

C# 00

Programmeren

LES 2

JOHAN DONNÉ

Overzicht

- Enkele tips
- Properties
- Enums
- Generic collections
- Voorbeeldcode

Enkele tips

Tips

Informatie in code editor:

- Go to definition
- 'Rename' van een identifier
- Refactoring: method extraction
- Automatische layout
- Code Snippets (for, foreach)

Tips: handige informatie in code-editor

Informatie in code editor: references, tests, 'eigenaar', commits

```
public class Scheduler
{
    ▲ Opgave02.Tests\ScheduleTests.cs (6)
    ⌕ 54: var p = sched.Participants;
    ⌕ 96: sched.Participants = list;
    ⌕ 98: var list2 = sched.Participants;
    ⌕ 117: sched.Participants = new List<Participant>
    ⌕ 152: sched.Participants = new List<Participant>
    ⌕ 218: sched.Participants = new List<Participant>
    Show on Code Map | Collapse All

    9 references | 5/5 passing | johan.donne, 2 days ago | 1 author, 1 change
    public List<Participant> Participants { get; set; }

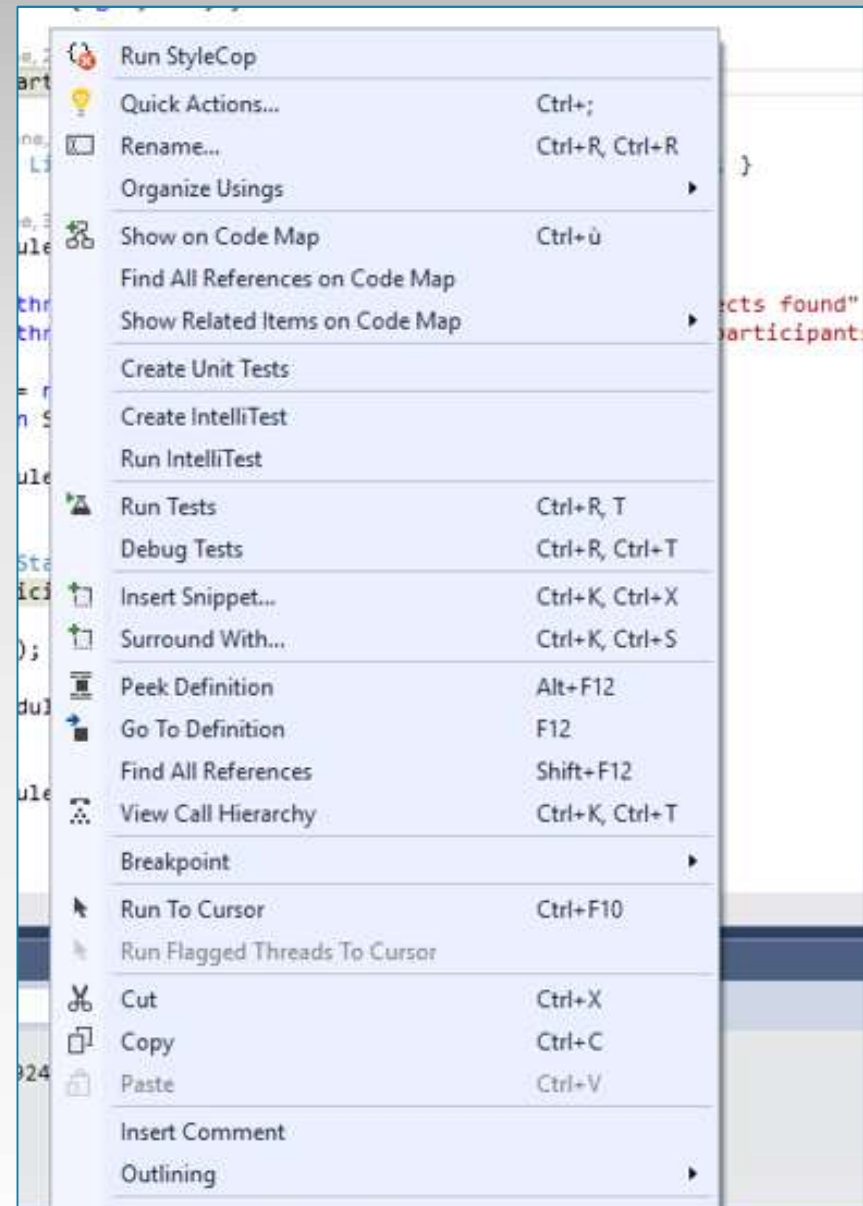
    13 references | 4/4 passing | johan.donne, 6 hours ago | 1 author, 1 change
    public Dictionary<Session, List<Participant>> SessionPersonSchedule { get; set; }

    4 references | 3/3 passing | johan.donne, 3 hours ago | 1 author, 5 changes
    public bool CalculateSchedule()
    {
        if (Sessions == null) throw new ArgumentNullException("Subjects", "No subjects found");
    }
}
```

Tips: info bij identifier

In code-editor: RMB op identifier:

- Go To Definition
- (Go To Implementation)
- Find All References
- Run to cursor
- **Rename !!!**



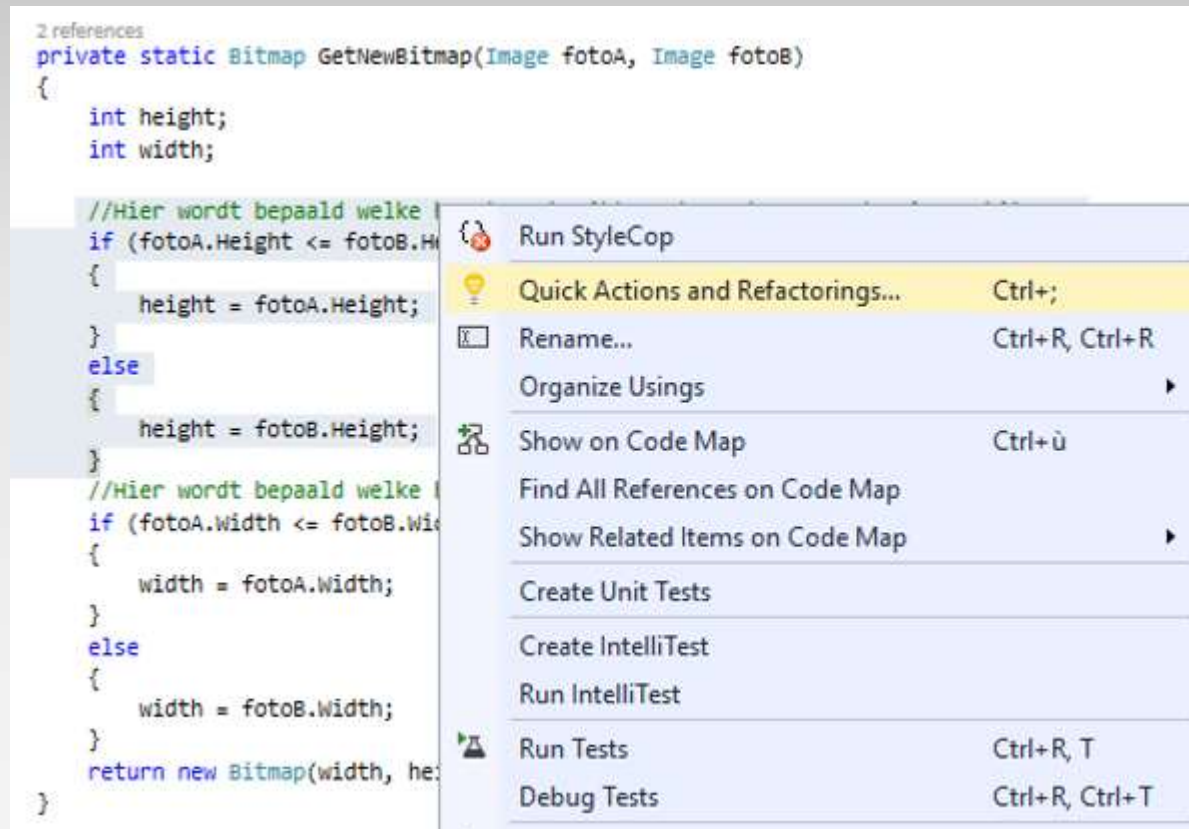
Tips: refactor - extract method

```
2 references
private static Bitmap GetNewBitmap(Image fotoA, Image fotoB)
{
    int height;
    int width;

    //Hier wordt bepaald welke hoogte gebruikt moet worden voor de nieuwe bitmap.
    if (fotoA.Height <= fotoB.Height)
    {
        height = fotoA.Height;
    }
    else
    {
        height = fotoB.Height;
    }
    //Hier wordt bepaald welke breedte gebruikt moet worden voor de nieuwe bitmap.
    if (fotoA.Width <= fotoB.Width)
    {
        width = fotoA.Width;
    }
    else
    {
        width = fotoB.Width;
    }
    return new Bitmap(width, height);
}
```

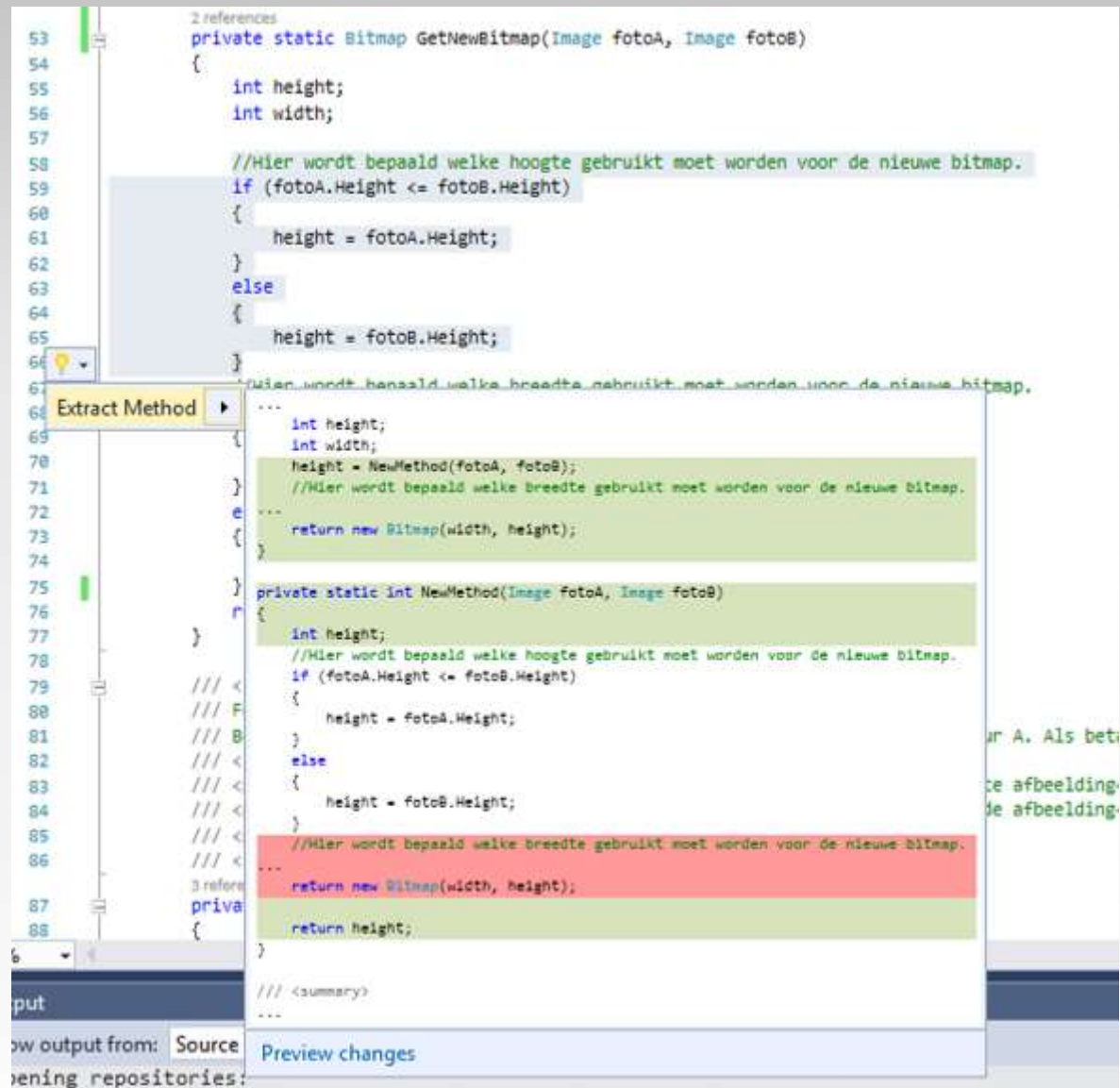
23 lijnen: te veel!

Tips: refactor - extract method



Selecteer blok, RMB, Refactorings

Tips: refactor - extract method



```
2 references
53 private static Bitmap GetNewBitmap(Image fotoA, Image fotoB)
54 {
55     int height;
56     int width;
57
58     //Hier wordt bepaald welke hoogte gebruikt moet worden voor de nieuwe bitmap.
59     if (fotoA.Height <= fotoB.Height)
60     {
61         height = fotoA.Height;
62     }
63     else
64     {
65         height = fotoB.Height;
66     }
67     //Hier wordt bepaald welke breedte gebruikt moet worden voor de nieuwe bitmap.
68     ...
69     int height;
70     int width;
71     height = NewMethod(fotoA, fotoB);
72     //Hier wordt bepaald welke breedte gebruikt moet worden voor de nieuwe bitmap.
73     ...
74     return new Bitmap(width, height);
75 }
76 private static int NewMethod(Image fotoA, Image fotoB)
77 {
78     int height;
79     //Hier wordt bepaald welke hoogte gebruikt moet worden voor de nieuwe bitmap.
80     if (fotoA.Height <= fotoB.Height)
81     {
82         height = fotoA.Height;
83     }
84     else
85     {
86         height = fotoB.Height;
87     }
88     //Hier wordt bepaald welke breedte gebruikt moet worden voor de nieuwe bitmap.
89     ...
90     return new Bitmap(width, height);
91 }
92     return height;
93 }
94
95 /// <summary>
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
```

Extract Method

Source Preview changes

ow output from: Source Preview changes

ening repositories:

Tips: refactor - extract method

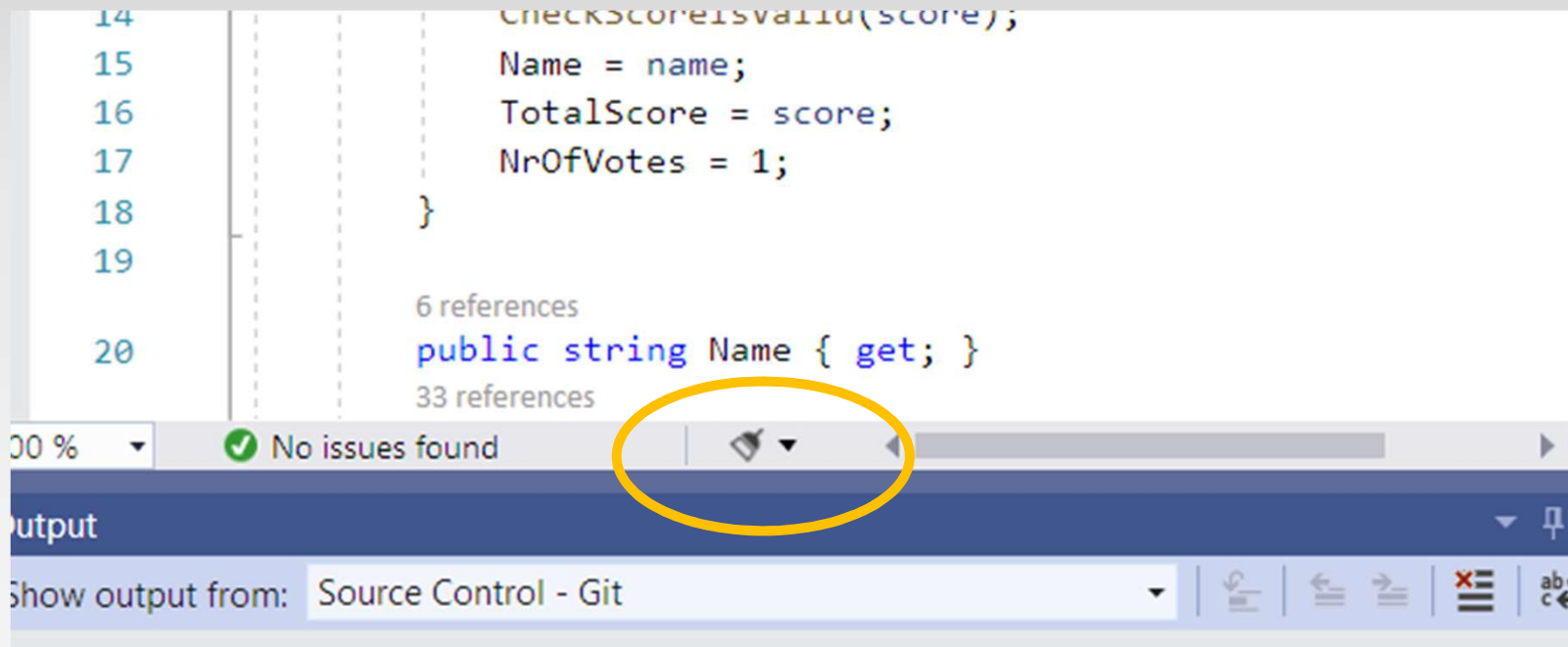
```
2 references
private static Bitmap GetNewBitmap(Image fotoA, Image fotoB)
{
    int height;
    int width;
    height = NewMethod(fotoA, fotoB);
    //Hier wordt bepaald welke breedte gebruikt moet worden voor de nieuwe bitmap.
    if (fotoA.Width <= fotoB.Width)
    {
        width = fotoA.Width;
    }
    else
    {
        width = fotoB.Width;
    }
    return new Bitmap(width, height);
}

1 reference
private static int NewMethod(Image fotoA, Image fotoB)
{
    int height;
    //Hier wordt bepaald welke hoogte gebruikt moet worden voor de nieuwe bitmap.
    if (fotoA.Height <= fotoB.Height)
    {
        height = fotoA.Height;
    }
    else
    {
        height = fotoB.Height;
    }

    return height;
}
```

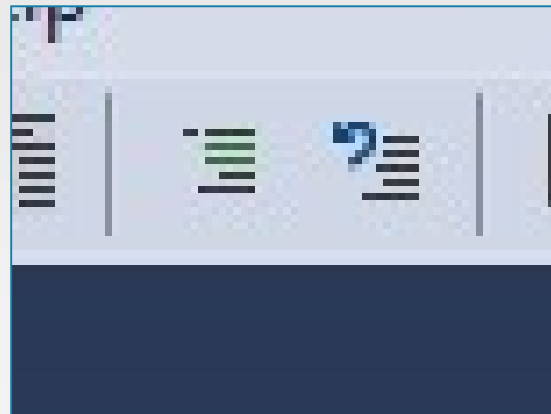
Tips

Automatische layout & cleanup:



Tips

(Un)Comment selection :



Properties

Vorm

'full' property

```
private int leeftijd;  
  
public int Leeftijd  
{  
    get  
    {  
        return leeftijd;  
    }  
    set  
    {  
        if (value < 0) throw new ArgumentException("Ongeldige leeftijd!");  
        leeftijd = value;  
    }  
}
```

Tip

Opgelet bij 'full' properties:

```
private int leeftijd;  
  
public int Leeftijd  
{  
    get  
    {  
        return leeftijd;  
    }  
    set  
    {  
        if (value >= 0) leeftijd = value;  
    }  
}
```

Wat gaat er mis ?

Auto-implemented properties

```
// auto-implemented  
public int Age { get; set; }  
  
// read-only (write-only)  
public int Count { get; }  
  
// met access-modifiers  
public string Name { get; private set; }
```


Expression bodied properties

```
private int number;

// expression bodied property
public int Number
{
    get => number;
    set => number = value;
}

// readonly expression bodied property
public int NumberBis => number;
```

Calculated properties

```
private int a, b;

// expression bodied
public int Som => a + b;

//classic syntax
public int Product
{
    get
    {
        return a * b;
    }
}
```

⇒ Langdurige bewerkingen ==> Methode ipv Property

Enum Types

enum

Het gebeurt regelmatig dat een variabele een beperkt aantal waarden kan aannemen, bv. de seizoenen.

Coderen met bv. een integer (waarden 1,2,3,4)

```
const int Spring = 0;  
const int Summer = 1;  
const int Autumn = 2;  
const int Winter = 3;  
  
int thisSeason = Zomer;  
thisSeason = 5;    // no error !
```

⇒ 'Magic numbers' niet zo'n goed idee!

enum

Het gebeurt regelmatig dat een variabele een beperkt aantal waarden kan aannemen, bv. de seizoenen.

Coderen met bv. een integer (waarden 1,2,3,4)



Zelf een enum type aanmaken met symbolische waarden:

```
public enum Season {Spring, Summer, Autumn, Winter}  
    ...  
Season thisSeason = Season.Summer;
```

⇒ Is eigenlijk een zelf-gedefinieerd type!

enum

Enum type:

- Leesbaarder (self-documenting)
- Betere type checking
- Onderliggend: voorgesteld als Integer type
(onderliggende waarden kunnen ook opgegeven worden)

```
public enum Season
{
    Spring = 1,
    Summer = 2,
    Autumn = 3,
    Winter = 4
}
```

enum

Enkele handige methodes van enum-types:

```
public enum Season {Spring, Summer, Autumn, Winter}  
...  
Season thisSeason = Season.Summer;  
  
Console.WriteLine(s.ToString()); // -> "Autumn"  
  
string[] seasonNames = Enum.GetNames(typeof(Season));  
  
Season season = (Season) Enum.Parse(typeof(Season), "Winter");
```

enum

Enums kunnen soms ook voorgesteld worden door bits zoals de vlaggen in een statusregister:

```
[Flags]
public enum CarAttributes
{
    None = 0,
    FourWheelDrive = 1,
    ManualShift = 2,
    HatchBack = 4
}

CarAttributes attributes = CarAttributes.FourWheelDrive | CarAttributes.ManualShift;
```

De gebruikte waarden moeten machten van twee zijn en kunnen gecombineerd worden via een 'OR' operator.

Generic Collections

Variabelen

- Enkelvoudige types (int, float, bool, char, enums...)
- Samengestelde types:
 - String
met eigen methodes (Substring, Trim, Replace...)
en toegang tot karakters : `char c = zin[i];`
 - Array
toegang tot elementen: `int x = matrix[i,j]`
 - Class
bevat o.a. velden en methodes

Generic collections

- = gespecialiseerde samengestelde datastructuren
- = aantal gemeenschappelijke eigenschappen
(het zijn “collections”, te gebruiken met ‘foreach’)
- = ‘type safe’ : bij declaratie wordt vastgelegd welk type de elementen zullen zijn.

List<T>, Dictionary<Tkey, Tvalue>, HashSet<T>,
SortedSet<T>, Queue<T>, Stack<T> ...

Generic collections: List<T>

⇒ Geen vaste lengte (anders dan bij Array).

```
List<Person> personList = new List<Person>();
```

- Elementen toegankelijk via zero-based Index:

```
Person p = personList[5];
```

- Property 'Count'

```
int nrOfPersons = personList.Count;
```

- Methoden:

```
personList.Add(new Person("John"));
```

```
personList.Remove(p);
```

```
if (personList.Contains(p)) ...
```

```
personList.Clear();
```

Generic collections

```
var personList = new List<Person>()
{
    new Person("John", 21),
    new Person("Paul", 18)
};

int averageAge = 0;
foreach (var person in personList)
{
    averageAge += person.Age;
}
averageAge /= personList.Count;
```

Generic collections

Opgelet:

- 'Contains', 'Remove', 'Find', 'IndexOf': T moet de interface 'IEquatable<T>' implementeren : 'public bool Equals(T other)' methode bevatten.
- 'Sort': T moet de interface IComparable<T> implementeren: 'public int CompareTo(T other)' methode bevatten.

Voorbeeld bij klasse 'Person':

```
public int CompareTo(Person other)
{
    return this.Name.CompareTo(other.Name);
}
```

Generic collections: Dictionary<Tkey, TValue>

```
Dictionary<Person, String> PhoneBook =  
    new Dictionary<Person, string>();  
  
PhoneBook.Add(new Person("Pete", 20), "+400 587 698");  
PhoneBook.Add(person, phoneNr);  
  
if (PhoneBook.ContainsKey(person)) PhoneBook.Remove(person);  
  
Person boss = new Person("Sam", 45);  
string nr = PhoneBook[boss];
```

Opgelet:

- Tkey moet de interface 'IEquatable<T>' implementeren.
- Key mag niet null zijn
- Key mag niet wijzigen
- Geen duplicate Keys

Generic collections

Alle generic Collections: Clear(), Count, foreach

Samenstelling is mogelijk:

```
Dictionary<string, List<int>> LookupTable = new...
```

Tip voor lokale variabelen:

```
var lookupTable = new Dictionary<string, List<int>> ();
```

(dit heet 'type inference')

Combinatie van collections:

Voorbeeld: Dictionary van Lists

```
var scores = new Dictionary<string, List<int>>();  
// Dictionary bestaat nu, maar is nog leeg  
...  
  
if(!scores.ContainsKey(name))  
{  
    scores.Add(name, new List<int>());  
}  
ScoresPunten[name].Add(score);  
  
...
```

Generic collections: Queue & Stack

Queue = FIFO, Stack = LIFO

```
var buffer = new Queue<Person>();  
buffer.Enqueue(new Person("Sam", 45));  
var p = buffer.Dequeue();
```

```
var personStack = new Stack<Person>();  
personStack.Push(new Person("Pete", 20));  
var p = PersonStack.Pop();
```

Voorbeeldcode

Voorbeeldcode: Movie Ranking

Een toepassing voor het bijhouden van een ranking van films waarvoor kijkers kunnen stemmen.

De gebruiker zal meermaals een filmnaam en een score (0..10) kunnen ingeven.

De toepassing toont de ingegeven films gerangschikt van de hoogste naar de laagste totaalscore, telkens met de totale en gemiddelde score, een indicatie van de populariteit en het aantal stemmen dat voor het betreffende film ingegeven is.

