

Inhaltsverzeichnis

Teil I Einleitung und Überblick

1	Motivation	3
1.1	Sind zwei Millisekunden viel?	4
1.1.1	Variante A	6
1.1.2	Variante B	9
1.1.3	Was bremst Variante A?	12
1.2	Eingrenzung	13
2	Messungen und deren Ziel	14
3	Grundlagen der Zeitmessung	18
3.1	Messen von Software	18
3.1.1	QPC und QPF	20
3.1.2	Stopwatch	21
3.2	Irrwege der Zeitmessung	23
3.2.1	Analyse mit dotTrace	26
3.2.2	Analyse mit VisualStudio	28
3.3	Prozessortick?	29
4	Kompilierung: Sourcecode → Exe	31
4.1	Infrastruktur	32
4.2	Schritt für Schritt: Erstellung einer Applikation	33
4.2.1	C# Sourcecode	33
4.2.2	VisualBasic Sourcecode	35
4.2.3	Windows oder Linux, egal!	35
4.3	CLI/CLR vs. Native Applikationen	36
4.4	IL-Code analysieren	37
4.5	IL-Code verstehen	39

Teil II C# Sprachreferenz

5	Einleitung	45
6	Funktionen und Methoden	46
	6.1 Methodenüberladung	48
7	Was sind Klassen und Strukturen?	49
	7.1 public, private, protected, internal	50
	7.2 Strukturen	52
	7.3 Spezialisierung und Klassenvererbung	53
	7.4 Polymorphie: späte Bindung	54
	7.5 Basis von Klassen	59
8	Variablen/Properties und Datentypen	60
	8.1 Elementare Datentypen	60
	8.2 Value Types	62
	8.2.1 Wie entstehen Referenzen?	63
	8.3 Object und String	64
	8.4 Typecast und Datenkonversion	65
	8.5 Boxing und Unboxing	66
9	Ablaufsteuerung	67
	9.1 if-Bedingung	67
	9.2 if-else-Bedingung	70
	9.3 switch-case-Bedingung	72
	9.4 for-Schleife – for-Loop	75
	9.5 foreach-Schleife	78
	9.6 while-Schleife	79
	9.7 do-while-Schleife	80
10	Arrays, Collection und Enumerables	82
	10.1 Arrays	82
	10.2 Collection oder Auflistungen	83
	10.2.1 List<T>	85
	10.2.2 Dictionary<TKey, TValue>	86
	10.2.3 Stack<T>	87
	10.2.4 Queue<T>	88
	10.2.5 HashSet<T>	89
	10.2.6 LinkedList<T>	92
	10.3 Enumerables	94
	10.3.1 Von einer Klasse, zu einer Collection	98
	10.3.2 Minimalaufwand für eine Collection	100
	10.3.3 yield	101

11 Polymorphie im Detail	107
11.1 IL-Code	108
11.1.1 Ermittlung der richtigen Methode	109
11.1.2 <code>callvirt</code> auch ohne Polymorphie	110
12 Asynchrone Methoden mit Async und Await	112
12.1 Möglichkeiten der Verwendung	112
12.2 IL-Code	113
13 Exceptions	119
Teil III Speicherverbrauch und Performance	
14 Garbage Collector	125
14.1 Heap und Stack	127
14.2 Analyse des Heap	130
14.3 Level 0, Level 1, Level 2 und Level N	135
14.3.1 Zeitaufwand des Garbage Collectors	138
14.3.2 GC-Wanderschaft von Objekten	142
15 Call by Value, Call by Reference	143
15.1 Performance: <i>Call by Value/Call by Reference</i>	144
16 Initialisierungen von Klassen und Strukturen	149
16.1 Analyse der Teilergebnisse	150
17 Behandlung von Zeichenketten	155
17.1 Zeichenketten verbinden	156
18 Schleifen	159
18.1 Parallelisierung von Schleifen	161
18.2 <code>Parallel.Invoke()</code>	165
18.3 Irrtümer der Schleifenparallelisierung	165
19 Zeitaufwand bei Collections	167
19.1 <code>List<T></code> → <code>Add()</code> und <code>Contains()</code>	168
19.2 <code>HashSet<T></code> → <code>Add()</code> und <code>Contains()</code>	170
19.3 <code>Dictionary<T></code> → <code>Add()</code> und <code>Contains*</code>	172

Teil IV Anhang

Abbildungsverzeichnis	185
Tabellenverzeichnis	186
Literaturverzeichnis	188
Sachverzeichnis	190