

```

انشاء الكائن وتعريفه يكون داخل الفصيل الرئيسي داخل الداله الرئيسييه ومن الممكن تعريف اكثر من كائن من نفس الفصيل ولهذا ميزه كبيره وهي ان عناصر الكائنين لا يتاثروا ببعضهم البعض رغم انهم ينتمون لفصيل واحد لكن لكل منهم مكانه في الذاكره



توصيات هامه : يفضل ان تكون المتغيرات داخل الفصيل تعطى المعرف Private حتى لا يتم العبث بها او تغييرها .

يفضل ان تعطى الدوال داخل الفصيل المعرف Public حتى تستطيع استخدامها خارج الفصيل .

📌 كيفية الاتصال باعضاء class باستخدام الكائن الجديد:

القاعده البرمجييه :

```

1 Accessing object members:
2
3 object_name . member_name;
4

```

اسم الكائن                      اسم العضو

سنقوم فى المثال السابق باعطاء قيم لعناصر الكائنين الذى تم تعريفهم من نفس الفصيل ونرى النتيجة لدالتين الجمع والمتوسط الحسابى :

```

static void Main(string[] args)
{
    EX1 obj1 = new EX1 ();
    EX1 obj2 = new EX1 ();

    obj1.setdata(5, 10);
    obj2.setdata(15, 20);
    Console.WriteLine(obj1.sum());
    Console.WriteLine(obj1.avg());
    Console.WriteLine(obj2.sum());
    Console.WriteLine(obj2.avg());
    Console.ReadLine();
}

```

نرى الان اختلاف النواتج مما يؤكد ان كل كائن يعمل بمفرده

هناك دالتين رئيسيتين تعطى قيما ثابتة Private members وهذا يحدث مع

اي لغة تستخدم البرمجة كائنية التوجه حيث تستخدم دالتين :

1 – Setter : لاعطاء القيم الابتدائية لاجزاء class

2 – Getter : لقراءة البيانات واظهارها على الشاشة

بعض الامثله على ذلك :

```

1 private string name;
2
3 //getter function :
4 public string getname()
5 {
6 return name;
7 }
8 //setter function
9 public void setname(string thename)
10 {
11 name = thename;
12 }
```

هذه الدوال من الممكن استخدامها داخل السي شارب لكن اللغة لها اسلوب متطور وهو استخدام الخواص او properties والتي تقوم بنفس المهمه بشكل اكثر كفاءه ويتوافق مع النظرية الكائنية التوجه

انواع الاداه Properties :

1 – Read Only

2 – Write only

3 – Read / Write



القاعده العامه لتعريفها داخل Classes :

syntax of properties:

```

    اسم الخاصيه          نوع البيانات التي تعود بها          نوع الإتصال بالخاصيه
    <access modifiers><return data type><name of property>
    {
    get  كلمه ثابتة معرفه داخل اللغه وتحل محل الداله
        {
            getter
        }
    return <some private field>;  اسماء المتغيرات التي سوف تعود
        الخاصيه بالقيم التي تعطى لها
    }
    set  كلمه محجوزه وتحل محل الداله
        {
            setter
        }
    <some private field> = value;  كلمه محجوزه وتعبّر عن القيم التي سوف تعطى للمتغيرات
        عن طريق الخاصيه اثناء تنفيذ البرنامج
    }  اسماء المتغيرات التي تريد
    }  اعطاء القيم لها عن طريق
    الخاصيه وهي نفسها الموجوده
    في الجزء الخاص ب
    get
  
```

بعض التوصيات :

الخواص التي تبني داخل classes ليس لها اي بارامترات

اسم الخاصيه لا بد ان يكون هو نفسه اسم المتغير التي تتعامل معه مع الفارق بان يكون

اول حرف من اسم الخاصيه capital حتى يتميز عن المتغير

ولنرى بعض الامثله على بناء الخواص :

```

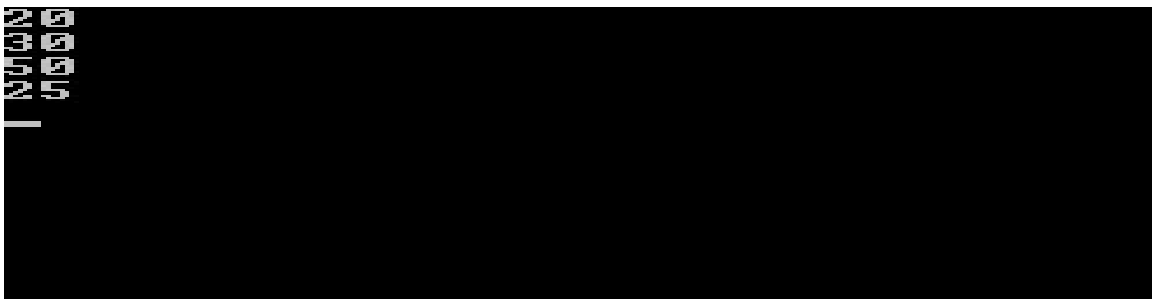
class example
{
    int num1, num2;
    public int Num1
    {get{ return num1; }
     set{ num1 = value; } }
    public int Num2
    {
        get { return num2; }
        set { num2 = value; }
    }
    public int sum()
    { return num1 + num2; }
    public float avg()
    { return sum() / 2; }
}

```

```

class Program
{
    static void Main(string[] args)
    {
        example obj1 = new example();
        obj1.Num1 = 20;
        obj1.Num2 = 30; اعطاء القيم عن طريق الخواص
        Console.WriteLine(obj1.Num1);
        Console.WriteLine(obj1.Num2); اظهار القيم عن طريق الخواص
        Console.WriteLine(obj1.sum());
        Console.WriteLine(obj1.avg()); اظهار نواتج الدوال
        Console.ReadLine();
    }
}

```



لاحظ ان القيم اعطيت للمتغيرات عن طريق الكلمه set داخل الخاصيه وتم اظهار  
البيانات عن طريق الكلمه get

: Static Member 🚩

ان هذا المعرف له ميزه خاصه جدا داخل السى شارب حيث انه يقوم بمشاركه العضو  
الذى يعرف بواسطته بين جميع الكائنات التى تعرف من نفس الفصيل ويلزم لتعريف



ولنرى مثال عام :

```

{
class student
{
    تعريف العضو المشارك عليه من جميع الكائنات
    public static int phonenumber;
    public int idnumber;
    public string name;
}
class Program
{
    static void Main(string[] args)
    {
        student st1 = new student();
        student st2 = new student();
        st1.idnumber = 3;
        st1.name = "ahmed";
        st2.idnumber = 4;
        st2.name = "kamel";
        student.phonenumber = 492067;
        عدا رقم التليفون فقد تم استدعاؤه عن طريق الفصيل
        مباشرة
    }
}
}

```

ان اى static member ينتمى الى الفصيل مباشرة وليس الكائن وهذه ميزه

هامه جدا داخل السى شارب

🚩 دوال البناء والهدم داخل السي شارب :

### 1 – دالة البناء Constructor :

هي دالة خاصة باى Class وتستخدم لوضع القيم الابتدائية لعناصره

خصائصها :

1 – تأخذ نفس اسم Class

2 – ليس لها قيم مسترجعه

3 – يتم تنفيذ دالة البناء تلقائيا مع كل تعريف لكائن جديد من Class

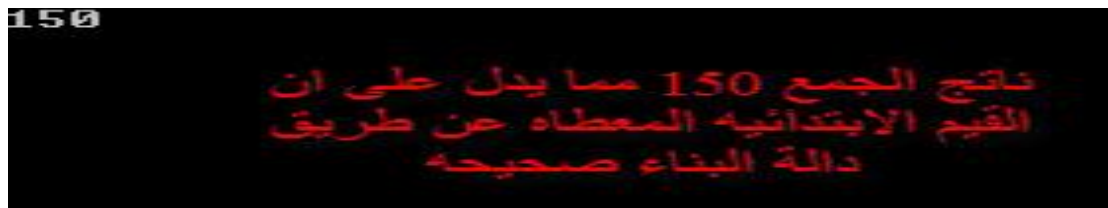
4 – الكود بداخله يستخدم لوضع Data لاي عضو داخل Class

امثله على ذلك :

```
class example
{
    private int x;
    private int y;
    public example ()
    {
        x = 50;
        y = 100;
    }
    public double sum()
    {
        return x + y;
    }
}
class Program
{
    static void Main(string[] args)
    {
        example ex = new example ();
        Console.WriteLine(ex.sum());
        Console.ReadLine();
    }
}
```

تعريف دالة البناء والتي تعطى قيما ابتدائية للمتغيرات داخل الفصيل ويكون اسمها نفس اسم الفصيل

بناء دالة اخرى لاطهار محتوى القيم الابتدائية



فى حالة عدم بناء هذه الداله بداخل اى فصيل فسيتم وضع قيمه ابتدائيه لاجزائه  
كالتالى :

String = null  
Boolean = false  
Int = 0  
Float = 0.0

وذلك لان دالة البناء يتم بنائها تلقائيا داخل اى فصيل وهذه ميزه هامه جدا داخل  
السى شارب حيث انه يتم تخزين هذه القيم داخل الذاكره منعا لتخزين قيم  
عشوائيه وبالتالي فانت تعرف مسبقا ما الذى يوجد داخل عناوين الذاكره بالنسبه  
للمتغيرات حتى ولو لم تعطيهما قيمه ابتدائيه ونستطيع التاكد بالمثل التالى :

```

class example
{
    private int x;
    private float y;
    private double z;
    private string name;
    private Boolean c;
    public example()
    {
        Console.WriteLine(x);
        Console.WriteLine(y);
        Console.WriteLine(z);
        Console.WriteLine(name);
        Console.WriteLine(c);
    }
}

class Program
{
    static void Main(string[] args)
    {
        example ex = new example();
        Console.ReadLine();
    }
}

```

فيما يلي انواع المتغيرات والتي لن يتم اعطاء  
قيما ابتدائية لها عن طريق دالة البناء فقط  
سنقوم باظهار القيم الافتراضية لها

هنا فقط ننشئ كائن جديد وسيقوم  
باستدعاء دالة البناء تلقائيا



False

هذه هي القيم الافتراضية  
لكل نوع من المتغيرات في  
حالة عدم وضع اي قيم لها



انواع دالة البناء داخل السى شارب :

: Without Parameter – 1

وتستخدم فى وضع قيم ابتدائيه ثابتة داخل اى كائن وتعرف بنفس اسم Class وفى هذا النوع يكون لجميع عناصر الفصيل نفس القيم مع تعريف كائن جديد من نفس

الفصيل

: With Parameter – 2

وتستخدم لوضع قيم ابتدائيه مختلفه داخل الكائن وفى هذا النوع يكون هناك قيم مختلفه لعناصر الفصيل على حسب البارامترات

```
class example
{
    private int num1;
    private int num2;
    //constructor with parameter
    public example(int a,int b)
    {
        num1 = a;
        num2 = b;
    }
    //constructor without parameter
    public example()
    {
        num1 = 50;
        num2 = 100;
    }
}
class Program
{
    static void Main(string[] args)
    {
        example ex1 = new example();
        example ex2 = new example(20, 30);
        Console.ReadLine();
    }
}
```

دالة البناء بوجود بارامترات لها

دالة البناء فى حالة عدم وجود بارامترات واستخدام القيم الابتدائيه

الان الداله التى لها احقية فى التنفيذ تكون على حسب تعريف الكائن

النوع الثانى بدون بارامترات

النوع الاول مع وجود البارامترات

مثال شامل على استخدام Classes في البرمجة :

نريد عمل مثال باستخدام Classes وذلك لحساب المساحات لثلاثة اشكال مختلفه

المربع - المستطيل - المثلث على ان يتم الاتي

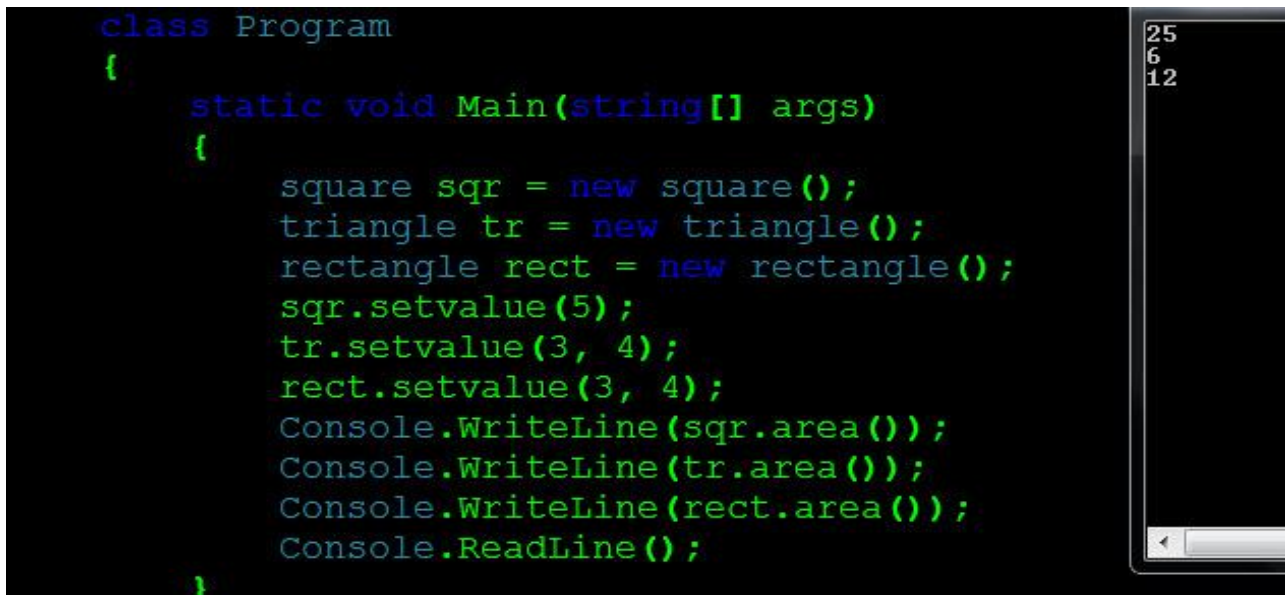
1 - تعريف المتغيرات اللازمه لايجاد المساحات

2 - بناء داله لاعطاء القيم لهذه المتغيرات

3 - بناء داله لحساب المساحات

```
class square
{int lenght;
    public void setvalue(int a)
    { lenght = a; }
    public int area()
    { return lenght * lenght; }}
class rectangle
{ int wedith, lenght;
    public void setvalue(int a, int b)
    { wedith = a;
      lenght = b; }
    public double area()
    { return wedith * lenght; } }
class triangle
{ int lenght,tbase;
    public void setvalue(int a,int b)
    {lenght = a;
      tbase = b; }
    public double area()
    { return (lenght * tbase) / 2; } }
```

```
class Program
{
    static void Main(string[] args)
    {
        square sqr = new square();
        triangle tr = new triangle();
        rectangle rect = new rectangle();
        sqr.setvalue(5);
        tr.setvalue(3, 4);
        rect.setvalue(3, 4);
        Console.WriteLine(sqr.area());
        Console.WriteLine(tr.area());
        Console.WriteLine(rect.area());
        Console.ReadLine();
    }
}
```



لكن هناك طرق اكثر احترافيه لكتابة مثل هذه البرامج وذلك لانه تم تكرار نفس المتغيرات ونفس الدوال فى جميع Classes ولفادى ذلك يجب ان نتعرف على شىء هام جدا يسمى الوراثة داخل البرمجه كائنية التوجه .

# Inheritance

ان الوراثة هي عبارة عن اشتقاق فصيل من اخر تحمل نفس خصائصها وهذه ميزه هامه جدا داخل البرمجه كائنية التوجه وسبق لنا ان شرحت هذا الجزء في كتاب هندسة البرمجيات الخاص بي الجزء الثالث منه  
يسمى Class الرئيسي بعدة اسماء :

Original class – base class – parent class – super class

ويسمى Class المشتق بعدة اسماء :

New class – derived class – child class – sub class

ان الوراثة لها ميزتين :

1 – انك تستطيع اشتقاق فصيل من اخر يرث جميع العناصر الموجوده فيه وبالتالي توفر على نفسك عناء اعاده بعض الاكواد يتم تكرارها في البرنامج كمثال المساحات السابق

2 – انك تستطيع الاضافه داخل الفصيل المشتق الى عناصر الفصيل الرئيسي

القاعده البرمجيه للاشتقاق داخل السى شارب :

```

1 Creating derived class from another
2
3 Class derived_classname : base_class name
4 {
5
6 }

```

اسم الفصيل الاساسي

اسم الفصيل المشتق

كلمه معرفه داخل اللغه  
وتعنى بناء الفصيل

يوجد نوعان من الاشتقاق :

1 – النوع الاول انك تشتق فصيل جديد يحمل جميع عناصر الفصيل الرئيسى دون اى

اضافات

2 – النوع الثانى انك تستطيع ان تضيف خصائص اخرى للفصيل المشتق

تطبيق الوراثة على مثال المساحات سوف نبني فصيل رئيسى يسمى الشكل او

Polygon ونشتق منه جميع Classes الاخرى

```

class polygon
{protected int w, l;
  public void setvalue(int a, int b)
  { l = a;
    w = b;} }
class squar : polygon
{public int area()
  { return l * w; }}
class rectangle : polygon
{ public double area()
  { return l * w; } }
class triangle : polygon
{ public double area()
  { return (l * w) / 2; } }

```

لقد تم بناء الفصيل الأساسي والذي يسمى الشكل واشتقاق جميع الفصائل منه مع تغيير الدالة التي تحسب المساحة في كل شكل على حدى

لاحظ في هذا المثال استخدمنا في تعريف المتغيرات المعرفة Protected وليس Private والسبب لكي تستطيع استخدام المتغير من فصيل مشتق وذكرنا هذا من قبل

التعديل على بعض الخصائص داخل الفصيل الرئيسي واستخدام قاعدة :

## Overriding Base Class Functionality

تستطيع التعديل على بعض الخصائص التي كانت موجوده داخل الفصيل الاب وذلك باستخدام الكلمه الثابته داخل السى شارب Overriding واستخدام اسم الداله التي تريد

التغيير فيها او ما يسمى Match Signature

وحتى تكتمل قاعدة التغيير وتكون صحيحه فعند بناء الداله فى الفصيل الاساسى والتي تنوى التغيير فيها مستقبليا عند اشتقاق فصائل اخرى من هذا الفصيل فيجب ان تعطى

المعرف Virtual

مثال على ذلك فى برنامج المساحات :

```

{
class polygon
{protected int w, l;
  public virtual void setvalue(int a, int b)
  { l = a;
    w = b; } }
class squar : polygon
{
  public override void setvalue(int a, int b)
  {
    l = a;
    w = a;
  }
  public int area()
  { return l * w; }}

```

اعطيناها المعرف  
Virtual  
عند تعريفها داخل الفصيل الاساسى

وعند التعديل اعطيناها المعرف  
Override



هناك ميزة اخرى داخل الاشتقاق وهى انك تستطيع استدعاء الدوال فى الفصيل

الاساسى داخل الفصيل المشتق وذلك باستخدام الكلمه المعرفه داخل اللغه Base

```
class polygon
{protected int w, l;
  public void setvalue(int a, int b)
  { l = a;
    w = b;} }
class squar : polygon
{
  public int area()
  {
    base.setvalue(2, 2);
    return l * w;
  }
}
```

استدعاء الداله بواسطة الكلمه Base داخل الفصيل المشتق

: Multi – level Hierarchies

عملية الاشتقاق المتتابع لفصيل من اخر بمعنى ان عملية اشتقاق تمت من فصيل

رئيسى تلتها عملية اشتقاق من الفصيل المشتق وهكذا مما يعطيها شكل هرمى

ولاحظ انك فى كل مره تشتق فيها فصيل من الاخر تستطيع ان تعدل او تضيف

خصائص جديده ويجب ان تلاحظ ان التعديلات لها الاولويه فى التنفيذ عن

الخصائص الاساسيه

```

class polygon
{protected int w, l;
  public void setvalue(int a, int b)
  { l = a;
    w = b;} }
class squar : polygon
{ public int area()
  { base.setvalue(2, 2);
    return l * w; } }
class asquar : squar
{ public int area()
  { base.setvalue(3, 3);
    return l * w; }}
class bsquar : asquar
{public int area()
  { base.setvalue(4, 4);
    return l * w;} }

```

تابع عمليات الاشتقاق المتدرج

## : Preventing Inheritance 🚧

منع الوراثة او الاشتقاق داخل البرمجه كائنية التوجه حفاظا على فصيل معين من الاشتقاق وتحدث كثيرا مع Classes المبنيه داخل السى شارب اساسا والتي قامت شركة مايكروسوفت ببرمجتها حفاظا على حقوق ملكية الكود وتسمح باشتقاق بعض

الفصائل الاخرى

القاعده البرمجيه لمنع اشتقاق Class :

```

1 preventing inheritance
2
3 Sealed Class Class_name
4 {
5
6 }

```

اسم الفصيل الذي تريد منعه من الوراثة

الكلمه المعرفه والتي تقوم ببناء الفصيل

الكلمه المعرفه داخل اللغه والتي تمنع وراثة الفصيل

ولاحظ ان هذه العمليه تتم عند بناء الفصيل الرئيسي قبل ان يشتق

```

sealed class polygon
{
    protected int w, l;
    public void setvalue(int a, int b)
    { l = a;
      w = b;}
}

class squar:po
class Progra

```

لم يعطيك الفرصه لاشتقاق الفصيل الاساسي

class System.EntryPointNotFoundException  
The exception that is thrown when an attempt to load a class fails due to method.

# Polymorphism

ان مفهوم Polymorphism داخل السى شارب هو تماما كمفهوم التعديل على بعض خصائص الفصيل الاساسى وذلك باستخدام المعرف Override ومن الممكن ان تقوم ببناء Class الاساسى دون عمل تكويد له ثم بعد ذلك تعدل وتقوم بعمل الاكواد فى الفصائل المشتقه

# Abstract Class

ومعناه Class بدون كود وهو مجرد Class عادى لكن دون سطر كدو واحد والهدف

1 – تعريف نسخه اصلية من Class على انها Abstract وهى كلمه محجوزه

واستعمال كلمة Partial معها داخل السى شارب

2 – جميع الدوال والخواص التى تكتب بداخله يجب ان تعطى نفس المعرف وهو

Abstract ولا يجوز تعريف اى متغيرات بداخله او انشاء كائن منه فقط تعريف

الدوال والخواص

لدينا المثال التالي :

abstract class definition

abstract partial class car

```
{
    public abstract int calc(int fuel);
}
```

المعرف

من الملاحظ ان الداله لم يكتي لها كود لكن عند اشتقاق فصيل اخر منه فان المبرمج سيكون مجبرا على عمل تكويد لهذه الداله

abstract partial class House

```
{
    public abstract int roomnumber { set; get; }
    public abstract string label { set; get; }
    public abstract double budgetinmonth { set; get; }
    public abstract int roomnumber(int roomnuber);
    public abstract string houselabel(string label);
    public abstract double housebudget(double budget);
}
```

من الملاحظ ان جميع اعضاء الفصيل اخذو نفس المعرف ولا يوجد اي متغيرات بل خواص ودوال فقط ولا يوجد اي تكويد لهم فقط تعريف لكل منهم

class Program

```
{
    static void Main(string[] args)
    {
    }
}
```

لنرى الان اشتقاق class جديد من السابق ونرى ماذا يكون شكله :

```

}
class myhouse : House
{int roomnumber;
  string label;
  double budgetinmonth;
  public override int Roomnumber
  {
    get{return roomnumber; }
    set{roomnumber = value; }
  }
  public override string Label
  {
    get { return label; }
    set { label = value; }
  }
}

```

من الملاحظ في الفصيل المشتق انه تم تعريف المتغيرات وتم كتابة كود للخواص وبالإمكان أيضا كتابة الكود للدوال ولاحظ أيضا ان استخدمنا المعرف override وذلك للتعديل على اعضاء الفصيل الاساسي مما يدعم نظرية polymorphism التي تحدثنا عنها سابقا يمكنك الان انشاء كائن من هذا الفصيل المشتق والتصرف به داخل البرنامج كما تعلمنا



# Interface

تصميم الواجهات داخل الكود :

وهي بديل عن تعدد الوراثة الهرميه كما انه يتميز بعمل نسخه منه بالوراثة لكن

الاضافه على الفصائل المشتقه تكون بالتعديل على اعضاء Interface

كما انه يمكن عمل تكويد لاكثر من Interface معا والشكل العام له يكون كالتالى :

interface definition

```
Access_modifier interface interface_name
{
  functions;
  properties;
  without any implementation
}
```

ثم تعريف جميع اعضائه لكن دون  
تكويد لها ولاحظ ايضا لا توجد  
متغيرات

ولدينا المثال التالى :



```

interface IPoint
{ int x
  { set; get;}
  int y
  { set; get; } }
class mypoint
{| private int myx;
  private int myy;
  public int x
  { set { x = value; }
    get { return x; }}
  public int y
  { set {y = value ;}
    get {return y;} }

```

بناء الواجهات دون اى توكويد لها  
ودون اى معرفات محدد

اشتقاق فصيل منها ثم عمل التوكويد اللازم  
اى انه يمكن اشتقاق فصيل من الواجهه اى  
انها تقوم بعمل الفصيل

ملحوظه هامه : لا يمكن انشاء كائن من الواجهات او Interface لكن يمكن اشتقاق

فصائل منها وبالتالي انشاء الكائنات من classes المشتقه

لا يمكن تعريف المتغيرات داخل interface بدلا منها خواص ودوال والمتغيرات

تعرف داخل classes المشتقه

ان عمل interface بديل عن الوراثة الهرميه وتشبه فى عملها abstract class

# Types of Errors

: syntax error – 1

هذا هو الاسباب اكتشافه فور كتابة الكود والسبب ان compiler يعترض عليها لانها غير ملائمة للقواعد التي بنيت عليها اللغة او التي تعمل بها داخل لغة معينه

: Logical errors – 2

الاشياء المنطقية وهذه الانواع تظهر عند التنفيذ يكون الكود فيها صحيحا لكن سير البرنامج غير منطقي كالقسمه على الصفر مثلا ويسمى هذا النوع Exception او الاستثناءات

: Bugs – 3

هذه هو النوع الاشهر ولا يوجد برنامج تقريبا يخلو منه وهي نسيان حذف متغير من الذاكره مثلا عند غلق البرنامج ولذلك تكون هناك عدة نسخ تجريبية لاصطياد مثل هذه الاخطاء

الشكل العام بالنسبة Exception داخل الكود :

```

try ← كلمة محجوزة
{
  يوضع الكود الاصيل هنا
}
Catch (exception) ← كلمة معرفه ويوضع تعريف للاستثناء بين القوسين
  كمتغير مثلا
  Exception e
{
  رسالة الخطاء التي تريد اظهارها في
  حالة حدوث الخطاء
  MessageBox.Show("error message");
}
Finally ← كلمة معرفه
{
  what you want if error ot no error;
}

```

ماذا تريد ان تفعل هنا سواء ظهر الخطاء ام لم يظهر  
وعادة ما يكتب هنا اكواد تقوم بحذف المتغيرات من  
الذاكرة

# Collections

هي عبارة عن مخزن او Container للبيانات ويوفر لك مزايا لا توفرها لك

مثيلتها من Array

ان namespace الذي يحتوى فصيل Array List هو Collections

واستدعائه يكون كالتالى :

```
System.Collections;
```

## استدعاء فصيل التجمعات

ويعد فئة Array List هي اهم class فيها وتعرف على انها قائمه من Array لها نفس

سمات المصفوفات من حيث تخزين مجموعه من البيانات دفعه واحده كالمتغيرات

المركبه ولها ايضا مزايا القوائم او List داخل لغة البرمجه فهى اداه اقوى من عمل

المصفوفات

طريقة استخدامه داخل السى شارب :

1 – يجب تعريف كائن منه لاستخدام جميع مزايا class

create object from ArrayList class: **انشاء كائن من الفصيل**  
 ArrayList carlist = new ArrayList(); **ArrayList**

you can add object :  
 carlist.Add(temp); **تستطيع اضافة كائنات بواسطة دالة الاضافه**

carlist.Insert(temp); **وتستطيع عمل ادراج في جزء معين داخل المصفوفه بواسطة دالة الادراج**

carlist.Clear(); **وتستطيع مسح جميع عناصر القائمه دفعه واحده**

carlist.RemoveAt(4); **وتستطيع ازالة عنصر معين برقمه او ازالة الكائن**  
 carlist.RemoveAt(temp) **المخزن فيها**

carlist.Count; **وتستطيع معرفة عدد عناصرها باستخدام دالة العد**

ToArray **وتستطيع تحويلها الى مصفوفه عاديه**

Reverse **وتستطيع عكس ترتيب عناصرها**

IndexOf(temp, 0); **وتستطيع اجراء عملية البحث عن عنصر معين بداخلها**

**العنصر**   
**بداية البحث** 

# Preprocessor Directives

و هذه طريقه لترتيب الكود وجعله ضمن نطاقات معينه لعدم التشويش وتستخدم

لتحسين وتنسيق مظهر الكود داخل ملف الشفرة المصدريه للبرنامج

```
#region "class employee"
public class employee
{
}
#endregion
```

تعريف معين لمنطقه محدد داخل الكود

يكتب هنا كود الفصل كاملا

الاعلان عن نهاية منطقته  
معينه من الكود



# Comments

بناء التعليقات داخل الشفرة المصدرية :

تعتبر التعليقات جزء هام جدا لا يستهان به خاصة داخل المشاريع التي تحتاج الى

شفرة مصدرية ضخمة كقواعد البيانات مثلا ولها فائده كبيره جدا عند الرجوع مره

اخرى الى الكود وعند عمل الفريق الجماعى

XML Comments:

Important elements in comments :

مقطع بداية التعليق وسيكون بنوع خط مختلف عن الكود

<C> the following must be in differnt fonts

المقطع الذى يحتوى على الشفرة المصدرية او الكود

<Code> the following is code

المقطع الذى يحتوى على الاستثناءات او الاخطاء ومعالجتها

<exception> file containg exceptions

المقطع الذى يشرح بارامتر معين داخل الكود ووظيفته

<param> explain parameter in function

عادة لا تستخدم جميعها مره واحده  
فى كل كود تكتبه انما تستخدم  
حسب الحاجه اليها

المقطع الذى يضم بناء فصيل معين

///<Summary> employee class of the company

///</Summary>

المقطع الذى يشرح وظيفة القيمه التى تعود من داله

معينه داخل الكود

<returns>explain the function result

الشرح المختصر لجزء معين من الكود

<Summary>abstract explain to code

المقطع الذى يشرح الخاصيه والقيمه التى تعطيها لمتغير او

<Value>explain property القيمه التى تعود بها



```
{  
  /// <summary>  
  /// class of all operations  
  /// class operations  
  /// <c>  
  /// now we build new class contain on all operation  
  /// we need  
  /// </c>  
  ///<code>  
  ///  
  class Operations  
  {int x, y, z;  
    public int X { ///<value>thie properity well set value to number x </value>  
      set{x = value;}  
      get{return x;} }  
    public int xsquar(int a) {///<param>a well give the value of x</param>  
      x = a;  
      ///<return>function well return squar of value x</return>  
      return x ^ 2; }  
  }  
  ///</code>  
  /// </summary>  
}
```

انتهى الجزء الاول من هذا الكتاب من اساسيات البرمجه فى لغة السى شارب  
اعتمد على المعلومه المباشره واهتم بها اكثر وعذرا على عدم الاهتمام بالتنسيق

memorycode\_84@yahoo.com