

Table of Contents

Table of Contents

1	Autoformation C# - préparation du passage de test de certification 70-483.....	11
1.1	Questions, Réponses, Explications	11
1.1.1	Question 1 Objective : Create and Use Types. Subobjective : Enforce encapsulation.	11
1.1.1.1	Correction	12
1.1.2	Question 2 Objective : Implement Data Access. Subobjective : Serialize and deserialize data.....	13
1.1.2.1	Correction	14
1.1.3	Question 3 Objective : Create and Use Types. Subobjective : Enforce encapsulation.	15
1.1.3.1	Correction :	17
1.1.4	Question 4. Objective : Implement Data Access. Subobjective : Perform I/O operations.	20
1.1.4.1	Correction	21
1.1.5	Question 5 Objective : Debug Applications and Implement Security. Subobjective : Manage assemblies.	
	22	
1.1.5.1	Correction	23
1.1.6	Question 6 Objective : Manage Program Flow Subobjective : Create and Implement event and callbacks.	
	25	
1.1.6.1	Correction	26
1.1.7	Question 7 Objective : Manage Program Flow. Subobjective :Implement program flow.....	27
1.1.7.1	Correction	29
1.1.8	Question 8 Objective : Create and Use Types. Subobjective : Create types.....	31
1.1.8.1	Correction	32
1.1.9	Question 9 Objective : Create and Use Types Subobjective : Create types.....	33
1.1.9.1	Correction	35
1.1.10	Question 10 Objective : Create and Use Types Subobjective : Find, execute, and create types at runtime by using reflexion.	37
1.1.10.1	Correction :	38
1.1.11	Question 11 Objective : Implement Data Access Subobjective : Query and manipulate data aned objects by using LINQ	39
1.1.11.1	Correction	40
1.1.12	Question 12 Objective : Create and Use Types. Subobjective : Consume types.....	41
1.1.12.1	Correction	42
1.1.13	Question 13 Objective : Implement Data Access Subobjective : Store data and retreive data from collections.....	44
1.1.13.1	Correction	45

1.1.14 Question 14 Objective : Implement Data Access Subobjective : Query and manipulate data and objects by using LINQ	47
1.1.14.1 Correction	48
1.1.15 Question 15 Objective : Debug Applications and Implement Security Subobjective : Validate application Input	49
1.1.15.1 Correction	50
1.1.16 Question 16 Objective : Create and Use Type Subobjective : Enforce encapsulation.....	51
1.1.16.1 Correction	53
1.1.17 Question 17 Objective : Implement Data Access Subobjective : Perform I/O operations.	55
1.1.17.1 Correction :	56
1.1.18 Question 18 Objective : Create and Use Type Subobjective : Consume types	58
1.1.18.1 Correction	60
1.1.19 Question 19 Objective : Create and Use Types Subobjective : Consume types.....	63
1.1.19.1 Correction	64
1.1.20 Question 20 Objective : Implement Data Access Subobjective : Serialize and deserialize data	66
1.1.20.1 Correction	67
1.1.21 Question 21 Objectif : Debug Application and Implement Security Subobjectif : Perform symmetric and asymmetric encryption	68
1.1.21.1 Correction	69
1.1.22 Question 22 Objectif : Debug Application and Implement Security Subobjectif : Validate application input	70
1.1.22.1 Correction	71
1.1.23 Question 23 Objective : Create and Use Types Subobjective : Enforce encapsulation.....	72
1.1.23.1 Correction	74
1.1.24 Question 24 Objective : Create and Use Types Subobjective : Enforce encapsulation.....	77
1.1.24.1 Correction	78
1.1.25 Question 25 Objective : Create and Use Types Subobjective : Create types	80
1.1.25.1 Correction	81
1.1.26 Question 26 Objective : Create and Use Types Subobjective : Enforce encapsulation.....	82
1.1.26.1 Correction	83
1.1.27 Question 27 Objective : Create and Use Types Subobjective : Enforce encapsulation.....	85
1.1.27.1 Correction	87
1.1.28 Question 28 Objective : Create and Use Types Subobjective : Create and implement a class hierarchy	90
1.1.28.1 Correction	91
1.1.29 Question 29 Objective : Manage Program Flow Subobjective : Implement program flow	93
1.1.29.1 Correction	95
1.1.30 Question 30 Objective : Manage Program Flow Subobjective : Implement exception handling	97
1.1.30.1 Correction	99
1.1.31 Question 31 Objective : Debug Application and Implement Security Subobjective : Manage assemblies.....	101
1.1.31.1 Correction	102

1.1.32 Question 32 Objective : Debug Application and Implement Security Subobjective : Debug an application	104
1.1.32.1 Correction	105
1.1.33 Question 33 Objective : Debug Application and Implement Security Subobjective : Perform symmetric and asymmetric encryption	106
1.1.33.1 Correction	107
1.1.34 Question 34 Objective : Debug Application and Implement Security Subobjective : Debug an application	108
1.1.34.1 Correction	109
1.1.35 Question 35 Objective : Create and Use Types Subobjective : Manipulate strings	110
1.1.35.1 Correction	111
1.1.36 Question 36 Objective : Manage Program Flow Subobjective : Implement exception handling	113
1.1.36.1 Correction	114
1.1.37 Question 37 Objective : Manage Program Flow Subobjective : Create and implement events and callbacks.	116
1.1.37.1 Correction	117
1.1.38 Question 38 Objective : Debug Application and Implement Security. Subobjective : Perform symmetric and asymmetric encryption.	118
1.1.38.1 Correction	119
1.1.39 Question 39 Objective : Create and Use Types Suobjective : Create and implement a class hierarchy	120
1.1.39.1 Correction	121
1.1.40 Question 40 Objective : Manage Program Flow Subobjective : Manage multithreading	122
1.1.40.1 Correction	124
1.1.41 Question 41 Objective : Manage Program Flow Subobjective : Create and implement events and callbacks.	127
1.1.41.1 Correction	128
1.1.42 Question 42 Objective : Manage Program Flow Subobjective : Create and implement events and callbacks.	129
1.1.42.1 Correction	130
1.1.43 Question 43 Objective : Manage Program Flow Subobjective : Implement exception handling	131
1.1.43.1 Correction	132
1.1.44 Question 44 Objective : Manage Program Flow Subobjective : Implement exception handling	133
1.1.44.1 Correction	134
1.1.45 Question 45 Objective : Manage Program Flow Subobjective : Implement exception handling	135
1.1.45.1 Correction	136
1.1.46 Question 46 Objective : Manage Program Flow Subobjective : Implement exception handling	137
1.1.46.1 Correction	139
1.1.47 Question 47 Objective : Manage Program Flow Subobjective : Implement exception handling	140
1.1.47.1 Correction	142
1.1.48 Question 48 Objective : Manage Program Flow Subobjective : Implement program flow	144
1.1.48.1 Correction	145

1.1.49 Question 49 Objective : Manage Program Flow Subobjective : Implement program flow	147
1.1.49.1 Correction	148
1.1.50 Question 50.....	149
1.1.50.1 Correction	150
1.1.51 Question 51 Objective Manage Program Flow Subobjective : Implement program flow	151
1.1.51.1 Correction	152
1.1.52 Question 52 Objective Manage Program Flow Subobjective : Implement program flow	153
1.1.52.1 Correction	154
1.1.53 Question 53 Objective : Manage Program Flow Subobjective : Implement program flow	155
1.1.53.1 Correction	156
1.1.54 Question 54 Objective : Manage Program Flow Subobjective : Implement program flow	157
1.1.54.1 Correction	159
1.1.55 Question 55 Objective : Manage Program Flow Subobjective : Implement program flow	161
1.1.55.1 Correction	163
1.1.56 Question 56.....	165
1.1.56.1 Correction	166
1.1.57 Question 57 Objective : Manage Program Flow Subobjective : Implement program flow	167
1.1.57.1 Correction	169
1.1.58 Question 58 Objective : Manage Program Flow Subobjective : Implement multithreading and asynchronous processing.....	171
1.1.58.1 Correction	172
1.1.59 Question 59 Objective : Manage Program Flow Subobjective : Implement multithreading and asynchronous processing.....	173
1.1.59.1 Correction	174
1.1.60 Question 60 Objective : Manage Program Flow Subobjective : Implement multithreading and asynchronous processing.....	175
1.1.60.1 Correction	176
1.1.61 Question 61 Objective : Manage Program Flow Subobjective : Implement multithreading and asynchronous processing.....	177
1.1.61.1 Correction	178
1.1.62 Question 62 Objective : Manage Program Flow Subobjective : Implement multithreading and asynchronous processing.....	180
1.1.62.1 Correction	180
1.1.63 Question 63 Objective : Manage Program Flow Subobjective : Implement multithreading and asynchronous processing.....	182
1.1.63.1 Correction	184
1.1.64 Question 64 Objective : Manage Program Flow Subobjective : Implement multithreading and asynchronous processing.....	186
1.1.64.1 Correction	187
1.1.65 Question 65 Objective : Manage Program Flow Subobjective : Manage multithreading	189
1.1.65.1 Correction	191
1.1.66 Question 66 Objective : Manage Program Flow Subobjective : Manage multithreading	193

1.1.66.1	Correction	194
1.1.67	Question 67 Objective : Manage Program Flow Subobjective : Manage multithreading	195
1.1.67.1	Correction	196
1.1.68	Question 68 Objective : Manage Program Flow Subobjective : Manage multithreading	197
1.1.68.1	Correction	198
1.1.69	Question 69 Objective : Manage Program Flow Subobjective : Manage multithreading	199
1.1.69.1	Correction	200
1.1.70	Question 70 Objective : Manage Program Flow Subobjective : Manage multithreading	201
1.1.70.1	Correction	202
1.1.71	Question 71 Objective : Manage Program Flow Subobjective : Manage multithreading	203
1.1.71.1	Correction	205
1.1.72	Question 72 Objective : Manage Program Flow Subobjective : Manage multithreading	208
1.1.72.1	Correction	210
1.1.73	Question 73 Objective : Create and Use Types Subobjective : Find, execute, and create at runtime by using reflection.	213
1.1.73.1	Correction	214
1.1.74	Question 74 Objective : Create and Use Types Subobjective : Find, execute, and create at runtime by using reflection.	216
1.1.74.1	Correction	217
1.1.75	Question 75 Objective : Create and Use Types Subobjective : Find, execute, and create at runtime by using reflection.	218
1.1.75.1	Correction	219
1.1.76	Question 76 Objective : Create and Use Types Subobjective : Find, execute, and create at runtime by using reflection.	220
1.1.76.1	Correction	221
1.1.77	Question 77 Objective : Create and Use Types Subobjective : Enforce encapsulation.....	222
1.1.77.1	Correction	223
1.1.78	Question 78 Objective : Manage Program Flow Subobjective : Manage multithreading	224
1.1.78.1	Correction	226
1.1.79	Question 79 Objective : Manage Program Flow Subobjective : Manage multithreading	229
1.1.79.1	Correction	230
1.1.80	Question 80 Objective : Debug Applications and Implement Security Subobjective : Implement diagnostics in an application	231
1.1.80.1	Correction	232
1.1.81	Question 81 Objective : Debug Applications and Implement Security Subobjective : Validate application input.....	234
1.1.81.1	Correction	235
1.1.82	Question 82 Objective : Create and Use Types Subobjective : Enforce encapsulation.....	236
1.1.82.1	Correction	237
1.1.83	Question 83 Objective : Implement Data Access Subobjective : Query and manipulate data and object by using LINQ.....	238
1.1.83.1	Correction	240

1.1.84 Question 84 Objective : Debug Application and Implement Security Subobjective : Perform symmetric and asymmetric encryption.	242
1.1.84.1 Correction	244
1.1.85 Question 85 Objective : Create and Use Types Subobjective : Manage the object life cycle	245
1.1.85.1 Correction	246
1.1.86 Question 86 Objective : Create and Use Types Subobjective : Manage the object life cycle	248
1.1.86.1 Correction	250
1.1.87 Question 87 Objective : Create and Use Types Subobjective : Manage the object life cycle	251
1.1.87.1 Correction	253
1.1.88 Question 88 Objective : Create and Use Types Subobjective : Manage the object life cycle	255
1.1.88.1 Correction	256
1.1.89 Question 89 Objective : Create and Use Types Subobjective : Manage the object life cycle Example: WeakReference.....	258
1.1.89.1 Correction	259
1.1.90 Question 90 Objective : Create and Use Types Subobjective : Manage the object life cycle	260
1.1.90.1 Correction	261
1.1.91 Question 91 Objective : Create and Use Types Subobjective : Manage the object life cycle	263
1.1.91.1 Correction	264
1.1.92 Question 92 Objective : Create and Use Types Subobjective : Manipulate string	265
1.1.92.1 Correction	267
1.1.93 Question 93 Objective : Create and Use Types Subobjective : Manipulate string	270
1.1.93.1 Correction	271
1.1.94 Question 94 Objective : Create and Use Types Subobjective : Manipulate string	273
1.1.94.1 Correction	274
1.1.95 Question 95 Objective : Create and Use Types Subobjective : Create and implement a class hierarchy	276
1.1.95.1 Correction	278
1.1.96 Question 96 Objective : Create and Use Types Subobjective : Create and implement a class hierarchy	280
1.1.96.1 Correction	282
1.1.97 Question 97 Objective : Create and Use Types Subobjective : Create and implement a class hierarchy	284
1.1.97.1 Correction	285
1.1.98 Question 98 Objective : Create and Use Types Subobjective : Create types	286
1.1.98.1 Correction	287
1.1.99 Question 99 Objective : Create and Use Types Subobjective : Create types	288
1.1.99.1 Correction	289
1.1.100 Question 100 Objective : Create and Use Types Subobjective : Create types	291
1.1.100.1 Correction	292
1.1.101 Question 101 Objective: Create and Use Types Subobjective: Create types	295
1.1.101.1 Correction	296
1.1.102 Question 102 Objective : Create and Use Types Subobjective : Create types	297

1.1.102.1	Correction	298
1.1.103	Question 103 Objective : Create and Use Types Subobjective : Create types	299
1.1.103.1	Correction	300
1.1.104	Question 104 Objective : Create and Use Types Subobjective : Consume types.....	302
1.1.104.1	Correction	303
1.1.105	Question 105 Objective : Create and Use Types Subobjective : Consume types.....	304
1.1.105.1	Correction	305
1.1.106	Question 106 Objective : Create and Use Types Subobjective : Enforce encapsulation.....	306
1.1.106.1	Correction	307
1.1.107	Question 107 Objective : Debug Application and Implement Security Subobjective : Validate application input.....	309
1.1.107.1	Correction	310
1.1.108	Question 108 Objective : Debug Application and Implement Security Subobjective : Validate application input.....	311
1.1.108.1	Correction	312
1.1.109	Question 109 Objective : Debug Application and Implement Security Subobjective : Validate application input.....	313
1.1.109.1	Correction	314
1.1.110	Question 110 Objective : Debug Application and Implement Security Subobjective : Validate application input.....	315
1.1.110.1	Correction	316
1.1.111	Question 111 Objective : Debug Application and Implement Security Subobjective : Debug an application	317
1.1.111.1	Correction	318
1.1.112	Question 112 Objective : Debug Application and Implement Security Subobjective : Debug an application	319
1.1.112.1	Correction	320
1.1.113	Question 113 Objective : Debug Application and Implement Security Subobjective : Perform symmetric and asymmetric encryption	321
1.1.113.1	Correction	322
1.1.114	Question 114 Objective : Debug Application and Implement Security Subobjective : Perform symmetric and asymmetric encryption	323
1.1.114.1	Correction	324
1.1.115	Question 115 Objective : Create and Use Types Subobjective : Create and implement a class hierarchy	325
1.1.115.1	Correction	326
1.1.115.2	Feedback sent.....	326
1.1.116	Question 116 Objective : Create and Use Types Subobjective : Create types	328
1.1.116.1	Correction	329
1.1.117	Question 117 Objective : Create and Use Types Subobjective : Create types.	330
1.1.117.1	Correction	331
1.1.118	Question 118 Objective : Create and Use Types Subobjective : Create types.	333
1.1.118.1	Correction	334

1.1.119 Question 119 Objective : Create and Use Types Subobjective : Consume types.....	335
1.1.119.1 Correction	336
1.1.120 Question 120 Objective : Create and Use Types Subobjective : Consume types.....	338
1.1.120.1 Correction	339
1.1.121 Question 121 Objective : Create and Use Types Subobjective : Consume types.....	341
1.1.121.1 Correction	342
1.1.122 Question 122 Objective : Create and Use Types Subobjective : Find, execute, and create types at runtime by using reflection	344
1.1.122.1 Correction	345
1.1.123 Question 123 Objective : Create and Use Types Subobjective : Enforce encapsulation.....	346
1.1.123.1 Correction	347
1.1.124 Question 124 Objective : Create and Use Types Subobjective : Enforce encapsulation.....	349
1.1.124.1 Correction	350
1.1.124.2 Question 125 Objective : Debug Applications and Implement Security Subobjective : Perform symmetric and asymmetric encryption.....	351
1.1.124.3 Correction	352
1.1.125 Question 126 Objective : Debug Applications and Implement Security Subobjective : Perform symmetric and asymmetric encryption.....	353
1.1.125.1 Correction	354
1.1.126 Question 127 Objective : Debug Applications and Implement Security Subobjective : Manage assemblies.....	355
1.1.126.1 Correction	356
1.1.127 Question 128 Objective : Debug Applications and Implement Security Subobjective : Manage assemblies.....	357
1.1.127.1 Correction	358
1.1.128 Question 129 Objective : Debug Applications and Implement Security Subobjective : Manage assemblies.....	359
1.1.128.1 Correction	360
1.1.129 Question 130 Objective : Debug Applications and Implement Security Subobjective : Manage assemblies.....	362
1.1.129.1 Correction	363
1.1.130 Question 131 Objective : Debug Applications and Implement Security Subobjective : Manage assemblies.....	364
1.1.130.1 Correction	365
1.1.131 Question 132 Objective : Debug Applications and Implement Security Subobjective : Implement diagnostics in an application.	366
1.1.131.1 Correction	367
1.1.132 Question 133 Objective : Debug Applications and Implement Security Subobjective : Implement diagnostics in an application.	368
1.1.132.1 Correction	369
1.1.133 Question 134 Objective : Debug Applications and Implement Security Subobjective : Implement diagnostics in an application.	370
1.1.133.1 Correction	371

1.1.134 Question 135 Objective : Debug Applications and Implement Security Subobjective : Implement diagnostics in an application	372
1.1.134.1 Correction	373
1.1.135 Question 136 Objective : Debug Applications and Implement Security Subobjective : Implement diagnostics in an application	374
1.1.135.1 Correction	375
1.1.136 Question 137 Objective : Implement Data Access Subobjective Query and manipulate data and object by using LINQ	376
1.1.136.1 Correction	377
1.1.137 Question 138 Objective : Implement Data Access Subobjective Query and manipulate data and object by using LINQ	378
1.1.137.1 Correction	379
1.1.138 Question 139 Objective : Implement Data Access Subobjective Query and manipulate data and object by using LINQ	380
1.1.138.1 Correction	381
1.1.139 Question 140 Objective : Implement Data Access Subobjective Query and manipulate data and object by using LINQ	382
1.1.139.1 Correction	383
1.1.140 Question 141 Objective : Implement Data Access Subobjective Query and manipulate data and object by using LINQ	384
1.1.140.1 Correction	385
1.1.141 Question 142 Objective : Implement Data Access Subobjective Store data and retrieve data from collections	386
1.1.141.1 Correction	387
1.1.142 Question 143 Objective : Implement Data Access Subobjective Store data and retrieve data from collections	388
1.1.142.1 Correction	389
1.1.143 Question 144 Objective : Implement Data Access Subobjective Store data and retrieve data from collections	390
1.1.143.1 Correction	391
1.1.144 Question 145 Objective : Implement Data Access Subobjective Store data and retrieve data from collections	392
1.1.144.1 Correction	394
1.1.145 Question 146 Objective : Implement Data Access Subobjective Consume data	395
1.1.145.1 Correction	397
1.1.146 Question 147 Objective : Implement Data Access Subobjective Consume data	399
1.1.146.1 Correction	400
1.1.147 Question 148 Objective : Implement Data Access Subobjective Consume data	401
1.1.147.1 Correction	402
1.1.148 Question 149 Objective : Implement Data Access Subobjective Consume data	403
1.1.148.1 Correction	404
1.1.149 Question 150 Objective : Implement Data Access Subobjective Consume data	405
1.1.149.1 Correction	406

1.1.150	Question 151 Objective : Implement Data Access Subobjective Consume data.....	407
1.1.150.1	Correction	408
1.1.151	Question 152 Objective : Implement Data Access Subobjective Serialize and deserialize data.	410
1.1.151.1	Correction	412
1.1.152	Question 153 Objective : Implement Data Access Subobjective Consume data.....	414
1.1.152.1	Correction	415
1.1.153	Question 154 Objective : Implement Data Access Subobjective Consume data.....	416
1.1.153.1	Correction	417
1.1.154	Question 155 Objective : Implement Data Access Subobjective Perform I/O operations.	418
1.1.154.1	Correction	419
1.1.155	Question 156 Objective : Implement Data Access Subobjective Perform I/O operations.	420
1.1.155.1	Correction	421
1.1.156	Question 157 Objective : Implement Data Access Subobjective Perform I/O operations.	423
1.1.156.1	Correction	424
1.1.157	Question 158 Objective : Implement Data Access Subobjective Perform I/O operations.	426
1.1.157.1	Correction	427
1.1.158	Question 159 Objective : Implement Data Access Subobjective Perform I/O operations.	428
1.1.158.1	Correction	429
1.1.159	Question 160 Objective : Implement Data Access Subobjective Perform I/O operations.	430
1.1.159.1	Correction	431

1 Autoformation C# - préparation du passage de test de certification 70-483

1.1 Questions, Réponses, Explications

1.1.1 Question 1 Objective : Create and Use Types. Subobjective : Enforce encapsulation.

You are creating a C# application by using Visual Studio 2015. You are designing a class to manage customers and you have written the code below. (Line numbers are included for reference only.)

```
01  class Customer
02 {
03     string account;
04     bool verified;
05     string Account()
06     {
07         if(Verified)
08         {
09             return account;
10         }
11         return "";
12     }
13     bool Verified
14     {
15         get { return verified; }
16         set { verified = value; }
17     }
18 }
```

The class must meet the following requirements:

- * Assemblies external to the file this class is defined in should not be able to access the class or its members.
- * The member variables can only be accessed by methods in the Customer class.
- * The Account property can be exposed by any instance of the Customer class and classes that derive from it.
- * The Verified property can only be accessed from methods in the Customer class or classes that derive from it.

Which accessors should you apply to the code? Select each accessor and place it next to the correct code line number.

Drag and drop the answers

Accessor	Line Number
	Line 01
	Line 03
	Line 04
	Line 05
	Line 13

protected

public

internal

private

1.1.1.1 Correction

Drag and drop the answers

Accessor	Line Number
internal	Line 01
private	Line 03
private	Line 04
public	Line 05
protected	Line 13

protected	public	internal	private
-----------	--------	----------	---------

^ Explanation

You should choose internal for line 01. A class prefixed by the internal access modifier can only be accessed from the file the class is defined in. This means that external assemblies and classes defined in other files will not be able to instantiate this class.

You should choose private for line 03. A private member variable can only be accessed by methods in the class it is defined in. This means that it will not be exposed to instances of the class, and it will not be accessible from classes that derive from this class.

You should choose private for line 04. A private member variable can only be accessed by methods in the class it is defined in. This means that it will not be exposed to instances of the class, and it will not be accessible from classes that derive from this class.

You should choose public for line 05. A public variable is exposed to instances of the class it is defined in as well as any classes that derive from this class. This allows programs to have direct access to the Account property.

You should choose protected for line 13. A protected variable can be accessed by methods in the class it was defined in or any classes that derive from that class, but it is not exposed to instances of those variables.

Objective:

Create and Use Types

Subobjective:

Enforce encapsulation.

References

MSDN

Microsoft

Copyright Year: 2013

internal (C# Reference)

[Click here for more information](#)

MSDN

Microsoft

Copyright Year: 2013

1.1.2 Question 2 Objective : Implement Data Access. Subobjective : Serialize and deserialize data.

You use Microsoft .NET Framework 4.5 to create an application. This class exists:

```
[DataContract]
public class Customer
{
    [DataMember]
    public string Name {get; set;}
    [DataMember]
    public Account Account {get; set;}
}
```

The Account class is defined in a separate assembly over which you have no control.

You write this code:

```
Customer customer = new Customer();
customer.Name = "Jane Doe";
customer.Account = GetAccount();
```

```
DataContractJsonSerializer json = new DataContractJsonSerializer(typeof(Customer));
MemoryStream stream = new MemoryStream();
json.WriteObject(stream, customer);
```

When the code executes, the properties of the Account class do not get serialized.

You need to ensure that properties of the Account class get serialized.

What should you do?

Choose the correct answer

- Create a class that implements `IDataContractSurrogate` and pass an instance of this class to the `DataContractJsonSerializer` constructor.
- Create a new method in the `Customer` class and apply the `OnSerialized` attribute to the method.
- Modify the `DataMember` attribute of the `Account` property as follows:
`[DataMember(IsRequired=true)]`
- Create a new method in the `Customer` class and apply the `OnSerializing` attribute to the method.

1.1.2.1 Correction

Choose the correct answer



Create a new method in the Customer class and apply the OnSerializing attribute to the method.



Create a class that implements IDataContractSurrogate and pass an instance of this class to the DataContractJsonSerializer constructor.



Modify the DataMember attribute of the Account property as follows:
[DataMember(IsRequired=true)]



Create a new method in the Customer class and apply the OnSerialized attribute to the method.

Explanation

You should create a class that implements IDataContractSurrogate and pass an instance of this class to the DataContractJsonSerializer constructor. The IDataContractSurrogate interface allows you to override serialization specified by theDataContract and DataMember attributes. This is necessary if you need to serialize a member whose type is not marked withDataContract and DataMember attributes. In this scenario, the Account class is defined in a separate assembly. When you implement the

IDataContractSurrogate interface, you specify a surrogate type that is used during serialization and deserialization. For example, you can define a new class named AccountSurrogate that has all the same members as the Account class. You can mark the members of the AccountSurrogate class with the DataMember attribute. The IDataContractSurrogate interface can use this class when serializing and deserializing Account instances.

You should not modify the DataMember attribute of the Account property by setting the IsRequired property to true. This does not solve the problem. This property instructs the serialization engine that a value must be present when the object is being deserialized. In this scenario, the code is serializing an object, not deserializing it.

You should not create a new method in the Customer class and apply the OnSerializing attribute to the method. The method with the OnSerializing attribute applied gets called when a class is about to be serialized. This allows you to make changes to class members before they get serialized. This does not solve the problem in this scenario because the problem is that the Account property does not get serialized.

You should not create a new method in the Customer class and apply the OnSerialized attribute to the method. The method with the OnSerialized attribute applied gets called after a class has been serialized. This allows you to make changes to class members immediately after serialization. This does not solve the problem in this scenario because the problem is that the Account property does not get serialized.

Objective:

Implement Data Access

Subobjective:

Serialize and deserialize data.

1.1.3 Question 3 Objective : Create and Use Types. Subobjective : Enforce encapsulation.

You are creating a C# application by using Visual Studio 2015. You are creating a class to handle customer accounts. The class must meet the following requirements:

- * The member variables, id, and balance can only be accessed by methods inside the class.
- * A property should be defined to allow instances of the class to read, but not write the id.
- * A property should be defined to allow instances of the class to read and write the balance.

Which block of code should you write?

Choose the correct answer

```
public class Account
{
    private int id;
    private float balance;
    public int Id
    {
        get { return id; }
    }
    protected float Balance
    {
        get { return balance; }
    }
}
```



```
public class Account
{
    public int id;
    public float balance;
    public int Id
    {
        get { return id; }
    }
    public float Balance
    {
        set { balance = value; }
        get { return balance; }
    }
}
```



```
public class Account
{
    private int id;
    private float balance;
    public int Id
    {
        get { return id; }
    }
    public float Balance
```



```
    {
        set { balance = value; }
        get { return balance; }
    }
}
```

public class Account

```
{
    private int id;
    private float balance;
    public int Id
    {
        set { id = value; }
        get { return id; }
    }
    protected float Balance
    {
        set { balance = value; }
        get { return balance; }
    }
}
```



1.1.3.1 Correction :

Choose the correct answer

```
public class Account
{
    private int id;
    private float balance;
    public int Id
    {
        get { return id; }
    }
    public float Balance
    {
        set { balance = value; }
        get { return balance; }
    }
}
```



Explanation

You should write the code below:

```
public class Account
{
    private int id;
    private float balance;
    public int Id
    {
        get { return id; }
    }
    public float Balance
    {
        set { balance = value; }
        get { return balance; }
    }
}
```

This code properly uses the private access modifier so that the member variables id and balance are only accessible by methods of the Account class. It uses the public access modifier so that the Id and Balance properties can be accessed by instances of the class. Id only has a get, making it read-only. Balance has both a get and set, making it read and write.

You should not write the code below:

```
public class Account
{
    private int id;
    private float balance;
    public int Id
    {
        set { id = value; }
        get { return id; }
    }
    protected float Balance
    {
        set { balance = value; }
        get { return balance; }
    }
}
```

This class is almost coded correctly. However, the `Id` property has both a `get` and a `set`, making it read and write. It should only have a `get`, making it read-only instead.

You should not use the code below:

```
public class Account
{
    public int id;
    public float balance;
    public int Id
    {
        get { return id; }
    }
    public float Balance
    {
        set { balance = value; }
        get { return balance; }
    }
}
```

This class is almost coded correctly. The member variables `id` and `balance` are coded as `public`, which makes them accessible to instances of the class as well as classes that derive from this class. Instead, these variables should be prefixed with `private` so that they are only accessible by methods of the `Account` class.

You should not use the code below:

```
public class Account
{
    private int id;
    private float balance;
    public int Id
    {
        get { return id; }
    }
    protected float Balance
    {
        get { return balance; }
    }
}
```

This class is almost coded correctly. However, the Balance property only has a get, which makes it read-only. Instead, a get and set should be defined so that it is read and write.

Objective:

Create and Use Types

Subobjective:

Enforce encapsulation.

1.1.4 Question 4. Objective : Implement Data Access. Subobjective : Perform I/O operations.

You use Microsoft .NET Framework 4.5 to create an application. This code exists (line numbers are included for reference only):

```
01 public async Task ProcessPdfFiles(List<string> files)
02 {
03     foreach (string file in files)
04     {
05         using (StreamReader reader = new StreamReader(file))
06         {
07             string data = reader.ReadToEnd();
08             ProcessData(data);
09         }
10     }
11     Console.WriteLine("Processing complete");
12 }
```

It takes a long time to read each file. You must ensure that the `WriteLine` method of the `Console` class is not called until all files are processed. However, the `ProcessPdfFiles` method must return immediately to its caller while the files are being read and processed.

You need to modify the code to accomplish your goal.

What should you do?

Choose the correct answer

Remove the `async` modifier from the `ProcessPdfFiles` method.

Modify line 07 as follows:

`yield return reader.ReadToEnd();`

Change the return type of the `ProcessPdfFiles` method from `Task` to `void`.

Modify line 07 as follows:

`String data = await reader.ReadToEndAsync();`

1.1.4.1 Correction

You should modify line 07 as follows:

```
String data = await reader.ReadToEndAsync();
```

The `ReadToEndAsync` method is an asynchronous version of the `ReadToEnd` method of the `StreamReader` class. The `await` keyword allows a method to execute asynchronously while control returns to the calling code. Control immediately returns to the code that calls the `ProcessPdfFiles` method. Because the method returns a `Task` instance, the calling code can use the `Task` instance to determine when the asynchronous method completes.

You should not remove the `async` modifier from the `ProcessPdfFiles` method. This modifier marks a method as an asynchronous method. Used in conjunction with the `await` keyword, you allow the calling code to call the method asynchronously.

You should not change the return type of the `ProcessPdfFiles` method from `Task` to `void`. By returning a `Task` instance, you allow the calling code to determine when the `ProcessPdfFiles` method completes.

You should not modify line 07 by adding the `yield return` statement. You should use the `yield return` statement in methods that return an `IEnumerable` instance. The `yield return` statement cannot be used on methods that return any instance other than `IEnumerable`.

Objective:

Implement Data Access

Subobjective:

Perform I/O operations.

1.1.5 Question 5 Objective : Debug Applications and Implement Security. Subobjective : Manage assemblies.

You use Microsoft .NET Framework 4.5 to create an application. The application references version 1.0.0.0 of an assembly named DataAccess.dll. You obtain version 3.0.0.0 of the assembly. Both assemblies exist in the global assembly cache (GAC).

You need to force the application to use version 3.0.0.0 of the assembly.

What should you do?

Choose the correct answer

- Copy version 3.0.0.0 of the assembly to the directory that contains the application.

Modify the application configuration file as follows:

```
<configuration>
  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <probing privatePath="GAC\3.0.0.0"/>
    </assemblyBinding>
  </runtime>
</configuration>
```

- Copy version 3.0.0.0 of the assembly to the Windows System32 directory.

Modify the application configuration file as follows:

```
<configuration>
  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
        <assemblyIdentity name="DataAccess" culture="neutral"/>
        <bindingRedirect oldVersion="1.0.0.0" newVersion="3.0.0.0"/>
      </dependentAssembly>
    </assemblyBinding>
  </runtime>
</configuration>
```

1.1.5.1 Correction

^ Explanation

You should modify the application configuration file as follows:

```
<configuration>
  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
        <assemblyIdentity name="DataAccess" culture="neutral"/>
        <bindingRedirect oldVersion="1.0.0.0" newVersion="3.0.0.0"/>
      </dependentAssembly>
    </assemblyBinding>
  </runtime>
</configuration>
```

This configuration redirects the application from version 1.0.0.0 of DataAcess.dll to version 3.0.0.0. The assemblyIdentity element specifies the assembly to be redirected. The bindingRedirect element specifies the version to be redirected. The oldVersion attribute specifies the version from which you are redirecting, and the newVersion attribute specifies the version to which you are redirecting.

You should not copy version 3.0.0.0 of the assembly to the directory that contains the application. Because version 1.0.0.0 is installed in the GAC and the application references it, that version is automatically loaded. The runtime always loads assemblies from the GAC if they exist there.

You should not modify the application configuration file as follows:

```
<configuration>
  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <probing privatePath="GAC\3.0.0.0"/>
    </assemblyBinding>
  </runtime>
</configuration>
```

The probing element specifies hints to allow the runtime to find referenced assemblies. Specifying GAC\3.0.0.0 as the value of the privatePath attribute does not force the application to use version 3.0.0.0 from the GAC. It tells the application to look in a subdirectory named GAC\3.0.0.0 to find the reference assembly, but only if it does not exist in the GAC. In this scenario, the assembly does exist in the GAC, so specifying the probing element would not help solve the problem.

You should not copy version 3.0.0.0 of the assembly to the Windows System32 directory. This would not solve your problem. The runtime loads assemblies from the GAC if they exist there. Therefore, you need to configure the application to redirect from version 1.0.0.0 to version 3.0.0.0 of the assembly by using the bindingRedirect element.

Objective:

Debug Applications and Implement Security

Subobjective:

Manage assemblies.

References

MSDN

Microsoft

Copyright Year: 2013

Assembly Versioning

[Click here for more information](#)

1.1.6 Question 6 Objective : Manage Program Flow Subobjective : Create and Implement event and callbacks.

You use Microsoft .NET Framework 4.5 to create an application. The .NET Framework Comparison delegate is defined as follows:

```
public delegate int Comparison<in T>(T x,T y);
```

The method that implements this delegate must return a:

- * negative number if x is less than y
- * positive number if x is greater than y
- * 0 if x is equal to y

The Sort method of the .NET Framework generic List class is defined as follows:

```
public void Sort<T>(Comparison<T> comparison);
```

This class exists in your application:

```
public class Product
{
    public int ID {get; set;}
    public string Name {get; set;}
    public double Price {get; set;}
}
```

You declare this generic List instance:

```
List<Product> products = GetProducts();
```

You must call the Sort method of the products instance to sort the list by the ID property of the Product class.

You need to use a lambda expression to accomplish your goal.

Which lambda expression should you pass to the Sort method?

Choose the correct answer

- (x,y) => x.ID < y.ID ? -1 : x.ID > y.ID ? 1 : 0
- (Product y) => x.ID < y.ID ? -1 : x.ID > y.ID ? 1 : 0
- (Product x) => x.ID < y.ID ? -1 : x.ID > y.ID ? 1 : 0
- () => x.ID < y.ID ? -1 : x.ID > y.ID ? 1 : 0

1.1.6.1 Correction

^ Explanation

You should use this lambda expression:

`(x,y) => x.ID < y.ID ? -1 : x.ID > y.ID ? 1 : 0`

The expression on the left side of the `=>` lambda operator represents the input parameters to the delegate. Just like any method that implements a delegate, you can use any variables that you like, as long as you specify the correct number of parameters. When specifying a single input parameter, you do not need to use parenthesis. However, when specifying more than one input parameter, you must enclose them in parenthesis, separated by commas. The expression on the right side of the `=>` lambda operator represents the method body of the delegate. If your expression contains only a single code statement, then you do not need to enclose the code within curly braces {}, you do not need to end the code statement with a semicolon, and you do not need to use the return keyword to return a value from the method. However, had your expression contained multiple code statements, you would have needed to enclose the statements within curly braces {}, end each statement with a semicolon, and add a final statement with the return keyword to return a value from the method.

You should not use this lambda expression:

`() => x.ID < y.ID ? -1 : x.ID > y.ID ? 1 : 0`

This lambda expression does not specify the required number of input parameters for the expression on the left side of the lambda operator.

You should not use this lambda expression:

`(Product x) => x.ID < y.ID ? -1 : x.ID > y.ID ? 1 : 0`

This lambda expression specifies only a single input parameter. This expression also specifies a type for the parameter; however, specifying a type is not required unless the compiler cannot infer the type from the usage.

You should not use this lambda expression:

`(Product y) => x.ID < y.ID ? -1 : x.ID > y.ID ? 1 : 0`

This lambda expression specifies only a single input parameter. This expression also specifies a type for the parameter; however, specifying a type is not required unless the compiler cannot infer the type from the usage.

Objective:

Manage Program Flow

<http://msdn.microsoft.com/en-us/library/bb397687.aspx> <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/statements-expressions-operators/lambda-expressions>

<http://msdn.microsoft.com/en-us/library/w56d67y8z.aspx> https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1.sort?redirectedfrom=MSDN&view=netframework-4.7.2#System_Collections_Generic_List_1_Sort_System_Comparison_0

1.1.7 Question 7 Objective : Manage Program Flow. Subobjective :Implement program flow.

You are writing a C# application for a convenience store by using Visual Studio 2015. The app must return an enum indicating whether the customer is a minor (under age 18), can buy tobacco (at least age 18), or can buy alcohol (at least age 21) based on the customer's age. You create the following enum:

```
enum AgeType
{
    Unknown,
    Minor,
    Tobacco,
    Alcohol
}
```

You need to write a function that will return the appropriate value.

Which block of code should you write?

Choose the correct answer

```
AgeType GetType(int age)
{
    AgeType type = AgeType.Unknown;
    if (age > 21)
```

```
AgeType GetType(int age)
{
    AgeType type = AgeType.Unknown;
    if (age > 21)
    {
        type = AgeType.Alcohol;
    }
    else if (age < 21)
    {
        type = AgeType.Tobacco;
    }
    else
    {
        type = AgeType.Minor;
    }
    return type;
}
```



AgeType GetType(int age)
{
 AgeType type = AgeType.Unknown;
 if (age < 18)
 {
 type = AgeType.Minor;
 }
 else if (age < 21)
 {
 type = AgeType.Tobacco;
 }
 else if (age > 21)
 {
 type = AgeType.Alcohol;
 }
 return type;
}

AgeType GetType(int age)
{
 AgeType type = AgeType.Unknown;
 if (age < 18)
 {
 type = AgeType.Minor;
 }
 else if (age < 21)
 {
 type = AgeType.Tobacco;
 }
 else
 {
 type = AgeType.Alcohol;
 }
 return type;
}

1.1.7.1 Correction

^ Explanation

You should write the block of code below:

```
AgeType GetType(int age)
{
    AgeType type = AgeType.Unknown;
    if (age < 18)
    {
        type = AgeType.Minor;
    }
    else if (age < 21)
    {
        type = AgeType.Tobacco;
    }
    else
    {
        type = AgeType.Alcohol;
    }
    return type;
}
```

This code uses an if-else block to test each age condition and returns the correct result based on the customer's age. If the customer is under 18, then Minor is returned. Otherwise, the code checks to see if the customer is under 21, which returns Tobacco. Otherwise the code returns Alcohol because at this point the customer must be 21 or older.

You should not write the block of code below:

```
AgeType GetType(int age)
{
    AgeType type = AgeType.Unknown;
    if (age < 18)
    {
        type = AgeType.Minor;
    }
    if (age < 21)
    {
        type = AgeType.Tobacco;
    }
    if(age > 21)
    {
        type = AgeType.Alcohol;
    }
    return type;
}
```

This code uses several if-then blocks to test each age condition. However, because each if-then block will be tested independently, there are situations where the correct result will not be returned. For example, if the customer is 10 years old, the first if-then condition will set the result to Minor. Then the second if-then condition will run. Since the customer is also less than 18, the result will be set to Tobacco. When you have several mutually exclusive conditions, always consider using an if-else structure so that one and only one condition can be evaluated as true.

You should not write the block of code below:

```
AgeType GetType(int age)
{
    AgeType type = AgeType.Unknown;
    if (age < 18)
    {
        type = AgeType.Minor;
    }
    else if (age < 21)
    {
        type = AgeType.Tobacco;
    }
    else if (age > 21)
    {
        type = AgeType.Alcohol;
    }
    return type;
}
```

This code uses an if-else block to test each age condition. It is mostly correct, except there is one error. The second condition tests to see if the age is less than 21, and the final condition tests to see if the age is greater than 21. So, any customer who is exactly 21 will fail all three conditions, resulting in a value of AgeType.Unknown being returned. You should always double-check to make sure that your conditions cover all of the desired results. In this case, the third condition should be age \geq 21.

You should not write the block of code below:

```
AgeType GetType(int age)
{
    AgeType type = AgeType.Unknown;
    if (age  $\geq$  21)
    {
        type = AgeType.Alcohol;
    }
    else if (age < 21)
    {
        type = AgeType.Tobacco;
    }
    else if(age < 18)
    {
        type = AgeType.Minor;
    }
    return type;
}
```

This code uses an if-else block to test each age condition. However, it incorrectly tests the oldest age first, and then moves down through the younger ages. In some situations, this will have undesired results. For example, if the customer is 10, the second condition of Age $<$ 21 will be true, and the AgeType will be set to Tobacco. In fact, the final condition will never evaluate to true. You should always consider the order in which you are evaluating numerical conditions in ranked lists of comparisons.

1.1.8 Question 8 Objective : Create and Use Types. Subobjective : Create types

You are developing a product catalog application by using C# and Microsoft Visual Studio 2015. You are programming entity classes.

You have declared this class:

```
public partial class Product : INotifyPropertyChanged
{
    public double ListPrice
    {
        get { return this._listPrice; }
        set
        {
            if (value != this._listPrice)
            {
                this.OnListPriceChanging(value);
                this.OnPropertyChanging("ListPrice");
                this._listPrice = value;
            }
        }
    }
    public event PropertyChangedEventHandler PropertyChanging;
    protected virtual void OnPropertyChanging(string propertyName)
    {
        var ev = this.PropertyChanging;
        if (ev != null)
            this.PropertyChanging(this,
                new PropertyChangedEventArgs(propertyName));
    }
}
```

If another developer defines the `OnListPriceChanging` method in a separate file, you want to call it. If no one defines this method, you want the code to compile successfully.

Which declaration should you use to complete the `Product` class?

Choose the correct answer

- partial void OnListPriceChanging(double value);
- virtual void OnListPriceChanging(double value);
- protected void OnListPriceChanging(double value);

1.1.8.1 Correction

^ Explanation

You should use this declaration:

```
partial void OnListPriceChanging(double value);
```

The partial keyword on the class declaration causes the compiler to combine the declarations from multiple partial class declarations into a single class if they share the same fully-qualified class name. Within a partial class block, a partial method declaration causes the compiler to allow a partial class block to refer to methods that have been declared with a return type, name, and signature but without the implementation block. If another partial class block contains the implementation, then the compiler will use it. If no other partial class blocks implement the method, the compiler automatically implements an empty, or no-op, method block.

You should not use this declaration:

```
protected void OnListPriceChanging(double value);
```

The protected keyword allows other class declarations to access the member. When you declare a method with the protected keyword, you must implement the method block as part of the declaration. The protected keyword cannot be used in the same declaration as the partial keyword.

You should not use this declaration:

```
new void OnListPriceChanging(double value);
```

The new keyword enables member hiding. You can use this keyword when you inherit from a class. If the base class does not declare a member with the virtual keyword, your derived class can declare a replacement method by using the new keyword. When you declare a method with the new keyword, you must implement the method block as part of the declaration. The protected keyword cannot be used in the same declaration as the partial keyword.

You should not use this declaration:

```
virtual void OnListPriceChanging(double value);
```

The virtual keyword causes the common language runtime (CLR) to dynamically select the most-specific method declaration. You can create a derived class that declares a method with the override keyword to create a more-specific method in your derived class. When you declare a method with the virtual keyword you must implement the method block as part of the declaration. The virtual keyword cannot be used in the same declaration as the partial keyword.

Objective:

Create and Use Types

Subobjective:

Create types.

1.1.9 Question 9 Objective : Create and Use Types Subobjective : Create types.

You are developing an application by using C# and Visual Studio 2015. You have this class definition (Line numbers are provided for reference only.):

```
01 public class Manager  
02  
03 {  
04     private IList<Employee> _employees = new List<Employee>();  
05  
06 }
```

You want to add and retrieve employee objects from the internal collection by using the EmployeeID with index notation. For example, managerInstance[17] would return the Employee instance that has the ID property set to 17.

Which code block(s) should you use?

Choose the correct answer

At Line 02:

: IQueryable

At Line 05:

```
public IEnumerator GetEnumerator()  
{ //implementation omitted }  
public Type ElementType {  
    get { //implementation omitted }  
}  
public Expression Expression {  
    get { //implementation omitted }  
}  
public IQueryProvider Provider {  
    get { //implementation omitted }  
}
```



At Line 02:
: IEnumerable

At Line 05:

```
public IEnumerator GetEnumerator()  
{ //implementation omitted }
```

At Line 05:

```
public Employee index[int employeeID] {  
    get { //implementation omitted }  
}
```

At Line 05:

```
public Employee this[int employeeID] {  
    get { //implementation omitted }  
    set { //implementation omitted }  
}
```

1.1.9.1 Correction

^ Explanation

You should use this code block at Line 05 to implement index notation in a custom class:

```
public Employee this[int employeeID] {  
    get { //implementation omitted }  
    set { //implementation omitted }  
}
```

This syntax appears very similar to a standard property. An indexer includes a get block that accepts arguments of the types and names declared between the square brackets ([and]) and returns a value of the type declared before the this keyword. The set block accepts arguments of the types and names declared between the square brackets plus a special argument referenced by the value keyword which is of the type declared before the this keyword.

You should not use this code block at Line 05 that declares a read-only indexer:

```
public Employee index[int employeeID] {  
    get { //implementation omitted }  
}
```

Like property declarations, if an indexer does not contain a set block declaration, the indexer cannot be assigned to. If an indexer does not contain a get block declaration, the indexer cannot be used as an expression.

You should not use these code blocks at Line 02 and 05 that implement the System.IEnumerable interface:

At line 02:

```
: IEnumerable
```

At line 05:

```
public IEnumerator GetEnumerator()  
{ //implementation omitted }
```

The IEnumerable interface is used by the C# compiler to permit a class to be used after the in keyword when using a foreach loop. Many of the .NET Framework Class Library (FCL) classes that implement IEnumerable also implement one or more indexer properties. However, the IEnumerable interface and index notation are not technically related.

You should not use these code blocks at Line 02 and 05 that implement the System.Linq.IQueryable interface:

At line 02:

: IQueryable

At line 05:

```
public IEnumerator GetEnumerator()
{ //implementation omitted }
public Type ElementType {
    get { //implementation omitted }
}
public Expression Expression {
    get { //implementation omitted }
}
public IQueryProvider Provider {
    get { //implementation omitted }
}
```

The IQueryable interface is part of Language-Integrated Query (LINQ). This interface permits a query provider to process a LINQ query prior to execution. For instance, LINQ to SQL and LINQ to Entities are providers that can translate from a LINQ query to a Structured Query Language (SQL) query prior to execution.

Objective:

Create and Use Types

1.1.10 Question 10 Objective : Create and Use Types Subobjective : Find, execute, and create types at runtime by using reflexion.

You are developing an application by using C# and Visual Studio 2015. You are using .NET Reflection to dynamically load an Assembly and instantiate types discovered within it.

You have this code: (Line numbers are for reference purposes only.)

```
01 var assemblyPath = "C:\someAssembly.dll";
02
03 foreach (Type publicType in assembly.ExportedTypes)
04     InstantiateTypeByUsingReflection(publicType); // Defined elsewhere
```

You need to complete the implementation by inserting a statement at line 02.

Which statement correctly completes the implementation?

Choose the correct answer

- var assembly = Assembly.LoadFrom(assemblyPath);
- var assembly = Assembly.ReflectionOnlyLoadFrom(assemblyPath);
- var assembly = Assembly.Load(assemblyPath);
- var assembly = new Assembly {Location = assemblyPath};

1.1.10.1 Correction :

Choose the correct answer



```
var assembly = Assembly.LoadFrom(assemblyPath);
```



```
var assembly = Assembly.ReflectionOnlyLoadFrom(assemblyPath);
```



```
var assembly = Assembly.Load(assemblyPath);
```



```
var assembly = new Assembly {Location = assemblyPath};
```

Explanation

You should use this statement at line 02 to complete the implementation:

```
var assembly = Assembly.LoadFrom(assemblyPath);
```

The static `Assembly.LoadFrom` method accepts a file path argument and loads the identified assembly into the current AppDomain. You may then use .NET Reflection to discover and instantiate types, and execute methods.

You should not use this statement at line 02 to complete the implementation:

```
var assembly = Assembly.ReflectionOnlyLoadFrom(assemblyPath);
```

The static `Assembly.ReflectionOnlyLoadFrom` method accepts a file path argument and loads the identified assembly into a reflection-only context. This context allows you to use .NET Reflection to discover metadata including type names, member names, and annotations. The reflection-only context is unsuitable for instantiating discovered types because it prevents the execution of any code from the assembly.

You should not use this statement at line 02 to complete the implementation:

```
var assembly = new Assembly {Location = assemblyPath};
```

The `Assembly` class constructor has internal access and cannot be called from your code. You should use one of the static methods that can discover and load assemblies.

You should not use this statement at line 02 to complete the implementation:

```
var assembly = Assembly.Load(assemblyPath);
```

The static `Assembly.Load` method accepts a string that identifies a fully qualified assembly name. The string "mscorlib, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" identifies the standard `mscorlib.dll` assembly. You should not pass a file path to this method.

Objective:

1.1.11 Question 11 Objective : Implement Data Access Subobjective : Query and manipulate data and objects by using LINQ

You use Microsoft .NET Framework 4.5 to create an application. This class exists:

```
public class Group
{
    public int Id {get; set;}
    public string Name {get; set;}

    public static IQueryable<Group> GetGroups()
    {
        // Implementation Omitted
        // This code queries a Microsoft SQL Server database for a list of groups
    }
}
```

This code exists:

```
var query = from g in Group.GetGroups() select g.Name;
```

When you run the application, the previous code statement executes, but no queries are sent to the database server. You ensure that data exists and that the GetGroups method is implemented correctly.

You need to force execution of the database query.

What should you do?

Choose the correct answer

- Call the `ToList` method of the query variable.
- Replace the code statement with this:
`var query = Group.GetGroups().Select(g => g.Name);`
- Call the `Select` method of the query variable.
- Replace the code statement with this:
`var query = Group.GetGroups().Select(g => g);`

1.1.11.1 Correction

CHOOSE THE CORRECT ANSWER



Call the `ToList` method of the query variable.



Replace the code statement with this:

```
var query = Group.GetGroups().Select(g => g.Name);
```



Call the `Select` method of the query variable.



Replace the code statement with this:

```
var query = Group.GetGroups().Select(g => g);
```

^ Explanation

You should call the `ToList` method of the query variable. The query variable's type is `IQueryable<Group>`. A LINQ query is converted to a command tree at compile time. At runtime, the command tree is converted to the appropriate data source query and then executed at the data source, but only when the query is requested to be iterated. One way to force execution of a query is to call the `ToList` method. This forces the LINQ engine to query the underlying data source and return the results as a list. Because of this delayed execution, you can combine multiple LINQ queries together, such as with joins, and the LINQ engine converts the queries to efficiently execute at the data source.

You should not replace the code statement with this:

```
var query = Group.GetGroups().Select(g => g);
```

This method calls the `Select` method to return a list of `Group` instances. However, the `Select` method does not force immediate execution.

You should not replace the code statement with this:

```
var query = Group.GetGroups().Select(g => g.Name);
```

This method calls the `Select` method to return a list of group names. However, the `Select` method does not force immediate execution.

You should not call the `Select` method of the query variable. The `Select` method does not force immediate execution.

Objective:

Implement Data Access

Subobjective:

Query and manipulate data and objects by using LINQ.

1.1.12 Question 12 Objective : Create and Use Types. Subobjective : Consume types

You are developing an application by using C# and Microsoft Visual Studio 2015. Your application needs to log messages generated by several libraries. You do not have access to the source for these libraries.

Each message class declares a method called Log that takes zero arguments.

You need to write code that logs the message classes from each of three libraries.

Which code block should you use? (Each correct answer presents a complete solution. Choose two.)

Choose the correct answers

```
 public interface ILoggable {  
    void Log();  
}  
public class Logger {  
    public void LogMessage(ILoggable message)  
    {/* omitted */}  
}
```

```
 public class Logger {  
    public void LogMessage(var message)  
    { /* omitted */}  
}
```

```
 public class Logger {  
    public void LogMessage(Library1.MessageClass message)  
    { /* omitted */}  
    public void LogMessage(Library2.MessageClass message)  
    { /* omitted */}  
    public void LogMessage(Library3.MessageClass message)  
    { /* omitted */}  
}
```

```
 public class Logger {  
    public void LogMessage(dynamic message)  
    { /* omitted */}  
}
```

1.1.12.1 Correction

```
public class Logger {  
    public void LogMessage(Library1.MessageClass message)  
    { /* omitted */ }  
    public void LogMessage(Library2.MessageClass message)  
    { /* omitted */ }  
    public void LogMessage(Library3.MessageClass message)  
    { /* omitted */ }  
}
```

```
public class Logger {  
    public void LogMessage(dynamic message)  
    { /* omitted */ }  
}
```

Explanation

You should use this code block:

```
public class Logger {  
    public void LogMessage(dynamic message)  
    { /* omitted */ }  
}
```

The dynamic keyword can be used in declarations where you know that specific properties and methods are present on objects that cannot be verified by the compiler. In this example, all of the Message classes support the same method signature. If you had access to the source code, you would extract an interface to represent the common method. In this example, you do not have control over the source. The dynamic keyword allows you to write a single method that can call the common method on these unrelated classes.

Or, you could use this code block:

```
public class Logger {  
    public void LogMessage(Library1.MessageClass message)  
    { /* omitted */ }  
    public void LogMessage(Library2.MessageClass message)  
    { /* omitted */ }  
    public void LogMessage(Library3.MessageClass message)  
    { /* omitted */ }  
}
```

You can use overloaded methods to perform similar operations by processing different arguments. The C# compiler allows you to create several methods that share the same name if the count or respective data type of the arguments are unique for each method. In this example, each library declares a unique `MessageClass` type that does not share a common type, except `object`, with the `MessageClass` types from the other libraries. The compiler distinguishes between the methods based on the data type of the passed argument.

You should not use this code block:

```
public interface ILoggable {  
    void Log();  
}  
public class Logger {  
    public void LogMessage(ILoggable message)  
    {/* omitted */}  
}
```

You should not create an interface. An interface is a data type that describes a class or struct that contains specified methods. Multiple interfaces can be declared in a single data type. Declaring interfaces and then implementing interfaces in multiple unrelated classes is a good way to create loosely-coupled code. In this example, you do not have access to the source code for the class libraries. Therefore, you are unable to implement the `ILoggable` interface in the different `MessageClass` types. The C# compiler does not infer interface types from classes that match the signature but do not implement the interface.

You should not use this code block:

```
public class Logger {  
    public void LogMessage(var message)  
    { /* omitted */}  
}
```

This code incorrectly uses the `var` keyword to declare a parameter. The `var` keyword is used to instruct the compiler to infer the data type of a local variable. The compiler requires `var` declarations to include an initialization expression, from which the data type is inferred. The `var` keyword is not valid as part of a parameter declaration.

Objective:

Create and Use Types

Subobjective:

Consume types.

1.1.13 Question 13 Objective : Implement Data Access Subobjective : Store data and retrieve data from collections.

You use Microsoft .NET Framework 4.5 to create an application. You must create a dictionary with two entries. The first entry should use the number 1 as a key and the string Sales as the value. The second entry should use the number 2 as a key and the string Marketing as the value.

You need to initialize the dictionary.

Which code segment should you use?

Choose the correct answer

var dictionary = new
{
 Entry1=new {Key=1, Value="Sales"},
 Entry2=new {Key=2, Value="Marketing"}
};

var dictionary = new Dictionary()
{
 {1, "Sales"},
 {2, "Marketing"}
};

var dictionary = new[,]
{
 {"1", "Sales"},
 {"2", "Marketing"}
};

var dictionary = new[]
{
 new {Key=1, Value="Sales"},
 new {Key=2, Value="Marketing"}
};

1.1.13.1 Correction

^ Explanation

You should use this code segment:

```
var dictionary = new Dictionary<int, string>()
{
    {1, "Sales"},
    {2, "Marketing"}
};
```

This code creates a dictionary with an integer as its key and a string as its value. To create an entry in the dictionary, you must enclose the key and value separated by a comma within curly braces {}, such as follows: {1, "Sales"}. The entire dictionary must also be enclosed in curly braces.

You should use this code segment:

```
var dictionary = new[]
{
    new {Key=1, Value="Sales"},
    new {Key=2, Value="Marketing"}
};
```

This code creates a one-dimensional array with two entries. Each entry is an anonymous type instance with two properties named Key and Value. This code initializes an array instead of a dictionary.

Figure 1 The second is False

You should not use this code segment:

```
var dictionary = new[,]
{
    {"1", "Sales"},
    {"2", "Marketing"}
};
```

This code creates a two-dimensional array with two entries in each dimension. This code initializes an array instead of a dictionary.

You should not use this code segment:

```
var dictionary = new
{
    Entry1=new {Key=1, Value="Sales"},
    Entry2=new {Key=2, Value="Marketing"}
};
```

This code creates an anonymous type instance that has two properties named Entry1 and Entry2. Each property is itself an anonymous type with two properties named Key and Value. This code initializes an anonymous type instead of a dictionary.

^ Explanation

You should use this code segment:

```
var query = from p in ProductList.GetProducts() select p.Name;
```

This code segment uses a LINQ query expression syntax to project the Name property from a collection of Product instances. The resulting query would return an `IEnumerable<string>` instance of product names.

You should not use this code segment:

```
var query = from p in ProductList.GetProducts() let Name = p select Name;
```

The `let` clause of a LINQ query expression allows you to store the result of an expression to be used in a later clause. In this scenario, `let Name = p` stores each Product instance in a temporary variable called `Name`. The `select` statement returns the values of this variable as a list. Therefore, this query returns a collection of Product instances rather than an `IEnumerable<string>` instance that represents product names.

You should not use this code segment:

```
var query = ProductList.GetProducts().Select( p => new {Name = p});
```

The `Select` method allows you to return data from a data source. This code uses a lambda expression to specify the data that should be returned. It specifies an anonymous type with a single property named `Name` that represents a Product instance. Therefore, this query returns a collection of anonymous type instances rather than an `IEnumerable<string>` instance that represents product names.

You should not use this code segment:

```
var query = ProductList.GetProducts().SelectMany(p => from Name in new[]{p} select Name);
```

The `SelectMany` method flattens a list of sequences into a single sequence. This code returns a collection of Product instances.

Objective:

Implement Data Access

Subobjective:

Query and manipulate data and objects by using LINQ.

1.1.14 Question 14 Objective : Implement Data Access Subobjective : Query and manipulate data and objects by using LINQ

You use Microsoft .NET Framework 4.5 to create an application. These classes exist:

```
public class Product
{
    public string Name {get; set;}
    public string ID {get; set;}
    public double Price {get; set;}
}

public class ProductList : List<Product>
{
    public static ProductList GetProducts()
    {
        // Implementation Omitted
    }
}
```

You need to create a query that returns an `IEnumerable<string>` instance of product names.

Which code segment should you use?

Choose the correct answer

- var query = ProductList.GetProducts().Select(p => new {Name = p});
- var query = from p in ProductList.GetProducts() select p.Name;
- var query = from p in ProductList.GetProducts() let Name = p select Name;
- var query = ProductList.GetProducts().SelectMany(p => from Name in new[]{p} select Name);

1.1.14.1 Correction

1.1.15 Question 15 Objective : Debug Applications and Implement Security Subobjective : Validate application Input

You use Microsoft .NET Framework 4.5 to create an application for your company. The application must communicate securely with the Department of Motor Vehicles (DMV). The DMV must be sure that data sent from your application was not modified while in transit.

You need to secure the data to be sent to the DMV.

What should you do?

Choose the correct answer

- Generate a hash of the data. Encrypt the hash with the public key of the DMV. Send the data and the encrypted hash to the DMV.
- Generate a hash of the data. Encrypt the hash with the private key of your company. Send the data and the encrypted hash to the DMV.
- Encrypt the data with the public key of your company. Hash the encrypted data. Send the encrypted data and the hash to the DMV.
- Encrypt the data with the private key of the DMV. Hash the encrypted data. Send the encrypted data and the hash to the DMV.

1.1.15.1 Correction

 the encrypted hash to the DMV.

 Generate a hash of the data. Encrypt the hash with the private key of your company. Send the data and the encrypted hash to the DMV.

 Encrypt the data with the public key of your company. Hash the encrypted data. Send the encrypted data and the hash to the DMV.

 Encrypt the data with the private key of the DMV. Hash the encrypted data. Send the encrypted data and the hash to the DMV.

Explanation

You should hash the data, encrypt the hash with your company's private key, and send the data and the encrypted hash to the DMV. To ensure the integrity of data that is transmitted, you should sign the data. This consists of encrypting a hash of data with a private key. Hashing is a one-way algorithm that generates a unique hash for the given data. If you encrypt the hash with a private key, only a corresponding public key can decrypt the encrypted hash. The DMV can decrypt the encrypted hash with the public key. The DMV can then re-hash the original data that is sent to it, and compare its own hash to the decrypted hash. If the two values match, then the DMV can be sure that the data was not modified in transit, because only your company should have access to the private key that encrypted the hash.

You should not generate a hash of the data, encrypt the hash with the DMV's public key, and send the data and the encrypted hash to the component. Because public keys are public, any rogue code can modify the data while it is in transit, generate a new hash of the modified data, and then encrypt it with the publicly available public key. So even when the DMV decrypts the encrypted hash with its private key, the DMV cannot be sure as to who or what originally encrypted the hash.

You should not encrypt the data with the DMV's private key, hash the encrypted data, and send the encrypted data and the hash to the DMV. Only the DMV should have access to its private key.

You should not encrypt the data with the DMV's public key, hash the encrypted data, and send the encrypted data and the hash to the DMV. Because public keys are public, any rogue code can intercept the message in transit, generate new data, encrypt it with the DMV's public key, and then hash the encrypted data.

Objective:

Debug Applications and Implement Security

Subobjective:

Validate application input.

1.1.16 Question 16 Objective : Create and Use Type Subobjective : Enforce encapsulation.

You are developing a media transcoding library by using C# and Microsoft Visual Studio 2015. You want to declare a class to encapsulate a single work item.

Your library requires that the WorkItem class contains properties for MediaType and JobID. These properties should be initialized in the constructor. After initialization, code outside of the library should be unable to change the values of these properties. This version of the library does not require custom validation.

You need to begin the WorkItem class declaration and keep the code as simple as possible.

Which code block satisfies these requirements?

Choose the correct answer

public class WorkItem
{
 internal int JobID { public get; set; }
 internal string MediaType { public get; set; }
}

internal class WorkItem
{
 public int JobID { get; set; }
 public string MediaType { get; set; }
}

public class WorkItem
{
 public int JobID { get; internal set; }
 public string MediaType { get; internal set; }
}

public class WorkItem
{
 public int JobID { internal set; }
 public string MediaType { internal set; }
}

internal class WorkItem
{
 public int JobID { get; set; }
 public string MediaType { get; set; }
}

public class WorkItem
{
 public int JobID { get; internal set; }
 public string MediaType { get; internal set; }
}

public class WorkItem
{
 internal int JobID { public get; set; }
 internal string MediaType { public get; set; }
}

1.1.16.1 Correction

^ Explanation

You should use this code:

```
public class WorkItem
{
    public int JobID { get; internal set; }
    public string MediaType { get; internal set; }
}
```

This code uses auto-implemented properties declared with the public access modifier. This specifies that any code that can access the WorkItem class may also directly reference the properties. However, the set keywords are declared with the internal access modifier. This indicates that only code in the same assembly as the WorkItem class may directly reference the set block.

You should not use this code:

```
public class WorkItem
{
    internal int JobID { public get; set; }
    internal string MediaType { public get; set; }
}
```

This code declares the auto-implemented properties with the internal access modifier. The get keywords are declared with the less-restrictive public access modifier. You may only use an alternative access modifier on the get or set keyword if it is more restrictive than the access modifier on the whole property.

You should not use this code:

```
internal class WorkItem
{
    public int JobID { get; set; }
    public string MediaType { get; set; }
}
```

This code declares the entire WorkItem class with the internal keyword. This hides the WorkItem class from code outside of its containing assembly. The requirements indicate that only the set blocks of the properties should be this restrictive.

You should not use this code:

```
public class WorkItem
{
    public int JobID { internal set; }
    public string MediaType { internal set; }
}
```

This code declares the set keyword with the internal access modifier and omits the get keyword. You may only declare a more-restrictive access modifier on the get or set block of an auto-implemented property if you declare both the get and set keywords.

Objective:

Create and Use Types

Subobjective:

Enforce encapsulation.

1.1.17 Question 17 Objective : Implement Data Access Subobjective : Perform I/O operations.

You use Microsoft .NET Framework 4.5 to create an application. The application must accept incoming TCP requests over port 500.

You need to write code that retrieves a single request as a string.

Which code segment should you use?

Choose the correct answer

TcpListener listener = new TcpListener(IPAddress.Any, 500);
listener.Start();
Socket socket = listener.AcceptSocket();
string message = socket.ToString();

TcpListener listener = new TcpListener(IPAddress.Any, 500);
listener.Start();
TcpClient client = listener.AcceptTcpClient();
NetworkStream stream = client.GetStream();
using (StreamReader reader = new StreamReader(stream))
{
 string message = reader.ReadToEnd();
}

TcpClient client = new TcpClient(new IPEndPoint(IPAddress.Any, 500));
client.Client.Accept();
NetworkStream stream = client.GetStream();
using (StreamReader reader = new StreamReader(stream))
{
 string message = reader.ReadToEnd();
}

TcpClient client = new TcpClient(new IPEndPoint(IPAddress.Any, 500));
client.Client.Listen(1);
NetworkStream stream = client.GetStream();
using (StreamReader reader = new StreamReader(stream))
{
 string message = reader.ReadToEnd();
}

1.1.17.1 Correction :

Explanation

You should use this code segment:

```
TcpListener listener = new TcpListener(IPAddress.Any, 500);
listener.Start();
TcpClient client = listener.AcceptTcpClient();
NetworkStream stream = client.GetStream();
using (StreamReader reader = new StreamReader(stream))
{
    string message = reader.ReadToEnd();
}
```

The `TcpListener` class allows you to listen for incoming TCP requests. The `Start` method starts the listening process. The `AcceptTcpClient` method accepts a single connection. The method blocks the current thread until a TCP client connection is established. It returns a `TcpClient` instance that represents the connected network client. The `GetStream` method of the `TcpClient` class returns a `NetworkStream` instance that represents the underlying stream that contains the messages sent by the client. You can read data from the `NetworkStream` instance by calling the `ReadToEnd` method of the `StreamReader` class.

You should not use this code segment:

```
TcpClient client = new TcpClient(new IPEndPoint(IPAddress.Any, 500));
client.Client.Accept();
NetworkStream stream = client.GetStream();
using (StreamReader reader = new StreamReader(stream))
{
    string message = reader.ReadToEnd();
}
```

This code creates a `TcpClient` instance. You should create instances of `TcpClient` when you need to send data over the network, not when you need to receive data from the network. In this scenario, you need to receive data from the network.

You should not use this code segment:

```
TcpListener listener = new TcpListener(IPAddress.Any, 500);
listener.Start();
Socket socket = listener.AcceptSocket();
string message = socket.ToString();
```

This code calls the `AcceptSocket` instance of the `TcpListener` class to return a `Socket` instance that represents a network socket. Although you can use the `Socket` class to receive network data, you should not call its `ToString` method to do so. This method simply returns the name of the `Socket` class as a string. Instead, you would need to call the `Receive` method of the `Socket` class.

You should not use this code segment:

```
TcpClient client = new TcpClient(new IPEndPoint(IPAddress.Any, 500));
client.Client.Listen(1);
NetworkStream stream = client.GetStream();
using (StreamReader reader = new StreamReader(stream))
{
    string message = reader.ReadToEnd();
}
```

This code creates a `TcpClient` instance. You should create instances of `TcpClient` when you need to send data over the network, not when you need to receive data from the network. In this scenario, you need to receive data from the network.

1.1.18 Question 18 Objective : Create and Use Type Subobjective : Consume types

You are developing a class library by using C# and Visual Studio 2015. Developers who write Component Object Model (COM)-based applications need to use one of the classes in your library.

Your library includes this code:

```
[ComVisible(true)]  
public class PolicyQuoter  
{  
    public decimal GetAutoQuote(int customerId) { // implementation omitted }  
    public decimal GetLifeQuote(int customerId) { // implementation omitted }  
    public decimal GetADDQuote(int customerId) { // implementation omitted }  
}
```

You need to export only the PolicyQuoter class into a COM Type Library.

What should you do? (Choose all that apply.)

Choose the correct answers

- Implement IUnknown in the PolicyQuoter class
- Implement IDispatch in the PolicyQuoter class
- Add this attribute to AssemblyInfo.cs:
[assembly: ComVisible(false)]
- Add this attribute to AssemblyInfo.cs:
[assembly: ComVisible(true)]
- Add this attribute to AssemblyInfo.cs
[assembly: Guid("C358E7BE-7516-473C-BB4C-CC8953DAF437")]

Add this attribute to each method declaration:

`[Guid("C358E7BE-7516-473C-BB4C-CC8953DAF437")]`

Run this command after building the assembly:

`regasm MyAssembly.dll`

Run this command after building the assembly:

`tlbexp MyAssembly.dll`

1.1.18.1 Correction

Choose the correct answers



Implement `IUnknown` in the `PolicyQuoter` class



Implement `IDispatch` in the `PolicyQuoter` class



Add this attribute to `AssemblyInfo.cs`:

`[assembly: ComVisible(false)]`



Add this attribute to `AssemblyInfo.cs`:

`[assembly: ComVisible(true)]`



Add this attribute to `AssemblyInfo.cs`

`[assembly: Guid("C358E7BE-7516-473C-BB4C-CC8953DAF437")]`



Add this attribute to each method declaration:

`[Guid("C358E7BE-7516-473C-BB4C-CC8953DAF437")]`



Run this command after building the assembly:

`regasm MyAssembly.dll`



Run this command after building the assembly:

`tlbexp MyAssembly.dll`

Explanation

You should add this attribute to `AssemblyInfo.cs`:

`[assembly: ComVisible(false)]`

When the `ComVisible` attribute is applied to an assembly, the boolean argument is treated as the default for all types in the library. In this example, `PolicyQuoter` is the only class that should be exported to a COM Type Library. By setting `ComVisible` to false for the whole assembly, only types that are annotated with `[ComVisible(true)]` are exported.

The AssemblyInfo.cs file is the conventional file that contains all attributes that apply to the whole assembly.

You should add this attribute to AssemblyInfo.cs:

```
[assembly: Guid("C358E7BE-7516-473C-BB4C-CC8953DAF437")]
```

To expose types from a .NET assembly as COM interfaces requires that the assembly is uniquely identified with a Globally Unique ID (GUID). The Guid attribute should be applied to the assembly, and optionally to any type that will be visible to COM. If you do not explicitly annotate COM-visible types with the Guid attribute, the export process generates a Guid for you.

You should run this command after building the assembly:

```
tlbexp MyAssembly.dll
```

The Type Library Export utility (tlbexp.exe) is installed with Visual Studio 2010 or as part of the Windows Platform SDK. This utility will create a COM Type Library that exposes managed code to COM. In this example, it will create a file named MyAssembly.tlb that can be registered with the Microsoft Windows operating system, and then called by a COM application.

You should not add this attribute to AssemblyInfo.cs:

```
[assembly: ComVisible(true)]
```

This attribute instructs tlbexp.exe to export all public classes in the managed assembly. In this example, you only want to export the PolicyQuoter class.

You should not add this attribute to each method declaration:

```
[Guid("C358E7BE-7516-473C-BB4C-CC8953DAF437")]
```

The Guid attribute may only be applied to assemblies and types. The Guid attribute must not be applied to individual methods, or the compiler will generate errors.

You should not run this command after building the assembly:

```
regasm MyAssembly.dll
```

The Assembly Registration Tool (regasm.exe) is used to register managed assemblies with the local operating system as a COM Type Library. In this example, the type library is used by other developers. Registering a type library on your local operating system will not be useful to other developers. The regasm utility does provide a /tlb flag that instructs regasm to first create a type library before registering the assembly.

You should not implement IDispatch in the PolicyQuoter class. This interface allows late-binding and dynamic languages to use COM interfaces. The tlbexp utility ensures that this interface is implemented in the appropriate COM coclasses in the type library.

You should not implement `IUnknown` in the `PolicyQuoter` class. This interface supports the point-counting memory management technique used by COM. The `tlbexp` utility ensures this interface is implemented in the appropriate COM coclasses in the type library.

Objective:

Create and Use Types

Subobjective:

Consume types.

1.1.19 Question 19 Objective : Create and Use Types Subobjective : Consume types

You are developing an application by using C# and Visual Studio 2015. You have these type declarations (Line numbers are included for reference purposes only.):

```
01 public struct Point2d
02 {
03     public double X { get; set; }
04     public double Y { get; set; }
05     public Point2d(double x, double y) : this()
06     // remaining logic omitted
07 }

08 public struct Point3d
09 {
10     public static explicit operator Point2d(Point3d value)
11     { return new Point2d(value.X, value.Y); }
12     public static implicit operator Point3d(Point2d value)
13     { return new Point3d(value.X, value.Y, 0d); }
14     public double X { get; set; }
15     public double Y { get; set; }
16     public double Z { get; set; }
17     public Point3d(double x, double y, double z) : this()
18     // remaining logic omitted
19 }
```

Which of these are valid statements? (Choose all that apply.)

Choose the correct answers

- Point2d p = new Point3d(5d, 3d, 1d);
- Point3d p = new Point2d(3d, 2d);
- Point2d p = new Point3d(1d, 1d, 1d) as Point2d;
- Point3d p = (Point3d) new Point2d(3d, 5d);
- Point3d p = new Point2d(11d, 13d) as Point3d;
- Point2d p = (Point2d) new Point3d(1d, 2d, 3d);

1.1.19.1 Correction

Choose the correct answers

Point2d p = new Point3d(5d, 3d, 1d);

Point3d p = new Point2d(3d, 2d);

Point2d p = new Point3d(1d, 1d, 1d) as Point2d;

Point3d p = (Point3d) new Point2d(3d, 5d);

Point3d p = new Point2d(11d, 13d) as Point3d;

Point2d p = (Point2d) new Point3d(1d, 2d, 3d);

Explanation

These statements are valid statements:

```
Point2d p = (Point2d) new Point3d(1d, 2d, 3d);
Point3d p = new Point2d(3d, 2d);
Point3d p = (Point3d) new Point2d(3d, 5d);
```

To define a custom conversion, your type must declare the conversion method using a specialized syntax:

```
public static [implicit or explicit] operator [return type]([expression type] <identifier>)
```

The sample code given includes one implicit conversion and one explicit conversion.

You can use this code because of the explicit operator that performs a narrowing operation from the more complex Point3d to the simpler Point2d type:

```
Point2d p = (Point2d) new Point3d(1d, 2d, 3d);
```

When you create a custom type, the new type can only be automatically converted to other types through reference conversion (such as converting to a base type), boxing and unboxing conversions (such as converting from a struct to System.Object and back again), wrapping conversions (such as creating a Nullable<T> from a type T), and null type conversions (literally returning a null reference). To extend this behavior, a type may define custom conversions.

Custom conversions can be defined in one of two forms. An implicit conversion allows the compiler to assign an expression of one type to a variable of another type. This is similar to what is commonly termed a widening conversion, such as assigning a 32-bit signed integer to a 64-bit signed integer. An explicit conversion requires the developer to explicitly add a cast operator to the expression before assigning the value to a variable. This is similar to what is commonly termed a narrowing conversion, such as assigning a long to an int. In the .NET framework, the system only defines widening conversions implicitly, others are explicit only.

You can use this code because of the implicit operator that performs a widening operation from the Point2d type to the Point3d type:

```
Point3d p = new Point2d(3d, 2d);
```

You can use this code because of the implicit operator that the compiler allows you to use with an explicit cast:

```
Point3d p = (Point3d) new Point2d(3d, 5d);
```

You cannot use this code because there is not an implicit operator for the narrowing operation from the Point3d type to the Point2d type.

```
Point2d p = new Point3d(5d, 3d, 1d);
```

The compiler translates the as keyword into code that checks the data type of the left-hand expression before performing the conversion. This code is functionally equivalent to the following:

```
(p1 is Point2d ? (Point2d)p1 : null)
```

Objective:

Create and Use Types

Subobjective:

Consume types.

1.1.20 Question 20 Objective : Implement Data Access Subobjective : Serialize and deserialize data

You use Microsoft .NET Framework 4.5 to create an application. This class exists:

```
public class Payment
{
    public double Amount;
    public string CardNumber;
    public string Payee;
}
```

You plan to use the BinaryFormatter class to serialize instances of Payment. You must ensure that the CardNumber field is not serialized.

You need to modify the Payment class to ensure that it gets serialized correctly.

What should you do?

Choose the correct answer

- Apply the SoapType attribute to the Payment class. Apply the SoapIgnore attribute to the CardNumber field.
- Apply theDataContract attribute to the Payment class. Apply theDataMember attribute to only the Amount and Payee fields.
- Apply the Serializable attribute to the Payment class. Apply the NonSerialized attribute to the CardNumber field.
- Apply the XmlRoot attribute to the Payment class. Apply the XmlIgnore attribute to the CardNumber field.

1.1.20.1 Correction

- Apply the SoapType attribute to the Payment class. Apply the SoapIgnore attribute to the CardNumber field.
- Apply the DataContract attribute to the Payment class. Apply theDataMember attribute to only the Amount and Payee fields.
- Apply the Serializable attribute to the Payment class. Apply the NonSerialized attribute to the CardNumber field.
- Apply the XmlRoot attribute to the Payment class. Apply the XmlIgnore attribute to the CardNumber field.

Explanation

You should apply the Serializable attribute to the Payment class and the NonSerialized attribute to the CardNumber field. The BinaryFormatter serializes classes that are marked with the Serializable attribute or those that implement ISerializable. The NonSerialized attribute specifies that a field should not be serialized.

You should not apply theDataContract attribute to the Payment class and theDataMember attribute to Amount and Payee fields. TheDataContractSerializer class uses theDataContract andDataMember attributes to serialize an object to XML. In this scenario, you are using the BinaryFormatter class for serialization. Therefore, you must use the Serializable and NonSerialized attributes. The BinaryFormatter class does not recognize theDataContract andDataMember attributes.

You should not apply the XmlRoot attribute to the Payment class and the XmlIgnore attribute to the CardNumber field. The XmlSerializer class uses the XmlRoot attribute to serialize a class as a root XML element. It uses the XmlIgnore attribute to ignore a class member during serialization. In this scenario, you are using the BinaryFormatter class for serialization, and it does not recognize the XmlRoot and XmlIgnore attributes.

You should not apply the SoapType attribute to the Payment class and the SoapIgnore attribute to the CardNumber field. You can use the XmlSerializer class to serialize classes as Simple Object Access Protocol (SOAP) messages. It uses the SoapType and SoapIgnore attributes to help control SOAP serialization. In this scenario, you are using the BinaryFormatter class for serialization, and it does not recognize the SoapType and SoapIgnore attributes.

Objective:

Implement Data Access

Subobjective:

Serialize and deserialize data.

1.1.21 Question 21 Objectif : Debug Application and Implement Security Subobjectif : Perform symmetric and asymmetric encryption

You use Microsoft .NET Framework 4.5 to create an application. You use the Makecert.exe utility to create a self-signed root authority certificate. The certificate file is TestCA.cer, and the private key file is TestCA.pvk.

You need to create a new certificate with the subject name Test that is signed by your self-signed root authority certificate.

Which command should you use?

Choose the correct answer

- Makecert.exe -n "CN=TestCA.pvk" -ic TestCA.cer Test.cer
- Makecert.exe -n "CN=Test" -ic TestCA.cer Test.cer
- Makecert.exe -n "CN=TestCA" -ic TestCA.cer Test.cer
- Makecert.exe -n "CN=Test.pvk" -ic TestCA.cer Test.cer

1.1.21.1 Correction

Choose the correct answer

Makecert.exe -n "CN=TestCA.pvk" -ic TestCA.cer Test.cer

Makecert.exe -n "CN=Test" -ic TestCA.cer Test.cer

Makecert.exe -n "CN=TestCA" -ic TestCA.cer Test.cer

Makecert.exe -n "CN=Test.pvk" -ic TestCA.cer Test.cer

Explanation

You should use this command:

Makecert.exe -n "CN=Test" -ic TestCA.cer Test.cer

The -n switch specifies the subject name of the certificate. You must specify the subject name as CN=subject name. In this case, it is CN=Test. The -ic switch specifies the issuer certificate, which is the certificate authority that is issuing the certificate. In this scenario, the certificate authority's certificate is the self-signed root authority certificate named TestCA.cer. The final parameter to the command specifies the name of the certificate that is created, which is Test.cer.

You should not use this command:

Makecert.exe -n "CN=TestCA" -ic TestCA.cer Test.cer

This command specifies a subject name of TestCA, which is incorrect because the subject name should be Test.

You should not use this command:

Makecert.exe -n "CN=TestCA.pvk" -ic TestCA.cer Test.cer

This command specifies a subject name of TestCA.pvk, which is incorrect because the subject name should be Test.

You should not use this command:

Makecert.exe -n "CN=Test.pvk" -ic TestCA.cer Test.cer

This command specifies a subject name of Test.pvk, which is incorrect because the subject name should be Test.

1.1.22 Question 22 Objectif : Debug Application and Implement Security Subobjectif : Validate application input

You use Microsoft .NET Framework 4.5 to create an application. This method exists:

```
public bool CanAdd(object obj)  
{  
}
```

The CanAdd method must return true if the parameter passed to it is an instance of Vehicle or Person. Otherwise it must return false.

You need to implement the CanAdd method. Your solution must not throw an exception.

Which code segment should you use?

Choose the correct answer

- return obj is Vehicle || obj is Person;
- return typeof(Object) == typeof(Vehicle) && typeof(Object) == typeof(Person);
- return obj.GetType() == typeof(Vehicle) && obj.GetType() == typeof(Person);
- return ((Vehicle)obj != null) || ((Person)obj != null);

1.1.22.1 Correction

Choose the correct answer

- return obj is Vehicle || obj is Person;
- return typeof(Object) == typeof(Vehicle) && typeof(Object) == typeof(Person);
- return obj.GetType() == typeof(Vehicle) && obj.GetType() == typeof(Person);
- return ((Vehicle)obj != null) || ((Person)obj != null);

Explanation

You should use this code segment:

```
return obj is Vehicle || obj is Person;
```

The `is` keyword allows you to determine whether an object is an instance of a specific type. The `||` operator represents a conditional OR. This code segment returns true if the `obj` instance's type is `Vehicle` or if it is `Person`.

You should not use this code segment:

```
return ((Vehicle)obj != null) || ((Person)obj != null);
```

This code attempts to cast the `obj` instance to `Vehicle` or `Person`. If the `obj` instance is not one of the two types, then the common language runtime (CLR) will throw an `InvalidCastException` instance.

You should not use this code segment:

```
return obj.GetType() == typeof(Vehicle) && obj.GetType() == typeof(Person);
```

This code uses the `&&` operator, which represents a conditional AND. It returns true if the `obj` instance's type is `Vehicle` and `Person`. The `GetType` method and the `typeof` operator both return `Type` instances that represent an object's type.

You should not use this code segment:

```
return typeof(Object) == typeof(Vehicle) && typeof(Object) == typeof(Person);
```

This code uses the `&&` operator, which represents a conditional AND. It returns true if the `Type` instance that represents `Object` is the same as the `Type` instances that represent `Vehicle` and `Person`, which always fails. The `typeof` operator returns a `Type` instance that represents an object's type.

Objective:

Debug Applications and Implement Security

1.1.23 Question 23 Objective : Create and Use Types Subobjective : Enforce encapsulation

You are creating a C# application by using Visual Studio 2015. You are creating an application that handles orders.

You must create an interface that meets the following requirements:

- * The order id is read-only.
- * The order amount has read and write access.

Which block of code should you write?

Choose the correct answer

```
interface IOrder
{
    string Id
    {
        get id;
        set id;
    }
    float Balance
    {
        get balance;
    }
}
```



```
interface IOrder
{
    public string Id
    {
        set;
    }
    public float Balance
    {
        get;
        set;
    }
}
```



interface IOrder

```
{
    string Id
    {
        get;
    }
    float Balance
    {
        get;
        set;
    }
}
```

interface IOrder

```
{
    string Id
    {
        set;
        get;
    }
    float Balance
    {
        get;
    }
}
```

1.1.23.1 Correction

```
interface IOrder
{
    string Id
    {
        get;
    }
    float Balance
    {
        get;
        set;
    }
}
```



^ Explanation

You should use the code below:

```
interface IOrder
{
    string Id
    {
        get;
    }
    float Balance
    {
        get;
        set;
    }
}
```

This code properly defines a get for the Id property. Any class that implements this interface will only be allowed to expose read access to the Id property. Balance has both a get and set. Any class that implements this interface will be able to expose read and write access to the Balance property.

You should not use the code below:

```
interface IOrder
{
    string Id
    {
        get id;
        set id;
    }
    float Balance
    {
        get balance;
    }
}
```

This code only defines a get for the Balance property. Any class that implements this interface would only be allowed to expose read access to the Balance property. Instead, Balance should have both a get and set so that any class that implements this interface would be able to expose read and write access to the Balance property.

You should not use the code below:

```
interface IOrder
{
    string Id
    {
        set;
        get;
    }

    float Balance
    {
        get;
    }
}
```

This code defines a get and set for the Id property. This means that any class that implements this interface will be allowed to expose read and write access to the Id property. Instead, Id should only have a get so that it is read-only. Balance has only a get, meaning any class that implements this interface will only be able to expose read access to the Balance property. Instead, it should have both a get and set.

You should not use the code below:

```
interface IOrder
{
    public string Id
    {
        set;
    }

    public float Balance
    {
        get;
        set;
    }
}
```

This code defines a set for the Id property. This means any class that implements this interface will only be allowed to grant write access to the Id property. Instead, Id should only have a get so that it is read-only. Also, you cannot apply an access modifier such as public to an interface member.

Objective:

Create and Use Types

Subobjective:

Enforce encapsulation.

1.1.24 Question 24 Objective : Create and Use Types Subobjective : Enforce encapsulation

You are developing a class library by using C# and Microsoft Visual Studio 2015. You want to implement a hierarchy of Product classes. These classes are used in other applications.

You have declared this Product class: (Line numbers are for reference purposes only.)

```
01 class Product {  
02     Product()  
03     { /* constructor implementation omitted */ }  
04 }
```

The Product class is the base class for the hierarchy. It should not be possible to directly construct new instances from other applications. It should be possible for other developers to inherit from the Product class. Other classes in your library are responsible for creating new instances of Product and Product-derived classes.

Which access modifiers complete the Product class to satisfy these requirements? (Each correct answer presents part of the solution. Choose two.)

Choose the correct answers

- private at line 01
- protected at line 02
- protected internal at line 02
- public at line 01
- internal at line 01
- internal at line 02

1.1.24.1 Correction

Choose the correct answers

private at line 01

protected at line 02

protected internal at line 02

public at line 01

internal at line 01

internal at line 02

^ Explanation

You should use the public access modifier at line 01 and the protected internal access modifier at line 02.

The public access modifier is required on the Product class declaration because the Product class must be usable in external assemblies. Only two access modifier keywords permit access outside of the declaring assembly: public and protected. The protected keyword does not apply to type declarations, such as classes.

The protected internal access modifier is required on the Product constructor. The internal keyword is required to permit other code in the declaring assembly to invoke the constructor. The internal keyword permits access to the declared type or member from any other code declared in the same assembly. The protected keyword is required to permit access to code that inherits from the containing type declaration. Derived classes must have access to at least one constructor in the base class. The C# compiler automatically chains all derived class constructors to the default (parameter-free) constructor.

You should not use the private access modifier at line 01. This access modifier should only be used when a type is declared within the code block of another type. This creates a nested type, which can only be used by code within the containing type block.

You should not use the internal access modifier. This restricts access to the Product class to only other code blocks in the same assembly. The requirements specify that other applications use the Product class.

You should not use only the protected access modifier at line 02 because it restricts access to derived classes only. The requirements specify that other classes in the assembly are responsible for creating Product instances. These classes must have access to the constructor by using the internal keyword.

You should not use only the internal access modifier at line 02 because it restricts access to code in the declaring assembly only. The requirements specify that other applications declare classes that derive from the Product class. These classes must have access to the constructor by using the protected keyword.

Objective:

Create and Use Types

Subobjective:

Enforce encapsulation.

References

1.1.25 Question 25 Objective : Create and Use Types Subobjective : Create types

You are developing a utility class library by using C# and Microsoft Visual Studio 2015. Your application has performance problems when you use the built-in collection classes in System.Collections.Generic. After thorough analysis, you have decided to implement an Intrusive Linked List. Because this data structure is a pattern that will be used in many classes, you have created the following interface:

```
public interface IIntrusiveList<T>
{
    IIntrusiveList<T> Next { get; set; }
    IIntrusiveList<T> Previous { get; set; }
    T Self { get; }
}
```

The implementation for common methods such as Add(), Remove(), and Count() can use the same implementation for any class that implements IIntrusiveList<T>. You have decided to implement these three methods as extensions to the IIntrusiveList<T> type.

Complete the implementation of the IntrusiveListExtensions class by dragging the keywords on the left to the appropriate location on the right. You may use items more than once, and you do not have to use each item.

Drag and drop the answers

```
public [Extension] class IntrusiveListExtensions
{
    public [Extension] void Add<T> ([Extension] IIntrusiveList<T> node,
        T item)...
    public [Extension] void Remove<T>([Extension] IIntrusiveList<T> node)...
    public [Extension] long Count<T> ([Extension] IIntrusiveList<T> node)...
}
```

[Extension] base static T this

1.1.25.1 Correction

Drag and drop the answers

```
public static class IntrusiveListExtensions
{
    public static void Add<T> (this IIIntrusiveList<T> node,
        T item)...
    public static void Remove<T>(this IIIntrusiveList<T> node)...
    public static long Count<T> (this IIIntrusiveList<T> node)...
}
```

[Extension] [base] [static] [T] [this]

Explanation

An extension method is a static method with a more convenient syntax. For instance, you could write a method that extends the string type to determine a file type based on the file extension:

```
public static StringExtensions
{
    public static bool IsImage(string fileName)
    { // implementation omitted }
}
```

You would then invoke this method by using the following method call (assume a variable called `fileName` of type `string`):

```
bool result = StringExtensions.IsImage(fileName);
```

An extension method simplifies the method call to:

```
bool result = fileName.IsImage();
```

To create an extension method, you must create a static method in a static class. The class name is unimportant, as long as it is accessible and in scope (by adding the namespace to the code file with the `using` directive). The data type of the first parameter determines the data type to be extended. The other important detail that turns this method from a standard static method to an extension method is the `this` keyword that prefixes the first parameter:

```
public static StringExtensions
{
    public static bool IsImage(this string fileName)
    { // implementation omitted }
}
```

The compiler automatically annotates the method, the class, and the containing assembly with the `ExtensionAttribute`. This allows the compiler to assign the value on the left of the dot (.) operator to the first parameter of the method.

Because the compiler does not require more elaborate design, such as the use of generics or inheritance, it is not necessary to use specialized keywords such as `base` or generic type parameters like `T`.

1.1.26 Question 26 Objective : Create and Use Types Subobjective : Enforce encapsulation

You are developing a library by using C# and Microsoft Visual Studio 2015. Your library handles customer data.

Your library requires a Customer class. This class must include a property called Name. The value of this property should not be stored when the value is either null or an empty string, except when it is initialized. Applications that use the Customer class should not be able to take a reference to the variable, and they should not be able to bypass the validation logic.

Complete the Name property declaration by dragging the code fragments from the list on the left to the appropriate location on the right. You may use items more than once, and you do not have to use each item.

Drag and drop the answers

```
public class Customer
{
    private string  = "";
    public string 
    {
        get
        {
            return this. ;
        }
        set
        {
            if (string.IsNullOrEmpty( ))
                throw new ArgumentException();
            this. = ;
        }
    }
}
```

implementation omitted

1.1.26.1 Correction

Drag and drop the answers

```
public class Customer
{
    private string _name = "";
    public string Name
    {
        get
        {
            return this._name;
        }
        set
        {
            if (string.IsNullOrEmpty(value))
                throw new ArgumentException();
            this._name = value;
        }
    }
    #region implementation omitted
}
```

^ Explanation

Your code should look like this:

```
public class Customer
{
    private string _name = "";
    public string Name
    {
        get
        {
            return this._name;
        }
        set
        {
            if (string.IsNullOrEmpty(value))
                throw new ArgumentException();
            this._name = value;
        }
    }
    #region implementation omitted
    #endregion
}
```

A property is a class member that encapsulates data by declaring two special methods that are invoked by using variable-assignment and variable-reference syntax. This allows you to centralize validation and dynamic calculation logic. This also prevents other code from taking a reference to your data, because it only obtains a reference to a method instead of a variable.

A basic property declaration requires a data type, a property name, a get block if the property is readable, and a set block if the property is writable. For example, the most simple property declaration - an auto-property - is:

```
public class Customer
```

```
public class Customer
{
    public string Name { get; set; }
}
```

When your property requires custom validation or calculation logic, you should implement the individual get and set blocks. These blocks require a storage location to retrieve and store the data, respectively. You generally use a private variable for storage. The get block is written like a function that accepts no parameters and has a return value of the same type as the property. The set block is written like a function that uses a keyword instead of a named parameter, and does not have a return value. For example, a basic property pattern is:

```
public class Customer
{
    private string _name;
    public string Name
    {
        get
        {
            return this._name;
        }
        set
        {
            this._name = value;
        }
    }
}
```

It is common for a set block to error-check the value argument before storing it in the variable. The only way to indicate a validation error is to throw an exception because a set block may not return a value. You could use the ArgumentException class to indicate an invalid value. In the example, the value must not be stored if it is null or empty. If you implement this error check and add code to initialize the variable, you create the complete declaration listed above.

Objective:

Create and Use Types

Subobjective:

Enforce encapsulation.

1.1.27 Question 27 Objective : Create and Use Types Subobjective : Enforce encapsulation

You are developing a service library by using C# and Microsoft Visual Studio 2015. You must implement several service contract interfaces in a single class.

These interfaces are already declared:

```
[ServiceContract]
public interface DropShipServiceV1
{
    [OperationContract]
    StatusCode DropShip(DropShipInfo shipInfo, Product[] products);
}
```

```
[ServiceContract]
public interface DropShipServiceV2
{
    [OperationContract]
    StatusCode DropShip(DropShipInfo shipInfo, Product[] products);
}
```

The implementation details for each of these interfaces are different. You must implement both interfaces in a single class.

Which code block should you use? (Each correct answer presents a complete solution. Choose two.)

Choose the correct answers

```
public class DropShipImplementation
    : DropShipServiceV1, DropShipServiceV2
{
    StatusCode DropShipServiceV1.DropShip(
        DropShipInfo shipInfo, Product[] products)
    { // implementation omitted }
    public StatusCode DropShip(
        DropShipInfo shipInfo, Product[] products)
    { // implementation omitted }
}
```



```
public class DropShipImplementation
    : DropShipServiceV1, DropShipServiceV2
{
    StatusCode DropShipServiceV1.DropShip(
        DropShipInfo shipInfo, Product[] products)
    { // implementation omitted }
    StatusCode DropShipServiceV2.DropShip(
        DropShipInfo shipInfo, Product[] products)
    { // implementation omitted }
}
```



`public class DropShipImplementation
 : DropShipServiceV1, DropShipServiceV2
{
 public StatusCode DropShipServiceV1.DropShip(
 DropShipInfo shipInfo, Product[] products)
 { // implementation omitted }
 public StatusCode DropShipServiceV2.DropShip(
 DropShipInfo shipInfo, Product[] products)
 { // implementation omitted }
}`

`public class DropShipImplementation
 : DropShipServiceV1, DropShipServiceV2
{
 public StatusCode DropShip(
 DropShipInfo shipInfo, Product[] products)
 { // implementation omitted }
 public StatusCode DropShip(
 DropShipInfo shipInfo, Product[] products)
 { // implementation omitted }
}`

1.1.27.1 Correction

```
public class DropShipImplementation
    : DropShipServiceV1, DropShipServiceV2
{
    StatusCode DropShipServiceV1.DropShip(
        DropShipInfo shipInfo, Product[] products)
    { // implementation omitted }
    public StatusCode DropShip(
        DropShipInfo shipInfo, Product[] products)
    { // implementation omitted }
}
```



```
public class DropShipImplementation
    : DropShipServiceV1, DropShipServiceV2
{
    StatusCode DropShipServiceV1.DropShip(
        DropShipInfo shipInfo, Product[] products)
    { // implementation omitted }
    StatusCode DropShipServiceV2.DropShip(
        DropShipInfo shipInfo, Product[] products)
    { // implementation omitted }
}
```


Explanation

You could use this code block:

```
public class DropShipImplementation
    : DropShipServiceV1, DropShipServiceV2
{
    StatusCode DropShipServiceV1.DropShip(
        DropShipInfo shipInfo, Product[] products)
    { // implementation omitted }
    StatusCode DropShipServiceV2.DropShip(
        DropShipInfo shipInfo, Product[] products)
    { // implementation omitted }
}
```

This code explicitly implements both of the service contract interfaces in the DropShipImplementation class. To explicitly implement an interface, you should declare the implementation methods without access modifiers. The method declaration must include the interface name, followed by the dot operator (.), followed by the actual method name. You should use explicit implementation when your code requires a different implementation for two or more methods with the same signature declared in different interfaces.

You could use this code block:

```
public class DropShipImplementation
    : DropShipServiceV1, DropShipServiceV2
{
    StatusCode DropShipServiceV1.DropShip(
        DropShipInfo shipInfo, Product[] products)
    { // implementation omitted }
    public StatusCode DropShip(
        DropShipInfo shipInfo, Product[] products)
    { // implementation omitted }
}
```

This code explicitly implements the DropShipServiceV1 interface. The DropShipServiceV2 interface is implicitly implemented. Generally, you may implicitly implement as many interfaces in a single class as you require. However, when two or more methods have an identical signature, you must explicitly implement all similar methods except for one.

You should not use this code block:

```
public class DropShipImplementation
    : DropShipServiceV1, DropShipServiceV2
{
    public StatusCode DropShip(
        DropShipInfo shipInfo, Product[] products)
    { // implementation omitted }
    public StatusCode DropShip(
        DropShipInfo shipInfo, Product[] products)
    { // implementation omitted }
}
```

This code implicitly implements both interfaces. This code does not compile. You must explicitly implement at least one of the interfaces because the DropShip methods declared in both interfaces have identical signatures. The compiler returns this error: "DropShipImplementation' already defines a member called 'DropShip' with the same parameter types."

You should not use this code block:

```
public class DropShipImplementation
    : DropShipServiceV1, DropShipServiceV2
{
    public StatusCode DropShipServiceV1.DropShip(
        DropShipInfo shipInfo, Product[] products)
    { // implementation omitted }
    public StatusCode DropShipServiceV2.DropShip(
        DropShipInfo shipInfo, Product[] products)
    { // implementation omitted }
}
```

This code specifies access modifiers for the explicitly implemented methods. You must not specify any access modifier on explicitly implemented members. The compiler returns this error: "The modifier 'public' is not valid for this item."

Objective:

Create and Use Types

Subobjective:

Enforce encapsulation.

1.1.28 Question 28 Objective : Create and Use Types Subobjective : Create and implement a class hierarchy

You are developing an application by using C# and Microsoft Visual Studio 2015. You have programmed the following business entity class:

```
public class Employee : IComparable  
{  
    public string FirstName {get; set;}  
    public string LastName {get; set;}  
}
```

You need to implement the `IComparable` pattern.

Which three tasks should you perform? (Choose three.)

Choose the correct answers

- Implement the `CompareTo(...)` method.
- Override the `Equals(...)` method.
- Overload the `==` operator.
- Overload the `!=` operator.
- Implement the `ToString(...)` method.
- Override the `ToString()` method.
- Override the `GetHashCode()` method.

1.1.28.1 Correction

Choose the correct answers

- Implement the CompareTo(...) method.
- Override the Equals(...) method.
- Overload the == operator.
- Overload the != operator.
- Implement the ToString(...) method.
- Override the ToString() method.
- Override the GetHashCode() method.

Explanation

You should perform these tasks:

- * Override the Equals(...) method.
- * Override the GetHashCode() method.
- * Implement the CompareTo(...) method.

The IComparable interface is used by the .NET Framework to describe any data type that has instances that can be compared for sorting purposes. For instance, the Array.Sort() static method can sort an array of any data type (system or custom) if the data type implements IComparable.

To correctly implement the IComparable interface, you should begin by implementing the CompareTo() method. This is the only method defined in the IComparable interface and is the only required component of the pattern. The CompareTo() method is an instance method that compares the current object to an object passed as the only argument. If the current object (the this reference) should sort before the argument, return -1. If the current object is the same value as the argument, return 0. If the current object should sort after the argument, return 1.

In addition to the CompareTo() method, a correct implementation of IComparable should (but is not required to) override the Equals(...) method. This method is inherited from the System.Object type, and is present on all classes in the .NET Framework. The responsibility of the Equals(...) method is to return true if the current object and the argument share the same values, regardless of whether they are the same actual object reference.

Finally, you should override the `GetHashCode()` method. The `GetHashCode()` method is used by the .NET Framework under a variety of circumstances. For instance, if an object is used as a key in a `Dictionary` instance, the instance calls the `GetHashCode()` method. The default implementation inherited from `System.Object` is only valid if two instances of a type are considered equal if and only if the references are equal. In all other cases (such as when overriding `Equals()`), a custom implementation should be provided.

You should not overload the `==` operator. The `==` operator is understood to compare references for equality, whereas the `Equals()` method compares values for equality. Overriding this operator is strongly discouraged for reference types (such as classes), except when you have immutable (unchangeable) classes.

You should not overload the `!=` operator. The `!=` operator and `==` operator should always be overloaded together in the same type.

You should not override the `ToString()` method. The `Object.ToString()` method is not involved in equality comparisons unless you intentionally use it in the `CompareTo()` method.

You should not implement the `ToString(...)` method. This method is defined in the `IFormattable` interface, and is not related to the `IComparable` interface.

Objective:

Create and Use Types

Subobjective:

Create and implement a class hierarchy.

1.1.29 Question 29 Objective : Manage Program Flow Subobjective : Implement program flow

You are writing a C# application by using Visual Studio 2015. The application processes vendors. You define the following:

```
string[] vendors = {"Acme", "Ltd", "Corp"};
```

You need to call the Process function each for each customer in the list.

Which code should you write? (Each correct answer presents a complete solution. Choose two.)

Choose the correct answers

void iterate()
{
 for(int i = 0; i < vendors.Count(); i++)
 {
 Process(vendors[i]);
 }
}

void iterate()
{
 for(int i = 0; i < vendors.GetUpperBound(0); i++)
 {
 Process(vendors[i]);
 }
}

void iterate()
{
 for(int i = 0; i < vendors.GetLength(1); i++)
 {
 Process(vendors[i]);
 }
}

void iterate()
{
 for(int i = 0; i < vendors.Rank; i++)
 {
 Process(vendors[i]);
 }
}



```
void iterate()
{
    for(int i = 0; i < vendors.Length; i++)
    {
        Process(vendors[i]);
    }
}
```

1.1.29.1 Correction

^ Explanation

You should write the block of code below:

```
void iterate()
{
    for(int i = 0; i < vendors.Length; i++)
    {
        Process(vendors[i]);
    }
}
```

This code uses a for loop to iterate from the first element in the array (0) to the last element in the array (Length - 1) by using the Length property of the array.

You can also write the block of code below:

```
void iterate()
{
    for(int i = 0; i < vendors.Count(); i++)
    {
        Process(vendors[i]);
    }
}
```

You should not write the block of code below:

```
void iterate()
{
    for(int i = 0; i < vendors.Rank; i++)
    {
        Process(vendors[i]);
    }
}
```

This code incorrectly uses Rank to obtain the number of elements in the array. The rank of an array is the number of dimensions in an array (in this case, 1). The correct expression is vendors.Length or vendors.Count().

You should not write the block of code below:

```
void iterate()
{
    for(int i = 0; i < vendors.GetLength(1); i++)
    {
        Process(vendors[i]);
    }
}
```

This code incorrectly uses GetLength(1) to obtain the number of elements in the array. GetLength returns the number of elements in the specified dimension. This array has 1 dimension. However, the dimensions are zero-based, so the correct expression is GetLength(0).

You should not write the block of code below:

```
void iterate()
{
    for(int i = 0; i < vendors.GetLength(0); i++)
    {
        Process(vendors[i]);
    }
}
```

This code incorrectly uses GetUpperBound(0) to obtain the number of elements in the array. GetUpperBound returns the highest index of the array for the specified dimension. In this case, the dimension of 0 is correct. However, since the index is zero-based, it is always one less than the actual size, which in this case is 2.

Objective:

Manage Program Flow

Subobjective:

Implement program flow.

1.1.30 Question 30 Objective : Manage Program Flow Subobjective : Implement exception handling

You are writing a C# application by using Visual Studio 2015. The application opens a data file and reads the first line. You create a custom exception for handling read errors named FileReadException.

If a read operation fails, you need to throw a FileReadException. You should ensure that the file is always closed properly.

You need to write the code to properly handle an exception during the read operation.

Which block of code should you write?

Choose the correct answer

System.IO.StreamReader file = new System.IO.StreamReader("data.txt");
try
{
 file.ReadLine();
}
catch (System.IO.IOException e)
{
 throw new FileReadException();
}
file.Close();

System.IO.StreamReader file = new System.IO.StreamReader("data.txt");
try
{
 file.ReadLine();
}
catch (System.IO.IOException e)
{
 throw new FileReadException();
 file.Close();
}

System.IO.StreamReader file = new System.IO.StreamReader("data.txt");
try
{
 file.ReadLine();
}
finally
{
 throw new FileReadException();
 file.Close();
}

```
System.IO.StreamReader file = new System.IO.StreamReader("data.txt");
try
{
    file.ReadLine();
}
catch (System.IO.IOException e)
{
    throw new FileReadException();
}
finally
{
    file.Close();
}
```

1.1.30.1 Correction

^ Explanation

You should write the block of code below:

```
System.IO.StreamReader file = new System.IO.StreamReader("data.txt");
try
{
    file.ReadLine();
}
catch (System.IO.IOException e)
{
    throw new FileReadException();
}
finally
{
    file.Close();
}
```

This code correctly calls the `read` operation in a catch block. It then properly catches an `IOException` error in the catch block and throws the `FileReadException`. It then correctly closes the file in a finally block, which means that the file will be closed whether the operation succeeds or fails.

You should not write the block of code below:

```
System.IO.StreamReader file = new System.IO.StreamReader("data.txt");
try
{
    file.ReadLine();
}
catch (System.IO.IOException e)
{
    throw new FileReadException();
    file.Close();
}
```

This code incorrectly closes the file in the catch block. This means that the file will only be closed if there is an error. Instead, the file should be closed in a finally block.

You should not write the block of code below:

```
System.IO.StreamReader file = new System.IO.StreamReader("data.txt");
try
{
    file.ReadLine();
}
finally
{
    throw new FileReadException();
    file.Close();
}
```

This code improperly throws the `FileReadException` in the finally block, which means that the `FileReadException` will always be thrown. Instead, the `FileReadException` should be placed in a catch block.

You should not write the block of code below:

```
System.IO.StreamReader file = new System.IO.StreamReader("data.txt");
try
{
    file.ReadLine();
}
catch (System.IO.IOException e)
{
    throw new FileReadException();
}
file.Close();
```

This code incorrectly closes the file outside of the try-catch structure, which means that the file will only be closed if the operation fails. Instead, the file should be closed in a finally block.

Objective:

Manage Program Flow

Subobjective:

Implement exception handling.

1.1.31 Question 31 Objective : Debug Application and Implement Security Subobjective : Manage assemblies

You use Microsoft .NET Framework 4.5 to create an application. The application has static references to AssemblyA and AssemblyB. AssemblyA is compiled against .NET Framework 2.0, and AssemblyB is compiled against .NET Framework 4.0.

You need to configure the application so that AssemblyA uses .NET Framework 4.0 when the application is run.

Which configuration should you use?

Choose the correct answer

<configuration>
 <startup >
 <supportedRuntime version="v4.0" />
 </startup>
</configuration>

<configuration>
 <runtime>
 <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
 <dependentAssembly>
 <assemblyIdentity name="AssemblyA" culture="neutral"/>
 <bindingRedirect oldVersion="2.0" newVersion="4.0"/>
 </dependentAssembly>
 </assemblyBinding>
 </runtime>
</configuration>

<configuration>
 <runtime>
 <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
 <dependentAssembly>
 <assemblyIdentity name="AssemblyA.dll" culture="neutral"/>
 <bindingRedirect oldVersion="2.0" newVersion="4.0"/>
 </dependentAssembly>
 </assemblyBinding>
 </runtime>
</configuration>

<configuration>
 <startup >
 <requiredRuntime version="v2.0" />
 <supportedRuntime version="v4.0" />
 </startup>
</configuration>

1.1.31.1 Correction

^ Explanation

You should use this configuration:

```
<configuration>
  <startup>
    <supportedRuntime version="v4.0" />
  </startup>
</configuration>
```

This configuration tells the runtime that the application must use .NET Framework 4.0. This means that all referenced assemblies use .NET Framework 4.0, regardless of the .NET Framework in which they are compiled.

You should not use this configuration:

```
<configuration>
  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
        <assemblyIdentity name="AssemblyA" culture="neutral"/>
        <bindingRedirect oldVersion="2.0" newVersion="4.0"/>
      </dependentAssembly>
    </assemblyBinding>
  </runtime>
</configuration>
```

This configuration attempts to redirect AssemblyA from version 2.0 to version 4.0. It does not force an application to use a specific version of the .NET Framework. You should only use this configuration to redirect references assemblies, not to have the application use a specific version of the .NET Framework.

You should not use this configuration:

```
<configuration>
  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
        <assemblyIdentity name="AssemblyA.dll" culture="neutral"/>
        <bindingRedirect oldVersion="2.0" newVersion="4.0"/>
      </dependentAssembly>
    </assemblyBinding>
  </runtime>
</configuration>
```

This configuration attempts to redirect AssemblyA.dll from version 2.0 to version 4.0. It does not force an application to use a specific version of the .NET Framework. You should only use this configuration to redirect references assemblies, not to have the application use a specific version of the .NET Framework. Also, when specifying the name attribute of the assemblyIdentity element, you should not include the file extension of the assembly.

You should not use this configuration:

```
<configuration>
<startup >
  <requiredRuntime version="v2.0" />
  <supportedRuntime version="v4.0" />
</startup>
</configuration>
```

You should only use the requiredRuntime element for applications that support .NET Framework 1.0. Applications that use later versions of the .NET Framework must use the supportedRuntime element.

Objective:

Debug Applications and Implement Security

Subobjective:

Manage assemblies.

1.1.32 Question 32 Objective : Debug Application and Implement Security Subobjective : Debug an application

You use Microsoft .NET Framework 4.5 to create an application. When you debug the application, you realize that an instance of the Customer class shows the type name of the Customer as the Value in the Watch window.

You need to ensure that the Value property in the Watch window shows the Name property of the Customer class during debugging.

Which attribute should you apply to the Customer class?

Choose the correct answer

- [DebuggerDisplay("Name")]
- [DebuggerDisplay("{Customer.Name}")]
- [DebuggerDisplay("{Name}")]
- [DebuggerDisplay("Customer.Name")]

1.1.32.1 Correction

Choose the correct answer

[DebuggerDisplay("Name")]

[DebuggerDisplay("{Customer.Name}")]

[DebuggerDisplay("{Name}")]

[DebuggerDisplay("Customer.Name")]

Explanation

You should apply this attribute to the Customer class:

[DebuggerDisplay("{Name}")]

This attribute specifies how a type should be displayed in the Watch window of the debugger. Literals between curly braces {} represent expressions. {Name} indicates that the value of the Name property should be displayed for the Customer instance.

You should not use this code segment:

[DebuggerDisplay("Name")]

This would display the string Name for each Customer instance.

You should not use this code segment:

[DebuggerDisplay("{Customer.Name}")]

This would attempt to retrieve the Name property of a Customer instance from the Customer class.

You should not use this code segment:

[DebuggerDisplay("Customer.Name")]

This would display the string Customer.Name for each Customer instance.

1.1.33 Question 33 Objective : Debug Application and Implement Security Subobjective : Perform symmetric and asymmetric encryption

You use Microsoft .NET Framework 4.5 to create an application. The application must send confidential data to a Web service. You plan to use a shared key for encryption and decryption.

You need to choose a class for encrypting the data.

Which class should you use?

Choose the correct answer

- DSACryptoServiceProvider
- RijndaelManaged
- SHA1Managed
- RSACryptoServiceProvider

1.1.33.1 Correction

^ Explanation

You should use the RijndaelManaged class. This class implements shared-key, or symmetric encryption, which is required in this scenario. With symmetric encryption, both your application and the Web service must know the encryption key that is used.

You should not use RSACryptoServiceProvider. This class implements public key or asymmetric encryption by using the Rivest, Shamir, Adleman (RSA) algorithm. With asymmetric encryption, a public key is used to encrypt the data, but a private key is used to decrypt the data. The public key is known to all entities that must be able to encrypt the data. The private key is known only to the entity that should decrypt the data. The public and private key pair cannot be used with other public or private keys. This does not work in this scenario because you plan to use one shared key.

You should not use DSACryptoServiceProvider. Like RSACryptoServiceProvider, this class implements asymmetric encryption. It uses the Digital Signature Algorithm (DSA). This does not work in this scenario because you plan to use one shared key.

You should not use SHA1Managed. This class uses the Secure Hash Algorithm (SHA) 1 to compute a unique hash of data. Hashing is one way, and it cannot be reversed. If you hash the data and send it to the Web service, the Web service cannot reverse the process.

Objective:

Debug Applications and Implement Security

Subobjective:

1.1.34 Question 34 Objective : Debug Application and Implement Security Subobjective : Debug an application

You are creating a C# native application by using Visual Studio 2015. You need to debug a crash that is occurring in a third party COM object.

What should you do?

Choose the correct answer

- Select the Load DLL Exports option in the Debugging General options.
- Select the Native option in the Debugging Just-in-Time options.
- Select the Microsoft Symbol Servers option in the Debugging Symbols options.
- Set the All debug output option to On in the Debugging Output Window options.

1.1.34.1 Correction

Choose the correct answer

- Select the Load DLL Exports option in the Debugging General options.
- Select the Native option in the Debugging Just-in-Time options.
- Select the Microsoft Symbol Servers option in the Debugging Symbols options.
- Set the All debug output option to On in the Debugging Output Window options.

Explanation

You should choose select the Load DLL Exports option in the Debugging General options. This feature scans and exports debug symbols from external files including COM objects. Because this adds overhead, the option is turned off by default.

You should not choose select the Microsoft Symbol Servers option in the Debugging Symbols options. This feature downloads debugging symbols for Microsoft external libraries and third-party components that have symbols published on a server, but would not have the symbols for a third party COM object.

You should not choose select the Native option in the Debugging Just-in-Time options. This option allows your native code to start a debugger even when it is running outside of Visual Studio, but this will not allow debugging of external COM objects.

You should not choose set the All debug output option to On in the Debugging Output Window options. This will direct all debug output to be directed to the Visual Studio Output Window, but it will not help in debugging external COM objects.

1.1.35 Question 35 Objective : Create and Use Types Subobjective : Manipulate strings

You are creating a C# application by using Visual Studio 2015. The application writes random words to a document as part of a process that creates dummy documents of various sizes.

You have created the code below. (Line numbers are included for reference only.)

```
01 public class DocBuilder
02 {
03     List<string> dict;
04     StringBuilder doc;
05     DocBuilder()
06     {
07         InitDictionary();
08     }
09     StringBuilder BuildDoc(ulong words)
10    {
11        int index;
12        doc = new StringBuilder(5000, 40000);
13        Random pick = new Random();
14        for(ulong i = 0; i < words; i++)
15        {
16
17            {
18                index = pick.Next(dict.Count());
19                doc.Append(dict[index]);
20            }
21        }
22        return doc;
23    }
24 }
```

The `InitDictionary` function initializes the `dict` list and fills it with words.

You need to protect `doc` from exceeding the 40,000 characters, while still allowing the string to reach the maximum allowed length.

Which line should you add at line 15?

Choose the correct answer

- if(`doc.Capacity <= doc.MaxCapacity`)
- if(`doc.Length + dict[index].Length <= doc.MaxCapacity`)
- if(`doc.Length <= doc.MaxCapacity`)
- if(`doc.Length + dict[index].Length <= doc.Capacity`)

1.1.35.1 Correction

if(doc.Capacity <= doc.MaxCapacity)

if(doc.Length + dict[index].Length <= doc.MaxCapacity)

if(doc.Length <= doc.MaxCapacity)

if(doc.Length + dict[index].Length <= doc.Capacity)

^ Explanation

You should choose the line of code below:

```
if(doc.Length + dict[index].Length <= doc.MaxCapacity)
```

The `StringBuilder` class dynamically allocates memory as necessary to build the string by increasing the capacity until the maximum capacity defined as the second parameter of the constructor is reached. Although the `Append` method can cause the `StringBuilder` object to grow beyond the defined maximum capacity, this is not desired in this scenario. The `Length` property indicates the current length of the string builder, and the `MaxCapacity` property indicates the maximum capacity of the `StringBuilder`. If the current length plus the length of the new string are less than or equal to the maximum capacity, then it is appropriate to append the new string.

You should not choose the line of code below:

```
if(doc.Length <= doc.MaxCapacity)
```

The `Length` property indicates the current length of the string in string builder. However, at any time, the `Length` is probably less than `MaxCapacity`. The correct line of code is `if(doc.Length + dict[index] <= doc.MaxCapacity)`, which will prevent the next append from exceeding the maximum capacity.

You should not choose the line of code below:

```
if(doc.Length + dict[index] <= doc.Capacity)
```

The `Capacity` property indicates the current capacity of the string, but it does not reflect the maximum capacity of the string. The starting value of `Capacity` is the value passed as the first parameter to the constructor. In this case, `Capacity` begins at 5,000 characters. If the capacity is less than the maximum capacity, then you can still append strings to the string builder and it will allocate more memory. The correct test is `if(doc.Length + dict[index].Length <= doc.MaxCapacity)` to ensure that the current string to append will not exceed the maximum capacity of the string builder.

You should not choose the line of code below:

```
if(doc.Capacity <= doc.MaxCapacity)
```

The Capacity property indicates the current capacity of the string, but it does not reflect the number of characters currently stored in the StringBuilder. If the capacity is less than the maximum capacity, then you can still append strings to the string builder and it will allocate more memory. The correct test is if(doc.Length + dict[index].Length <= doc.MaxCapacity) to ensure that the current string to append will not exceed the maximum capacity of the string builder.

1.1.36 Question 36 Objective : Manage Program Flow Subobjective : Implement exception handling

You use Microsoft .NET Framework 4.5 to create an application. You must call a method named RetrieveReport. This method might throw exceptions of different types. Regardless of whether or not RetrieveReport throws an exception, you need to call a method named EndConnection.

You need to write the code to accomplish your goal.

Which code segment should you use?

Choose the correct answer

try
 {
 RetrieveReport();
 }
 catch
 {
 EndConnection();
 }

try
 {
 RetrieveReport();
 }
 finally
 {
 EndConnection();
 }

try
 {
 RetrieveReport();
 }
 catch (Exception)
 {
 EndConnection();
 }

try
 {
 RetrieveReport();
 }
 catch (Exception ex)
 {
 EndConnection();
 }

1.1.36.1 Correction

^ Explanation

You should use this code segment:

```
try
{
    RetrieveReport();
}
finally
{
    EndConnection();
}
```

Within a try block, you should place code that might throw an exception. If an exception is thrown and not handled in a catch block, execution resumes up the call stack until a catch block is found that handles the exception. However, code within a finally block always executes before execution resumes up the call stack. By calling the EndConnection method within the finally block, you ensure that it gets called even if RetrieveReport throws an exception.

You should not use this code segment:

```
try
{
    RetrieveReport();
}
catch
{
    EndConnection();
}
```

This code calls the EndConnection method within the catch block. This would prevent the method from being called if no exception was thrown. When you specify the catch statement with no parameters, the catch block handles exceptions of all types.

You should not use this code segment:

```
try
{
    RetrieveReport();
}
catch (Exception)
{
    EndConnection();
}
```

This code calls the EndConnection method within the catch block. This would prevent the method from being called if no exception was thrown. When you specify a catch statement with only the exception type as a parameter, it handles all exceptions of that type and its inherited types.

You should not use this code segment:

```
try
{
    RetrieveReport();
}
catch (Exception ex)
{
    EndConnection();
}
```

This code calls the EndConnection method within the catch block. This would prevent the method from being called if no exception was thrown. When you specify a catch statement with the exception type and exception instance as a parameter, it handles all exceptions of that type and its inherited types. It also allows you to access properties of the exception instance.

Objective:

Manage Program Flow

Subobjective:

• Handle errors in your program

1.1.37 Question 37 Objective : Manage Program Flow Subobjective : Create and implement events and callbacks.

You use Microsoft .NET Framework 4.5 to create an application. The .NET Framework UnhandledException event is defined as follows:

```
public event EventHandler UnhandledException;
```

The .NET Framework UnhandledExceptionEventHandler delegate is defined as follows:

```
public delegate void UnhandledExceptionEventHandler  
(object sender, UnhandledExceptionEventArgs e);
```

You must create an event handler named HandleApplicationError that handles the UnhandledException event.

You need to define the event handler's method signature.

Which code segment should you use?

Choose the correct answer

`private void HandleApplicationError(UnhandledExceptionEventHandler sender,
UnhandledExceptionEventArgs e)
{
}`

`private void HandleApplicationError(object sender, UnhandledExceptionEventArgs e)
{
}`

`private void HandleApplicationError(object sender)
{
}`

`private void HandleApplicationError(UnhandledExceptionEventArgs e)
{
}`

1.1.37.1 Correction

^ Explanation

You should use this code segment:

```
private void HandleApplicationError(object sender, UnhandledExceptionEventArgs e)
{
}
```

The event handler's signature correctly matches the signature of the corresponding delegate. In this scenario, the `UnhandledExceptionEventHandler` delegate defines two parameters. The type of the first parameter is `Object`, and the type of the second parameter is `UnhandledExceptionEventArgs`.

You should not use this code segment:

```
private void HandleApplicationError(UnhandledExceptionEventHandler sender,
UnhandledExceptionEventArgs e)
{
}
```

The signature of the event handler must match the signature of the delegate. The type of the first parameter must be `Object`, not `UnhandledExceptionEventHandler`.

You should not use this code segment:

```
private void HandleApplicationError(object sender)
{
}
```

The signature of the event handler must match the signature of the delegate. This method contains only one parameter.

Objective:

Manage Program Flow

Subobjective:

Create and implement events and callbacks.

1.1.38 Question 38 Objective : Debug Application and Implement Security. Subobjective : Perform symmetric and asymmetric encryption.

You use Microsoft .NET Framework 4.5 to create an application that runs on Microsoft Windows 7. The application allows users to save credit card numbers to the local computer. The credit card number is stored in a byte array named data. You must generate an encrypted representation of the byte array. Only the user who runs the application when it is encrypted should be able to decrypt the data. The original byte array should not be modified.

You need to write code to accomplish your goal.

Which code segment should you use?

Choose the correct answer

- ProtectedData.Protect(data, null, DataProtectionScope.LocalMachine);
- ProtectedData.Protect(data, null, DataProtectionScope.CurrentUser);
- ProtectedMemory.Protect(data, MemoryProtectionScope.SameProcess);
- ProtectedMemory.Protect(data, MemoryProtectionScope.SameLogon);

1.1.38.1 Correction

^ Explanation

You should use this code segment:

```
ProtectedData.Protect(data, null, DataProtectionScope.CurrentUser);
```

The ProtectedData class represents a managed implementation of the Data Protection Application Programming Interface (DPAPI). It allows you to encrypt data by using information from the current user or computer account. Only that user or computer can decrypt the data. The DataProtectionScope.CurrentUser enumeration member indicates that only the current user should be able to decrypt the data. The method returns an encrypted version of the data and does not modify the original data.

You should not use this code segment:

```
ProtectedData.Protect(data, null, DataProtectionScope.LocalMachine);
```

The DataProtectionScope.LocalMachine enumeration member indicates that any user of the current computer can decrypt the data, which violates the requirement in this scenario. This member is useful if you are saving data to a network share, and there are multiple users on the same computer who need to access the same data. It prevents users from other computers from accessing the data.

You should not use this code segment:

```
ProtectedMemory.Protect(data, MemoryProtectionScope.SameLogon);
```

The ProtectedMemory class also represents a managed implementation of DPAPI. However, it modifies the data that you pass to it, which violates the requirement in this scenario. The MemoryProtectionScope.SameLogon enumeration member indicates that only the user who encrypted the data can decrypt it.

You should not use this code segment:

```
ProtectedMemory.Protect(data, MemoryProtectionScope.SameProcess);
```

The MemoryProtectionScope.SameProcess enumeration member indicates that all threads in the same process can decrypt the data.

1.1.39 Question 39 Objective : Create and Use Types Subobjective : Create and implement a class hierarchy

You are developing a class library by using C# and Microsoft Visual Studio 2015. You need to develop a hierarchy of classes to represent different kinds of customers. These classes will be used in other applications.

You need to declare the Customer class. This class serves as the base class for all Customer classes. You should not be able to create new instances of the Customer class. The Customer-derived classes you declare in this library must not be visible to other assemblies. Other developers derive the Customer class in their own assemblies.

Which declaration satisfies these requirements?

Choose the correct answer

- internal sealed class Customer { /* implementation omitted */ }
- internal abstract class Customer { /* implementation omitted */ }
- public sealed class Customer { /* implementation omitted */ }
- public abstract class Customer { /* implementation omitted */ }

1.1.39.1 Correction



```
public abstract class Customer { /* implementation omitted */ }
```

Explanation

You should use this declaration:

```
public abstract class Customer { /* implementation omitted */ }
```

The public access modifier allows all assemblies that take a reference to your library to access the Customer type. The abstract keyword causes the compiler to reject all attempts to create new instances of the Customer class, even in your own code. The only way to create a new instance that uses the Customer type is to use the new keyword with a class that inherits, directly or indirectly, from Customer.

You should not use this declaration:

```
internal abstract class Customer { /* implementation omitted */ }
```

The internal access modifier incorrectly limits the visibility of the Customer class only to your library assembly. No other assemblies have access to the Customer class. This is incorrect, because other developers inherit from this class in their own libraries. You should use the public access modifier to allow other developers to derive Customer.

You should not use this declaration:

```
public sealed class Customer { /* implementation omitted */ }
```

The public access modifier allows all assemblies that take a reference to your library to access the Customer type. The sealed keyword causes the compiler to reject all attempts to derive the Customer class. You should use the abstract keyword instead, because you have Customer-derived classes in your own assembly, and other developers have Customer-derived classes in their assemblies.

You should not use this declaration:

```
internal sealed class Customer { /* implementation omitted */ }
```

The internal access modifier incorrectly limits the visibility of the Customer class only to your library assembly. You should use the public access modifier instead, because other developers must have access to the Customer type to derive from it. The sealed keyword causes the compiler to reject all attempts to derive the Customer class. You should use the abstract keyword instead, because you have Customer-derived classes in your own assembly.

1.1.40 Question 40 Objective : Manage Program Flow Subobjective : Manage multithreading

You are writing a C# application by using Visual Studio 2015. The application processes payments. You want to check the balance before the payment is taken from the account.

You need to ensure that the block of code to check the balance can only be accessed by a single thread at one time.

Which block of code should you write?

Choose the correct answer

```
class Pay
{
    private float balance;
    private string l = "lock";
    public void Payment(float amount)
    {
        lock (l)
        {
            if(amount < balance)
            {
                balance -= amount;
            }
        }
    }
}
```

```
public class Pay
{
    private float balance;
    private Object l = new Object();
    public void Payment(float amount)
    {
        lock (l)
        {
            if(amount < balance)
            {
                balance -= amount;
            }
        }
    }
}
```

public class Pay
{
 private float balance;
 public void Payment(float amount)
 {
 lock (this)
 {
 if(amount < balance)
 {
 balance -= amount;
 }
 }
 }
}

public class Pay
{
 private float balance;
 public Object l = new Object();
 public void Payment(float amount)
 {
 lock (l)
 {
 if(amount < balance)
 {
 balance -= amount;
 }
 }
 }
}

1.1.40.1 Correction

```
public class Pay
{
    private float balance;
    private Object l = new Object();
    public void Payment(float amount)
    {
        lock (l)
        {
            if(amount < balance)
            {
                balance -= amount;
            }
        }
    }
}
```

Explanation

You should use the block of code below:

```
public class Pay
{
    private float balance;
    private Object l = new Object();
    public void Payment(float amount)
    {
        lock (l)
        {
            if(amount < balance)
            {
                balance -= amount;
            }
        }
    }
}
```

This code properly creates the lock as a private object and then uses the lock on the block of code that needs to have exclusive access.

You should not use the block of code below:

```
class Pay
{
    private float balance;
    private string l = "lock";
    public void Payment(float amount)
    {
        lock (l)
        {
            if(amount < balance)
            {
                balance -= amount;
            }
        }
    }
}
```

This code uses a literal string for the lock. This is not good practice because literal strings are intrinsically created as global objects in C#. If another block of code placed a lock on the same literal string, a deadlock condition could result.

You should not use the block of code below:

```
public class Pay
{
    private float balance;
    public void Payment(float amount)
    {
        lock (this)
        {
            if(amount < balance)
            {
                balance -= amount;
            }
        }
    }
}
```

This code places a lock on a public object (the Pay class), which is not a good practice because other code would be allowed to place a simultaneous lock on the same Pay object, causing a deadlock condition.

You should not use the block of code below:

```
public class Pay
{
    private float balance;
    public Object l = new Object();
    public void Payment(float amount)
    {
        lock (l)
        {
            if(amount < balance)
            {
                balance -= amount;
            }
        }
    }
}
```

This code places a lock on a public object (the l object), which is not a good practice because other code would be allowed to place a simultaneous lock on the l object, causing a deadlock condition.

1.1.41 Question 41 Objective : Manage Program Flow Subobjective : Create and implement events and callbacks.

You use Microsoft .NET Framework 4.5 to create an application. You need to declare an event named MouseMove that uses a MouseEventArgs instance as the event argument when the event is raised.

You need to declare the event. You must ensure that the event can be raised from within your class only.

Which code segment should you use?

Choose the correct answer

- public event EventHandler<MouseEventArgs> MouseMove;
- public delegate EventHandler<MouseEventArgs> MouseMove(int x, int y);
- public delegate EventHandler MouseMove(object sender, MouseEventArgs e);
- public EventHandler<MouseEventArgs> MouseMove;

1.1.41.1 Correction

^ Explanation

You should use this code segment:

public event EventHandler<MouseEventArgs> MouseMove; To define an event, you must specify the event keyword. This keyword instructs the compiler that the MouseMove instance represents a multi-cast delegate that can only be invoked from within the class where the event is defined. The EventHandler class represents a delegate that contains two parameters: the type of the first parameter is Object, and the type of the second parameter is EventArgs. By using the generic version of EventHandler, which is defined as EventHandler<T>, you can specify the type that should be used as the second parameter for the delegate that represents the event handler. In this scenario, EventHandler<MouseEventArgs> specifies that the event handler must adhere to this signature:

```
void SomeEventHandler(object sender, MouseEventArgs e)
```

You should not use this code segment:

```
public EventHandler<MouseEventArgs> MouseMove;
```

This code does not specify the event keyword. This allows any code that can publicly access the MouseMove instance to raise the MouseMove event.

You should not use this code segment:

```
public delegate EventHandler<MouseEventArgs> MouseMove(int x, int y);
```

This code does not declare an event. It defines a delegate. A delegate is simply a data type that describes a method signature.

You should not use this code segment:

```
public delegate EventHandler MouseMove(object sender, MouseEventArgs e);
```

This code does not declare an event. It defines a delegate. A delegate is simply a data type that describes a method signature.

Objective:

Manage Program Flow

Subobjective:

Create and implement events and callbacks.

1.1.42 Question 42 Objective : Manage Program Flow Subobjective : Create and implement events and callbacks.

You use Microsoft .NET Framework 4.5 to create an application. This delegate is defined:

```
public delegate string Combine(string s1, string s2);
```

You must declare an instance of the Combine delegate that uses an anonymous method to concatenate the string instances represented by s1 and s2 and return the concatenated string.

You need to declare the delegate instance.

Which code segment should you use?

Choose the correct answer

`Combine combine = delegate(string s1, string s2) { return s1 + s2; };`

`Combine combine = delegate(s1, s1) { s1 + s2 };`

`Combine combine = delegate() { return s1 + s2; };`

`Combine combine = delegate(s1, s2) { return s1 + s2; };`

1.1.42.1 Correction

Explanation

You should use this code statement:

```
Combine combine = delegate(string s1, string s2) { return s1 + s2; };
```

The expression on the right side of the = operator represents an anonymous method. An anonymous method is one that uses the special name delegate. Anonymous methods allow you to assign delegate instances to method bodies without having to explicitly define a method within a class. This is useful if the method will only be called by invocation of the delegate.

You should not use this code statement:

```
Combine combine = delegate() { return s1 + s2; };
```

This anonymous method does not define any parameters. The parameters must match the parameters defined by the Combine delegate.

You should not use this code statement:

```
Combine combine = delegate(s1, s1) { s1 + s2 };
```

This anonymous method does not specify the types of the parameters s1 and s2. Unlike with a lambda expression, you must specify the types of the parameters for anonymous methods. Also, the body inside the curly braces {} must represent a valid code statement. Therefore, you must include the return keyword, and you must end the statement with a semicolon.

You should not use this code statement:

```
Combine combine = delegate(s1, s2) { return s1 + s2; };
```

This anonymous method does not specify the types of the parameters s1 and s2. Unlike with a lambda expression, you must specify the types of the parameters for anonymous methods.

Objective:

Manage Program Flow

Subobjective:

Create and implement events and callbacks.

1.1.43 Question 43 Objective : Manage Program Flow Subobjective : Implement exception handling

You use Microsoft .NET Framework 4.5 to create an application. You must call a method named ImportData that might throw exceptions of multiple types. If it throws an instance of type SqlException, you must call the LogSqlError method and pass to it the SqlException instance as a parameter. If it throws any other type of exception, you must call the LogError method and pass to it the instance of the exception that was thrown.

You need to write code to accomplish your goal.

Which code segment should you use?

Choose the correct answer

try
{
 ImportData();
}
catch(SqlException ex)
{
 LogSqlError(ex)
 Exception ex2 = ex;

try
{
 ImportData();
}
catch(Exception ex)
{
 LogError(ex)
}
catch(SqlException ex)
{
 LogSqlError(ex);
}

try
{
 ImportData();
}
catch(Exception ex)
{
 SqlException ex2 = (SqlException)ex;
 LogSqlError(ex2)
 LogError(ex);
}

1.1.43.1 Correction

```
try
{
    ImportData();
}
catch(SqlException ex)
{
    LogSqlError(ex)
}
catch(Exception ex)
{
    LogError(ex);
}
```

1.1.44 Question 44 Objective : Manage Program Flow Subobjective : Implement exception handling

You use Microsoft .NET Framework 4.5 to create a reusable component.

You need to create a user-defined exception for your component. Your solution must adhere to Microsoft best practices.

From which base class should you derive your exception class?

Choose the correct answer

- Exception
- InvalidOperationException
- COMException
- SystemException

1.1.44.1 Correction

<input checked="" type="checkbox"/>	Exception	Faites glisser le curseur autour de la zone à capturer.
<input checked="" type="checkbox"/>	InvalidOperationException	
<input checked="" type="checkbox"/>	COMException	Outil Capture d'écran dition Utilis ?  SystemException

Explanation

You should derive your exception class from `Exception`. This adheres to Microsoft best practices. The `Exception` class represents the base type from which all exceptions should be derived.

You should not derive your exception class from `SystemException`. This is the base class from which all exceptions in the `System` namespace are derived. This would not adhere to Microsoft best practices.

You should not derive your exception class from `COMException`. The common language runtime (CLR) throws exceptions of this type when an unrecognized HRESULT is returned after calling a method on a COM object. Deriving your exception from this class would not adhere to Microsoft best practices.

You should not derive your exception class from `InvalidOperationException`. You should throw this exception whenever an action is performed on an object in an invalid state. Deriving your exception from this class would not adhere to Microsoft best practices.

Objective:

Manage Program Flow

Subobjective:

Implement exception handling.

References

1.1.45 Question 45 Objective : Manage Program Flow Subobjective : Implement exception handling

You use Microsoft .NET Framework 4.5 to create an application. You must create a class named MappingException to serve as a user-defined exception for your application. Code that throws instances of MappingException must be able to wrap other handled exceptions with your exception.

You need to create the MappingException class. Your solution must adhere to Microsoft best practices.

Which code segment should you use?

Choose the correct answer

public class MappingException : SystemException
{
 public MappingException(string msg, Exception inner) {}
}

public class MappingException
{
 public MappingException(){}
 public MappingException(string msg) {}
 public MappingException(string msg, Exception inner) {}
}

public class MappingException : Exception
{
 public MappingException(){}
 public MappingException(string msg) : base(msg) {}
 public MappingException(string msg, Exception inner) : base(msg, inner) {}
}

public class MappingException
{
 public MappingException(string msg, Exception inner) {}
}

1.1.45.1 Correction

```
public class MappingException : Exception
{
    public MappingException(){}
    public MappingException(string msg) : base(msg) {}
    public MappingException(string msg, Exception inner) : base(msg, inner) {}
}
```

1.1.46 Question 46 Objective : Manage Program Flow Subobjective : Implement exception handling

You are writing a C# application by using Visual Studio 2015. The application opens a data file and reads the first line. You create a custom exception for handling read errors named FileReadException.

If a read operation fails, you need to throw a FileReadException. You should ensure that the file is always closed properly.

You need to write the code to properly handle an exception during the read operation.

Which block of code should you write?

Choose the correct answer

```
System.IO.StreamReader file = new System.IO.StreamReader("data.txt");
try
{
    file.ReadLine();
}
catch (System.IO.IOException e)
{
    throw new FileReadException();
}
finally
{
    file.Close();
}
```



```
System.IO.StreamReader file = new System.IO.StreamReader("data.txt");
try
{
    file.ReadLine();
}
catch (System.IO.IOException e)
{
    throw new FileReadException();
}
file.Close();
```



```
System.IO.StreamReader file = new System.IO.StreamReader("data.txt");
try
{
    file.ReadLine();
}
catch (System.IO.IOException e)
{
    throw new FileReadException();
    file.Close();
}
```



```
System.IO.StreamReader file = new System.IO.StreamReader("data.txt");
try
{
    file.ReadLine();
}
finally
{
    throw new FileReadException();
    file.Close();
}
```



1.1.46.1 Correction

```
System.IO.StreamReader file = new System.IO.StreamReader("data.txt");
try
{
    file.ReadLine();
}
catch (System.IO.IOException e)
{
    throw new FileReadException();
}
finally
{
    file.Close();
}
```

1.1.47 Question 47 Objective : Manage Program Flow Subobjective : Implement exception handling

You are writing a C# application by using Visual Studio 2015. The application opens a file and reads a single line of data. You create a function named LogError for logging any errors.

If a read operation fails, you need to call LogError. You should then rethrow the error so that the calling function can handle it.

You need to write the code to properly handle an exception during the read operation.

Which block of code should you write?

Choose the correct answer

```
System.IO.StreamReader file = new System.IO.StreamReader("data.txt");
try
{
    file.ReadLine();
}
catch (System.IO.IOException e)
{
    LogError(e.Message);
    throw e;
}
```



```
System.IO.StreamReader file = new System.IO.StreamReader("data.txt");
try
{
    file.ReadLine();
}
catch (System.IO.IOException e)
{
    LogError(e.Message);
    throw;
}
```



```
System.IO.StreamReader file = new System.IO.StreamReader("data.txt");
try
{
    file.ReadLine();
}
catch (System.IO.IOException e)
{
    LogError(e.Message);
}
finally
{
    throw;
}
```



```
System.IO.StreamReader file = new System.IO.StreamReader("data.txt");
try
{
    file.ReadLine();
}

catch (System.IO.IOException e)
{
    throw;
}
finally
{
    LogError(e.Message);
}
```

1.1.47.1 Correction

^ Explanation

You should write the block of code below:

```
System.IO.StreamReader file = new System.IO.StreamReader("data.txt");
try
{
    file.ReadLine();
}
catch (System.IO.IOException e)
{
    LogError(e.Message);
    throw;
}
```

This block of code correctly places the read operation inside a try block. It then places the call to LogError in a catch block and uses the throw statement to rethrow the exception. This will allow the function that called this function (or any function higher up the call stack) to handle the error. Using throw without any arguments preserves the error object as well, so that it will be available to the parent error handler.

You should not write the block of code below:

```
System.IO.StreamReader file = new System.IO.StreamReader("data.txt");
try
{
    file.ReadLine();
}
catch (System.IO.IOException e)
{
    throw;
}
finally
{
    LogError(e.Message);
}
```

This block of code incorrectly places the LogError call in a finally block. This means that the LogError call will be made whether the operation succeeds or fails. Instead, you should place the call to LogError in the catch block.

You should not write the block of code below:

```
System.IO.StreamReader file = new System.IO.StreamReader("data.txt");
try
{
    file.ReadLine();
}
catch (System.IO.IOException e)
{
    LogError(e.Message);
    throw e;
}
```

This block of code uses the `throw e` statement to rethrow the exception. This will allow the function that called this function (or any function higher up the call stack) to handle the error, but rethrowing with the error object creates a new error object. Therefore, the data from the original error will not be available to the parent's error handler. Instead, you should simply call `throw` without anything following the `throw` statement to preserve the error details.

You should not write the block of code below:

```
System.IO.StreamReader file = new System.IO.StreamReader("data.txt");
try
{
    file.ReadLine();
}
catch (System.IO.IOException e)
{
    LogError(e.Message);
}
finally
{
    throw;
}
```

This block of code incorrectly places the `throw` statement in a `finally` block. This means that the error will be rethrown whether the operation succeeds or fails. Instead, you should place the `throw` statement in the `catch` block.

1.1.48 Question 48 Objective : Manage Program Flow Subobjective : Implement program flow

You are developing a business data library by using C# and Microsoft Visual Studio 2015. You are working on a method to create Employee objects from database records. You have created several classes that derive Employee: HourlyEmployee, SalariedEmployee, and Manager. Your database has several codes that determine which class you should use. You have created the EmployeeType class to contain several constant strings to represent these codes.

In your database, there is an additional code for Directors. When the Director code is used, your code should create a Manager object and set the Options property to the value stored in the protected static directorOptions field of the Employee class.

Complete the method implementation by dragging statement fragments from the left to the appropriate location on the right. You may use items more than once, and you do not have to use each item.

Drag and drop the answers

```
public static Employee CreateEmployee
    (string firstName, string lastName, string code)
{
    int options = 0;
    Manager newManager = null;
    switch (code)
    {
        case EmployeeType.Hourly:
            return new HourlyEmployee()
                { FirstName = firstName, LastName = lastName };
        case EmployeeType.Salary:
            return new SalariedEmployee()
                { FirstName = firstName, LastName = lastName };
        case EmployeeType.Director:
            options = Employee.directorOptions;
            return new Manager()
                { FirstName = firstName,
                  LastName = lastName, Options = options };
        default:
            throw new ArgumentException(code + " not recognized");
    }
    return newManager;
}
```

1.1.48.1 Correction

Drag and drop the answers

```
public static Employee CreateEmployee
    (string firstName, string lastName, string code)
{
    int options = 0;
    Manager newManager = null;
    switch (code)
    {
        case EmployeeType.Hourly:
            return new HourlyEmployee()
            { FirstName = firstName, LastName = lastName };
        case EmployeeType.Salary:
            return new SalariedEmployee()
            { FirstName = firstName, LastName = lastName };
        case EmployeeType.Director:
            options = Employee.directorOptions;
            goto case EmployeeType.Manager;
        case EmployeeType.Manager:
            newManager = new Manager { FirstName = firstName,
                LastName = lastName, Options = options };
            break;
        default:
            throw new ArgumentException(code + " not recognized");
    }
    return newManager;
}
```

Explanation

To evaluate a single expression against a list of possible outcomes, C# provides the switch statement. While similar to switch statements found in other C-derived languages, C# has some specific requirements.

The basic syntax of a switch statement is:

```
switch (<expression>)
{
    case <const 1>:
        // logic goes here
        <jump statement>;
    case <const 2>;
    case <const 3>;
        // logic goes here
        <jump statement>;
    case <const n>;
        // logic goes here
        <jump statement>;
    default:
        // logic goes here
        <jump statement>;
}
```

* Where the data type of <expression> is numeric, boolean, string, or an enum type.

* Where <expression> is any valid C# expression.

- * Where <const 1> through <const n> are values that can be evaluated statically (by the compiler). Literal expressions and constants are valid, other kinds of expressions are not.
- * Where the <jump statement> is required and is one of these statements: break, goto, goto case, continue, return, throw.
- * Where case expressions do not fall through to the next case expression unless there is a single code block shared by the entire sequence.

Even though it is possible to rewrite most switch statements as a series of if ... else if ... else blocks, the if statement and switch statement do not share common keywords, such as else.

Objective:

Manage Program Flow

Subobjective:

Implement program flow.

<https://www.dotnetperls.com/goto>

When used in a switch statement, goto can enhance performance and reduce code size.

1.1.49 Question 49 Objective : Manage Program Flow Subobjective : Implement program flow

You are developing the checkout process for an online sales application by using C# and Visual Studio 2015. You have created a Units class to contain the information and logic for the different types of units your products might be packaged in, for example, each , box of 10 , display box of 24. Among other properties, the Units class has properties to return the user-friendly text for single or plural forms of the unit:

```
public class Units
{
    SingularText { get; set; }
    PluralText { get; set; }
    // remaining implementation omitted
}
```

Which code block should you use to assemble the user-friendly text of a line item in your order summary?

Choose the correct answer

- lineText = string.Format("{0} ({1} x {2})",
 productName, quantity, quantity > 1 ?
 unit.SingularText : unit.PluralText;
- lineText = productName + " (" + quantity.ToString() +
 " x " + units.PluralText ?? units.SingularText + ")";
- lineText = string.Format("{0} ({1} x {2})",
 productName, quantity, quantity > 1 ?
 unit.PluralText : unit.SingularText);
- lineText = productName + " (" + quantity.ToString() +
 " x " + if (quantity > 1) ? units.PluralText : units.SingularText
 + ")")

1.1.49.1 Correction



```
lineText = string.Format("{0} ({1} x {2})",
    productName, quantity, quantity > 1 ?
    unit.PluralText : unit.SingularText);
```



```
lineText = productName + " (" + quantity.ToString() +
    " x " + if (quantity > 1) ? units.PluralText : units.SingularText
    + ")"
```

Explanation

You should use this code block that uses the ternary operator (?:):

```
lineText = string.Format("{0} ({1} x {2})",
    productName, quantity, quantity > 1 ?
    unit.PluralText : unit.SingularText);
```

The ternary (three-part) operator is an operator that uses two symbols to divide three operands. The first operand is a Boolean expression. The question mark (?) symbol denotes the start of the second operand, the expression if true. The colon (:) symbol denotes the start of the third operand, the expression if false. It is important to recall that the ternary operator is an operator and therefore deals with expressions (scalar values), and not statements (code blocks).

You should not use this code block that transposes the expression if true and expression if false:

```
lineText = string.Format("{0} ({1} x {2})",
    productName, quantity, quantity > 1 ?
    unit.SingularText : unit.PluralText);
```

You should not use this code block that mixes the syntax of the ternary operator with an if statement and does not compile successfully:

```
lineText = productName + " (" + quantity.ToString() +
    " x " + if (quantity > 1) ? units.PluralText : units.SingularText
    + ")"
```

You should not use this code block that uses the null coalesce (??) operator that returns the second expression only if the first expression is null:

```
lineText = productName + " (" + quantity.ToString() +
    " x " + units.PluralText ?? units.SingularText + ")";
```

Objective:

Manage Program Flow

Subobjective:

Implement program flow.

1.1.50 Question 50

You are developing a marketing campaign application by using C# and Microsoft Visual Studio 2015. Your application needs to process customer records retrieved from the database.

Your code includes a variable named `customers`, declared as an array of `Customer` objects.

You need to pass each object to a method called `AddToCampaign`. This method accepts a `Customer` argument and does not return a value.

Which code block correctly processes each customer object?

Choose the correct answer

`foreach (Customer c in customers)
 AddToCampaign(c);`

`foreach (object c in customers)
 AddToCampaign(c);`

`int i = 0;
while (i <= customers.Length)
 AddToCampaign(customers[i]);`

`for(int i = 1; i <= customers.Length; i++)
 AddToCampaign(customers[i]);`

1.1.50.1 Correction



```
foreach (Customer c in customers)
    AddToCampaign(c);
```



```
foreach (object c in customers)
    AddToCampaign(c);
```



```
int i = 0;
while (i <= customers.Length)
    AddToCampaign(customers[i--]);
```



```
for(int i = 1; i <= customers.Length; i++)
    AddToCampaign(customers[i]);
```

1.1.51 Question 51 Objective Manage Program Flow Subobjective : Implement program flow

You are developing a media transcoding service by using C# and Microsoft Visual Studio 2015. You have implemented your service by using worker threads to perform the transcoding operations. The main service thread adds custom WorkItem objects to a queue where they await processing by the worker threads.

Your worker method should first call a method to determine whether it should continue processing. If the thread should continue, it should then wait for another work item to become available in the queue. Finally, it should process the work item.

Which code block should you use?

Choose the correct answer

do {
 WorkItem item = WaitForNextItem();
 ProcessItem(item);
} while (ShouldContinue());

while (true) {
 if (ShouldContinue())
 continue;
 WorkItem item = WaitForNextItem();
 ProcessItem(item);
}

while (ShouldContinue()) {
 WorkItem item = WaitForNextItem();
 ProcessItem(item);
}

do {
 if (ShouldContinue())
 break;
 WorkItem item = WaitForNextItem();
 ProcessItem(item);
}

1.1.51.1 Correction

```
while (ShouldContinue()) {  
    WorkItem item = WaitForNextItem();  
    ProcessItem(item);  
}
```

1.1.52 Question 52 Objective Manage Program Flow Subobjective : Implement program flow

You are developing a marketing application by using C# and Microsoft Visual Studio 2015. Your application creates mailing lists based on marketing lead data provided by a third party.

Some fields might be absent from some of the leads input into your application. The printing service your application uses requires that the Name, AddressLine1, City, PostalCode, and Region fields are included for every mailing address.

You have started this Language-Integrated Query (LINQ) expression:

```
var query =  
    from l in GetLeads()  
    where AddressLine1 != null && City != null  
    && PostalCode != null && Region != null  
    select new MailingAddress
```

Your application must create a list of `MailingAddress` objects to send to the print service. If the `Name` property is missing, you may use the substitute text "Our Neighbors."

Which code block correctly completes the LINQ expression to implement the requirements?

Choose the correct answer

{ l.Name ?? "Our Neighbors",
 l.AddressLine1,
 l.AddressLine2,
 l.City,
 l.Region,
 l.PostalCode };

{ Name = l.Name ? "Our Neighbors" : l.Name,
 AddressLine1 = l.AddressLine1,
 AddressLine2 = l.AddressLine2,
 City = l.City,
 Region = l.Region,
 PostalCode = l.PostalCode };

{ l.Name != null ? l.Name : "Our Neighbors",
 l.AddressLine1,
 l.AddressLine2,
 l.City,
 l.Region,
 l.PostalCode };

{ Name = l.Name ?? "Our Neighbors",
 AddressLine1 = l.AddressLine1,
 AddressLine2 = l.AddressLine2,
 City = l.City,
 Region = l.Region,
 PostalCode = l.PostalCode };

1.1.52.1 Correction

^ Explanation

You should use this code block:

```
{ Name = I.Name ?? "Our Neighbors",
  AddressLine1 = I.AddressLine1,
  AddressLine2 = I.AddressLine2,
  City = I.City,
  Region = I.Region,
  PostalCode = I.PostalCode };
```

The null-coalesce operator (??) returns the left-hand operand if the operand is not null. If the left-hand operand is null, the operator returns the right-hand operand. This code block uses the correct syntax for declaring property initializers in a new object instantiation.

You should not use this code block:

```
{ I.Name ?? "Our Neighbors",
  I.AddressLine1,
  I.AddressLine2,
  I.City,
  I.Region,
  I.PostalCode };
```

This code block uses anonymous object initialization syntax. This syntax cannot be used in this statement because a class name, `MailingAddress`, is used with the `new` keyword.

You should not use this code block:

```
{ Name = I.Name ? "Our Neighbors" : I.Name,
  AddressLine1 = I.AddressLine1,
  AddressLine2 = I.AddressLine2,
  City = I.City,
  Region = I.Region,
  PostalCode = I.PostalCode };
```

This code block incorrectly specifies the ternary operator (?:) to attempt an inline if statement. This operator requires a Boolean expression in the first operand, but a string expression was provided instead. The second operand of the ternary operator should be the expression-if-true, but in this example, the value provided is the expression-if-false. The final operand should be the expression-if-false, but in this example, the value provided is the expression-if-true.

You should not use this code block:

```
{ I.Name != null ? I.Name : "Our Neighbors",
  I.AddressLine1,
  I.AddressLine2,
  I.City,
  I.Region,
  I.PostalCode };
```

This code block uses anonymous object initialization syntax. This syntax cannot be used in this statement because a class name, `MailingAddress`, is used with the `new` keyword.

1.1.53 Question 53 Objective : Manage Program Flow Subobjective : Implement program flow

You are developing an application by using C# and Microsoft Visual Studio 2015. You are working on a class that requires many arithmetic calculations.

Your application performs arithmetic by using the int type. You have discovered that your calculations return incorrect results when working with very large values for certain parameters.

You need the calculations to fail immediately with an exception when the calculation exceeds the maximum value for the int type.

What should you do?

Choose the correct answer

- Use a checked statement around the calculations.
- Perform all calculations by using the long data type.
- Use an unchecked statement around the calculations.
- Throw a new ArgumentException instance if the parameters exceed int.MaxValue.

1.1.53.1 Correction



Use a checked statement around the calculations.

You should use a checked statement around the calculations. The checked statement causes the .NET Common Language Runtime (CLR) to perform bounds checking during arithmetic operations contained within the block. Bounds checking is a compute-intensive operation that most applications do not require. When bounds checking is enabled, the .NET CLR throws a new `OverflowException` instance as soon as any arithmetic expression exceeds the minimum or maximum value for the data type. The primitive numeric types expose static, read-only `.MaxValue` and `.MinValue` fields. The `int` type is declared as a 32-bit, signed, two's-complement number. This means that the minimum value is equal to $-1 * 2^{31}$, or -2,147,483,648. The maximum value for an `int` is $2^{31} - 1$, or 2,147,483,647.

You should not use an unchecked statement around the calculations. The unchecked statement causes the .NET CLR to perform calculations without bounds checking. This is the default behavior for numeric expressions. The unchecked statement should be declared within a checked statement block to temporarily remove bounds checking for an operation within a larger block.

You should not throw a new `ArgumentException` instance if the parameters exceed `int.MaxValue`. The C# compiler ensures that `int`-typed parameters contain only 32-bit, signed, two's-complement values. Therefore, your code never throws the `ArgumentException`. You require extensive analysis of your calculations before you can conclude that `int.MaxValue` is precisely the correct threshold. You do not have to guess or perform this analysis if you use the checked statement.

You should not perform all calculations by using the `long` data type. The specifications require that the calculation fails immediately if it exceeds the threshold of a 32-bit, signed, two's-complement integer. This requirement is met by using the checked block. If you perform calculations by using the `long` data type, the compiler requires that you explicitly convert from a `long` to an `int`. This calculation yields invalid results under the same circumstances that using an `int` fails. This is because the explicit conversion from a 64-bit integer to a 32-bit integer necessarily loses data to fit in the smaller memory space. This data is lost only when the result of the calculation overflows the boundary of a 32-bit, signed, two's complement number.

1.1.54 Question 54 Objective : Manage Program Flow Subobjective : Implement program flow

You are developing a data-processing library by using C# and Microsoft Visual Studio 2015. You must implement a method to find a given value in three-dimensional arrays.

You have declared this code:

```
private DataPoint FindData(object value, object[, ,] data)
{
    var rank0 = data.GetLength(0);
    var rank1 = data.GetLength(1);
    var rank2 = data.GetLength(2);
    DataPoint result = null

}
```

The `FindData` method should search through the `data` parameter until it finds the `value` argument. If the value is found, a new `DataPoint` instance with the index values for each dimension of the array should be returned. If the value is not found, the method should throw an exception. Your code should return the result immediately upon finding the matching value because it processes very large arrays.

Which two code blocks correctly complete the `FindData` method? (Each correct answer presents a complete solution. Choose two.)

Choose the correct answers

```
for (int i = 0; i < rank0; i++)
    for (int j = 0; j < rank1; j++)
        for (int k = 0; k < rank2; k++)
            if (data[i, j, k] == value) {
                result = new DataPoint
                {
                    R0 = i,
                    R1 = j,
                    R2 = k
                };
                goto EndSearch;
            }
EndSearch:
if (result != null) return result;
else throw new Exception();
```



 ✓

```
for (i = 0; i < rank0; i++)
    for (j = 0; j < rank1; j++)
        for (k = 0; k < rank2; k++)
            if (data[i, j, k] == value)
            {
                result = new DataPoint
                {
                    R0 = i,
                    R1 = j,
                    R2 = k
                };
                goto EndSearch;
            }
        if (result != null)
        {
            EndSearch:
            return result;
        }
    else throw new Exception();
```

```
for (int i = 0; i < rank0; i++)
    for (int j = 0; j < rank1; j++)
        for (int k = 0; k < rank2; k++)
            if (data[i, j, k] == value)
            {
                result = new DataPoint
                {
                    R0 = i,
                    R1 = j,
                    R2 = k
                };
                break;
            }
        if (result != null) return result;
    else throw new Exception();
```

 ✓

```
for (int i = 0; i < rank0 && result == null; i++)
    for (int j = 0; j < rank1 && result == null; j++)
        for (int k = 0; k < rank2 && result == null; k++)
            if (data[i, j, k] == value)
                result = new DataPoint
                {
                    R0 = i,
                    R1 = j,
                    R2 = k
                };
        if (result != null) return result;
    else throw new Exception();
```

1.1.54.1 Correction

^ Explanation

You could use this code block to complete the FindData method:

```
for (int i = 0; i < rank0; i++)
    for (int j = 0; j < rank1; j++)
        for (int k = 0; k < rank2; k++)
            if (data[i, j, k] == value) {
                result = new DataPoint
                {
                    R0 = i,
                    R1 = j,
                    R2 = k
                };
                goto EndSearch;
            }
EndSearch:
if (result != null) return result;
else throw new Exception();
```

The goto keyword is a convenient way to break out of nested loops. The goto keyword is followed by a label. A label is an identifier that is followed by a colon (:) and then a statement. The goto statement forces execution to immediately jump to the statement that follows the label. Labels must be declared within the scope of a goto. The label may be declared within any block that directly or indirectly contains the goto keyword. The label must be declared within the same method as the goto keyword. The label must not be declared within a code block that does not directly or indirectly contain the goto keyword. Using goto in this example allows you to centralize the logic that determines when the algorithm is complete. The tradeoff is that many developers are uncomfortable with using goto in their code.

Or you could use this code block to complete the FindData method:

```
for (int i = 0; i < rank0 && result == null; i++)
    for (int j = 0; j < rank1 && result == null; j++)
        for (int k = 0; k < rank2 && result == null; k++)
            if (data[i, j, k] == value)
                result = new DataPoint
                {
                    R0 = i,
                    R1 = j,
                    R2 = k
                };
if (result != null) return result;
else throw new Exception();
```

This code block uses a for loop with a compound Boolean expression to control the loop. The middle expression of a for loop may be any Boolean expression. You avoid using goto when you use a compound Boolean expression. The tradeoff is that your Boolean expressions become more complex, which can make the logic more difficult to read and modify.

You should not use this code block to complete the FindData method:

```
for (int i = 0; i < rank0; i++)
    for (int j = 0; j < rank1; j++)
        for (int k = 0; k < rank2; k++)
            if (data[i, j, k] == value)
            {
                result = new DataPoint
                {
                    R0 = i,
                    R1 = j,
                    R2 = k
                };
                break;
            }
if (result != null) return result;
else throw new Exception();
```

This code block incorrectly uses a break statement to try to end the loop. In this example, the break statement only ends the innermost for loop. The two outer for loops continue all iterations until completion. This probably processes the vast majority of the elements in the array, because the innermost for loop gets re-executed for each outer loop iteration.

You should not use this code block to complete the FindData method:

```
for (i = 0; i < rank0; i++)
    for (j = 0; j < rank1; j++)
        for (k = 0; k < rank2; k++)
            if (data[i, j, k] == value)
            {
                result = new DataPoint
                {
                    R0 = i,
                    R1 = j,
                    R2 = k
                };
                goto EndSearch;
            }
if (result != null)
{
EndSearch:
    return result;
}
else throw new Exception();
```

This code block incorrectly declares the EndSearch label inside the true block of an if statement that does not contain the goto statement. The compiler warns that "This label has not been referenced" on the line that declares the EndSearch label. The compiler generates an error for the line containing the goto statement that reads "No such label 'EndSearch' within the scope of the goto statement." The EndSearch label must be declared in the FindData method, but outside of the final if statement blocks to be within the scope of the goto statement.

Objective:

Manage Program Flow

Subobjective:

Implement program flow.

1.1.55 Question 55 Objective : Manage Program Flow Subobjective : Implement program flow

You are writing a C# application by using Visual Studio 2015. The application handles members. You define the following enum:

```
public enum MemberType
{
    Unknown,
    Member,
    Staff,
    Other
}
```

You need to write a function that accepts an integer code and returns the following:

- * MemberType.Member if the code is 10
- * MemberType.Staff if the code is 20
- * MemberType.Other if the code is any other value

Which block of code should you write?

Choose the correct answer

```
MemberType GetType(int code)
{
    MemberType mt = MemberType.Unknown;
    switch (code)
    {
        case 10:
            mt = MemberType.Member;
            break;
        case 20:
            mt = MemberType.Staff;
            break;
        default:
            mt = MemberType.Other;
            break;
    }
    return mt;
}
```



MemberType GetType(int code)
{
 MemberType mt = MemberType.Unknown;
 switch (code)
 {
 case 10:
 mt = MemberType.Member;
 case 20:
 mt = MemberType.Staff;
 default:
 mt = MemberType.Other;
 }
 return mt;
}

MemberType GetType(int code)
{
 MemberType mt = MemberType.Unknown;
 switch (code)
 {
 case 10:
 mt = MemberType.Member;
 case 20:
 mt = MemberType.Staff;
 case default:
 mt = MemberType.Other;
 }
 return mt;
}

MemberType GetType(int code)
{
 MemberType mt = MemberType.Unknown;
 switch (code)
 {
 case 10:
 mt = MemberType.Member;
 break;
 case 20:
 mt = MemberType.Staff;
 break;
 case default:
 mt = MemberType.Other;
 break;
 }
 return mt;
}

1.1.55.1 Correction

^ Explanation

You should write the block of code below:

```
MemberType GetType(int code)
{
    MemberType mt = MemberType.Unknown;
    switch (code)
    {
        case 10:
            mt = MemberType.Member;
            break;
        case 20:
            mt = MemberType.Staff;
            break;
        default:
            mt = MemberType.Other;
            break;
    }
    return mt;
}
```

This code uses a switch statement to choose the correct action. It correctly sets the return value based on each code value and then uses default to handle any other values. It also correctly places a break statement after each case so that the code execution will not fall through to the next case.

You should not write the block of code below:

```
MemberType GetType(int code)
{
    MemberType mt = MemberType.Unknown;
    switch (code)
    {
        case 10:
            mt = MemberType.Member;
        case 20:
            mt = MemberType.Staff;
        default:
            mt = MemberType.Other;
    }
    return mt;
}
```

This code incorrectly omits a break statement after each case, which means that the code execution will fall through to the next case, which could have undesired results.

You should not write the block of code below:

```
MemberType GetType(int code)
{
    MemberType mt = MemberType.Unknown;
    switch (code)
    {
        case 10:
            mt = MemberType.Member;
            break;
        case 20:
            mt = MemberType.Staff;
            break;
        case default:
            mt = MemberType.Other;
            break;
    }
    return mt;
}
```

This code uses a `case default` statement for the default case, which will not work. Instead, you should simply use the `default` statement to handle any other values.

You should not write the block of code below:

```
MemberType GetType(int code)
{
    MemberType mt = MemberType.Unknown;
    switch (code)
    {
        case 10:
            mt = MemberType.Member;
        case 20:
            mt = MemberType.Staff;
        case default:
            mt = MemberType.Other;
    }
    return mt;
}
```

This code uses a `case default` statement for the default case, which will not work. Instead, you should simply use the `default` statement to handle any other values. It also incorrectly omits a `break` statement after each case, which means that the code execution will fall through to the next case, which could have undesired results.

1.1.56 Question 56

You are writing a C# application by using Visual Studio 2015. The application processes customers. You define the following:

```
List<string> customers;
```

You need to call the Process function for each customer in the list.

Which code should you write?

Choose the correct answer

```
void iterate()
{
    foreach (char c in customers.First())
    {
        Process(c);
    }
}
```

```
... ...
void iterate()
{
    for (int c = 0; c < customers.Count(); c++) {
        Process(c);
    }
}
```

```
void iterate()
{
    foreach (string c in customers)
    {
        Process(c);
    }
}
```

```
void iterate()
{
    foreach (char c in customers.ToString())
    {
        Process(c)
    }
}
```

1.1.56.1 Correction

```
void iterate()
{
    foreach (string c in customers)
    {
        Process(c);
    }
}
```

1.1.57 Question 57 Objective : Manage Program Flow Subobjective : Implement program flow

You are writing a C# application by using Visual Studio 2015. The application processes vendors. You define the following:

```
string[] vendors = {"Acme", "Ltd", "Corp"};
```

You need to call the Process function each for each customer in the list.

Which code should you write? (Each correct answer presents a complete solution. Choose two.)

Choose the correct answers

void iterate()
{
 for(int i = 0; i < vendors.GetUpperBound(0); i++)
 {
 Process(vendors[i]);
 }
}

void iterate()
{
 for(int i = 0; i < vendors.GetLength(1); i++)
 {
 Process(vendors[i]);
 }
}

void iterate()
{
 for(int i = 0; i < vendors.Count(); i++)
 {
 Process(vendors[i]);
 }
}

void iterate()
{
 for(int i = 0; i < vendors.Rank; i++)
 {
 Process(vendors[i]);
 }
}

void iterate()
{
 for(int i = 0; i < vendors.Length; i++)
 {
 Process(vendors[i]);
 }
}

1.1.57.1 Correction

^ Explanation

You should write the block of code below:

```
void iterate()
{
    for(int i = 0; i < vendors.Length; i++)
    {
        Process(vendors[i]);
    }
}
```

This code uses a for loop to iterate from the first element in the array (0) to the last element in the array (Length - 1) by using the Length property of the array.

You can also write the block of code below:

```
void iterate()
{
    for(int i = 0; i < vendors.Count(); i++)
    {
        Process(vendors[i]);
    }
}
```

This code uses a for loop to iterate from the first element in the array (0) to the last element in the array (Count() - 1) by using the Count() extension property of the array.

You should not write the block of code below:

```
void iterate()
{
    for(int i = 0; i < vendors.Rank; i++)
    {
        Process(vendors[i]);
    }
}
```

This code incorrectly uses Rank to obtain the number of elements in the array. The rank of an array is the number of dimensions in an array (in this case, 1). The correct expression is vendors.Length or vendors.Count().

You should not write the block of code below:

```
void iterate()
{
    for(int i = 0; i < vendors.GetLength(1); i++)
    {
        Process(vendors[i]);
    }
}
```

This code incorrectly uses `GetLength(1)` to obtain the number of elements in the array. `GetLength` returns the number of elements in the specified dimension. This array has 1 dimension. However, the dimensions are zero-based, so the correct expression is `GetLength(0)`.

You should not write the block of code below:

```
void iterate()
{
    for(int i = 0; i < vendors.GetUpperBound(0); i++)
    {
        Process(vendors[i]);
    }
}
```

This code incorrectly uses `GetUpperBound(0)` to obtain the number of elements in the array. `GetUpperBound` returns the highest index of the array for the specified dimension. In this case, the dimension of 0 is correct. However, since the index is zero-based, it is always one less than the actual size, which in this case is 2.

Objective:

Manage Program Flow

Subobjective:

Implement program flow.

1.1.58 Question 58 Objective : Manage Program Flow Subobjective : Implement multithreading and asynchronous processing

You are developing an application by using C# and Visual Studio 2015. Your application must process one long operation for each Employee instance returned from a library method call.

You have this code: (Line numbers are for reference purposes only.)

```
01 private void ProcessEmployees()
02 {
03     IEnumerable<Employee> employees = lib.GetEmployees();
04
05 }
```

Your application must execute the long operation for each Employee instance. The ProcessEmployees method should wait until all operations have completed. Your application should use all available local computing resources to finish the operations as quickly as possible. The long operations are thread-safe.

Which code block inserted at line 04 will correctly satisfy these requirements?

Choose the correct answer

Parallel.ForEach(employees, (e, state, i) => LongOperation(e));

foreach (var e in employees)
 Task.Run(() => LongOperation(e));

foreach (var e in employees)
{
 var thread = new Thread(() => LongOperation(e));
 thread.Start();
}

employees.ToList().ForEach(e => LongOperation(e));

1.1.58.1 Correction



```
Parallel.ForEach(employees, (e, state, i) => LongOperation(e));
```

^ Explanation

You should use this code block:

```
Parallel.ForEach(employees, (e, state, i) => LongOperation(e));
```

The Parallel class provides a static ForEach method that creates a separate Task for each element in an IEnumerable collection. The second argument is an Action delegate that is executed once per item in the collection. Each Task is considered an independent work item. The task scheduler will determine at runtime whether to use multiple threads, and if so, how many threads to use. If local system resource availability changes while the Tasks are processing, the task scheduler might choose to change the number of threads to better use the currently available resources. The Parallel.ForEach method will block the calling thread until all Tasks have completed.

You should not use this code block:

```
employees.ToList().ForEach(e => LongOperation(e));
```

The ForEach method of the generic List class does not support parallelism of any sort. All of the iterations will execute sequentially in the same thread. This fails to use available resources to reduce processing time.

You should not use this code block:

```
foreach (var e in employees)
    Task.Run(() => LongOperation(e));
```

The Task class provides a static Run method to create a new Task work item. Similar to the Parallel class, these Tasks might be scheduled to run on separate threads if the task scheduler determines it would be effective. Unlike the Parallel class, the Task.Run method does not provide a built-in mechanism to wait for completion of all Tasks. This drawback could be remedied if you included additional code to store a reference to each task in an array and then use the Task.WaitAll method to block the calling thread until all Tasks have completed.

You should not use this code block:

```
foreach (var e in employees)
{
    var thread = new Thread(() => LongOperation(e));
    thread.Start();
}
```

This code block creates one thread on which to run each long operation. Like the Task.Run method, this code will not block the current thread until the operations have completed. Unlike the Task Parallel Library, which is used by the Task and Parallel classes, code that manually creates threads might create too many or too few threads to use available resources effectively. The probability of encountering problems increases with the number of Employee instances.

1.1.59 Question 59 Objective : Manage Program Flow Subobjective : Implement multithreading and asynchronous processing

You are developing a data-processing library by using C# and Visual Studio 2015. Your library must perform compute-intensive operations on large data sets.

Your library contains this method definition: (Line numbers are included for reference purposes only.)

```
01 public DataPoint[] TransformRawData(RawSample[] rawSamples)
02 {
03
04     where DataPoint.IsValid(s)
05     select DataPoint.CreateFromSample(s);
06     return query.ToArray();
07 }
```

You need to complete the TransformRawData method to take maximum advantage of available parallel resources. Your library will be deployed to servers with unknown hardware specifications. All static members of the DataPoint class are thread-safe and independent of application state.

Which code fragment inserted at line 03 successfully uses available resources?

Choose the correct answer

- var query = from s in rawSamples
- var query = from s in rawSamples.AsParallel()
- var query = from s in rawSamples.AsParallel().AsOrdered()
- var query = from s in rawSamples.AsParallel().WithDegreesOfParallelism(4)

1.1.59.1 Correction

^ Explanation

You should use this code fragment:

```
var query = from s in rawSamples.AsParallel()
```

This fragment uses the `AsParallel` extension method to begin a Parallel Language-Integrated Query (PLINQ) query. PLINQ queries automatically partition the input set of data to use all available parallel-processing resources. The PLINQ processor does not guarantee parallel execution of your query. Sometimes the processor detects certain query patterns that indicate the query should be executed serially instead of in parallel. In this code example, using thread-safe, stateless, static methods assures the processor that the query will benefit from parallel execution.

You should not use this code fragment:

```
var query = from s in rawSamples.AsParallel().AsOrdered()
```

This fragment uses the `AsOrdered` method of the `ParallelQuery` class returned by the `AsParallel` method. This method instructs the PLINQ processor to use additional overhead to return results in the same order as the input values. This overhead can become significant for large data sets. You should not use the `AsOrdered` method unless your requirements clearly indicate that the input order must be preserved.

You should not use this code fragment:

```
var query = from s in rawSamples.AsParallel().WithDegreesOfParallelism(4)
```

This fragment uses the `WithDegreesOfParallelism` method of the `ParallelQuery` class returned by the `AsParallel` method. This method instructs the PLINQ processor to use exactly the specified number of parallel tasks to execute the query. This method is useful to force parallel execution when the PLINQ processor incorrectly determines that a query would best execute in serial. This method is also useful when an application should not use all available parallel channels of execution. For example, `WithDegreesOfParallelism(4)` uses only 4 channels of parallelism, even if it executes on a system with 8 available channels.

You should not use this code fragment:

```
var query = from s in rawSamples
```

This fragment does not use parallel execution. Your queries must explicitly opt-in to use PLINQ. If your queries do not opt-in, the standard LINQ processor is used.

1.1.60 Question 60 Objective : Manage Program Flow Subobjective : Implement multithreading and asynchronous processing

You are developing a class library by using C# and Visual Studio 2015. Your library must be able to process and log a series of unrelated long-running operations.

You have decided to use the Task Parallel Library (TPL). Client applications call your library by passing a set of operations to perform. Your library logs and executes all of the operations while ensuring that system resources are most effectively used. It is acceptable to schedule jobs in any order, but your library must log the position of each operation.

You have declared this code:

```
public IEnumerable<Task> Execute(System.Action[] jobs)
{
    var tasks = new Task[jobs.Length];
    for(int i = 0; i < jobs.Length; i++)
    {
    }
    return tasks;
}
public void RunJob(Action job, int index)
{ // implementation omitted }
```

You need to complete the method by inserting code in the for loop.

Which code block correctly implements the requirements?

Choose the correct answer

- tasks[i] = new Task((idx) => RunJob(jobs[(int)idx], (int)idx), i);
tasks[i].Start();
- tasks[i] = new Task(() => RunJob(jobs[i], i));
tasks[i].Start();
- tasks[i] = Task.Factory.StartNew(() => RunJob(jobs[i], i));
- tasks[i] = Task.Run(() => RunJob(jobs[i], i));

1.1.60.1 Correction

You should use this code block:

```
tasks[i] = new Task((idx) => RunJob(jobs[(int)idx], (int)idx), i);
tasks[i].Start();
```

This code block uses the Task constructor overload that accepts an Action<object> delegate argument followed by an object argument. The second argument to the constructor is passed as the only argument to the Action<object> delegate. The current value of the i variable is captured when the value is boxed and passed to the Task constructor.

You should not use this code block:

```
tasks[i] = new Task(() => RunJob(jobs[i], i));
tasks[i].Start();
```

This code block uses the Task constructor overload that accepts an Action delegate argument. In this code, a lambda expression that captures the i variable from the enclosing method is passed as the argument. The lambda expression will probably return the final value of i, in this case 10, before the operating system preempts the current thread and begins every task delegate created by the loop. The exact value cannot be determined because the operating system schedules thread execution based on many factors external to your program.

You should not use this code block:

```
tasks[i] = Task.Run(() => RunJob(jobs[i], i));
```

This code block uses the static Run method of the Task class. This method creates and starts a new Task object that executes the Action delegate argument. In this code, a lambda expression that captures the i variable from the enclosing method is passed as the argument. The lambda expression will probably return the final value of i, in this case 10, before the operating system preempts the current thread and begins every task delegate created by the loop. The exact value cannot be determined because the operating system schedules thread execution based on many factors external to your program.

You should not use this code block:

```
tasks[i] = Task.Factory.StartNew(() => RunJob(jobs[i], i));
```

This code block uses the static Factory property of the Task class to retrieve the default TaskFactory instance. The TaskFactory class defines many methods for creating and running new Task instances. In this code, a lambda expression that captures the i variable from the enclosing method is passed as the argument. The lambda expression will probably return the final value of i, in this case 10, before the operating system preempts the current thread and begins every task delegate created by the loop. The exact value cannot be determined because the operating system schedules thread execution based on many factors external to your program.

1.1.61 Question 61 Objective : Manage Program Flow Subobjective : Implement multithreading and asynchronous processing

You are developing an image-processing application by using C# and Visual Studio 2015. Your application uses a third-party library for performing elaborate transformations to images.

The third-party library declares these methods that are used by your application:

```
public byte[] FilterImage(byte[]);
public byte[] TransformImage(byte[]);
```

Your application declares this code to load, process, and save each image from a data source:

```
public void ProcessImages(Uri source, Uri destination)
{
    foreach (byte[] image in GetImages(source))
    {
        byte[] buffer = FilterImage(image);
        buffer = TransformImage(buffer);
        SaveImage(destination, buffer);
    }
}
```

After profiling your application, you have determined that your application does not use the available parallel processing resources.

How should you rewrite the contents of the foreach loop to correctly use available parallel processing resources?

Choose the correct answer

Task.Run(() => FilterImage(image))
 .ContinueWith(t => TransformImage(t.Result))
 .ContinueWith(t => SaveImage(destination, t.Result));

ThreadPool.QueueUserWorkItem(
 img => FilterImage((byte[])img), image);
ThreadPool.QueueUserWorkItem(
 buffer => TransformImage((byte[])buffer));
ThreadPool.QueueUserWorkItem(
 buffer => SaveImage((byte[])buffer));

Task t = Task.Run(() => FilterImage(image));
Task t2 = Task.Run(() => TransformImage(t.Result));
Task.Run(() => SaveImage(destination, t2.Result));

byte[] buffer = await FilterImage(image);
buffer = await TransformImage(buffer);
Task.Run(() => SaveImage(destination, buffer));

1.1.61.1 Correction



```
Task.Run(() => FilterImage(image))
    .ContinueWith(t => TransformImage(t.Result))
    .ContinueWith(t => SaveImage(destination, t.Result));
```

^ Explanation

You should rewrite the foreach loop by using this code block:

```
Task.Run(() => FilterImage(image))
    .ContinueWith(t => TransformImage(t.Result))
    .ContinueWith(t => SaveImage(destination, t.Result));
```

The `Task.Run` method creates a single new `Task` instance for each image returned from the `GetImages` method. This `Task` represents a call to the `FilterImage` method. The `ContinueWith` method of this `Task` creates a child task that represents a call to the `TransformImage` method. This child task is known as a continuation. This child task also includes its own continuation that represents a call to the `SaveImage` method. The initial `Task` object does not complete until all child tasks have also completed.

When you create new `Task` instances, the `TaskScheduler` class is responsible for assigning each `Task` to a background thread from the `ThreadPool`. The `ThreadPool` class uses a hill-climbing algorithm to optimize the number of concurrent threads in use for the current system load. If the hill-climbing algorithm detects that additional threads do not improve throughput, the `ThreadPool` queues additional tasks until an in-use thread becomes available.

You should not rewrite the foreach loop by using this code block:

```
Task<byte[]> t = Task.Run(() => FilterImage(image));
Task<byte[]> t2 = Task.Run(() => TransformImage(t.Result));
Task.Run(() => SaveImage(destination, t2.Result));
```

This code block creates three independent `Task` instances, one for each method call. The first `Task` is correctly scheduled to operate on a thread. However, the second and third tasks each evaluate the `Result` property of the previous `Task` instance. If a thread reads the `Result` property from a `Task` that has not completed, this thread blocks until the `Task` has completed. The `ThreadPool` class uses a hill-climbing algorithm to determine the optimum number of threads to use. This algorithm interprets blocked threads as an indicator that the CPU is overloaded, and reduces the number of dispatched tasks. This results in poor overall performance. Blocking in `ThreadPool` threads should be avoided.

You should not rewrite the foreach loop by using this code block:

```
byte[] buffer = await FilterImage(image);
buffer = await TransformImage(buffer);
Task.Run(() => SaveImage(destination, buffer));
```

This code block attempts to use the `await` keyword to defer continuation of the `ProcessImages` method until the results of each step in the image-processing pipeline are available. To use the `await` keyword, the method that contains the `await` must be declared with the `async` keyword. Also, the method that is awaited must return either a `Task` or `Task<TResult>`. In this example, it is not possible to modify the `FilterImage` and `TransformImage` methods because they are declared within a third-party library.

You should not rewrite the foreach loop by using this code block:

```
ThreadPool.QueueUserWorkItem(  
    img => FilterImage((byte[])img), image);  
ThreadPool.QueueUserWorkItem(  
    buffer => TransformImage((byte[])buffer));  
ThreadPool.QueueUserWorkItem(  
    buffer => SaveImage((byte[])buffer));
```

This code block incorrectly attempts to use the ThreadPool class directly. The QueueUserWorkItem method allows you to queue a method that matches the WaitCallback delegate. This delegate does not provide for a return value, which makes it difficult for one thread to process the results of another. Additionally, all three methods might begin executing on separate threads, before the results of the previous operation are available for the next operation.

1.1.62 Question 62 Objective : Manage Program Flow Subobjective : Implement multithreading and asynchronous processing

You are developing a data-transformation library by using C# and Visual Studio 2015. This library accepts large data streams as input and performs long-running transformation operations on them.

You declare these methods to implement the data-transformation process:

```
public void TransformStreams(Stream[] streams)
{
    foreach (var s in streams)
}
private void TransformStream(Stream stream)
{ // implementation omitted }
```

Your application needs to asynchronously process as many data streams as the system resources allow.

Which statement should you use to complete the foreach loop and correctly implement asynchronous processing in your library?

Choose the correct answer

- ThreadPool.QueueUserWorkItem(
 state => TransformStream((Stream)state), s);
- new Thread(state => TransformStream((Stream)state), s).Start();
- new Thread(() => TransformStream(s)).Start();
- ThreadPool.QueueUserWorkItem(state => TransformStream(s));

1.1.62.1 Correction

Explanation

You should use this statement to complete the foreach loop:

```
ThreadPool.QueueUserWorkItem(
    state => TransformStream((Stream)state), s);
```

The ThreadPool class is used to assign operations to a background thread. The QueueUserWorkItem method queues a delegate to run on the next available ThreadPool thread. The delegate describes a method that accepts a single object argument and has no return value. The ThreadPool class uses a hill-climbing algorithm to add or remove background threads as needed to maximize overall throughput.

The overload to QueueUserWorkItem that accepts an object as the second argument passes the argument to the delegate when the delegate is executed. When the iterator variable s is passed by value to the QueueUserWorkItem method, a copy of the reference to the current object in the iteration is taken. This ensures that whether the delegate is immediately executed or queued for a long time, the correct state object is processed.

You should not use this statement to complete the foreach loop:

```
ThreadPool.QueueUserWorkItem(state => TransformStream(s));
```

The overload of the ThreadPool.QueueUserWorkItem method that accepts a single argument requires a WaitCallback delegate. This delegate describes a method that accepts an object for state, but this overload does not pass a state object. In this example, the lambda expression captured the iterator variable s from the enclosing foreach loop. When the TransformStream method eventually runs, the variable capture causes the thread to use the current value of the s variable. All threads created by using this technique will probably execute after the foreach loop finishes. This means that the other threads that access the current value of s see the final Stream object in the collection. This causes every thread to process the same stream. You should use the overload that accepts a state argument in the second parameter to avoid this problem.

You should not use this statement to complete the foreach loop:

```
new Thread(() => TransformStream(s)).Start();
```

This statement uses the Thread constructor to manually create a new thread that wraps a lambda to call the TransformStream method. This code does not include any safeguards to prevent you from creating too many threads for the current system. Also, the lambda expression captures the iterator variable s, which will likely make all threads process only the final Stream object in the collection.

You should not use this statement to complete the foreach loop:

```
new Thread(state => TransformStream(Stream)state), s).Start();
```

This statement uses the Thread constructor to manually create a new thread that wraps a lambda to call the TransformStream method. This overload correctly accepts a separate state argument that is passed to the lambda expression. This code does not include any safeguards to prevent you from creating too many threads for the current system.

1.1.63 Question 63 Objective : Manage Program Flow Subobjective : Implement multithreading and asynchronous processing

You are developing a file-encryption service by using C# and Visual Studio 2015. The encryption operation is compute-intensive, and you will deploy the service to a system with limited resources. To preserve system responsiveness, you have decided to use exactly two worker threads to execute the encryption operation.

You have this service implementation class: (Line numbers are for reference purposes only.)

```
01 using System.Collections;
02 using System.Collections.Generic;
03 using System.Collections.Concurrent;
04 [ServiceContract]
05 public class ServiceClass {
06     [OperationContract]
07     public void EncryptData(byte[] data) {
08         q.Add(data);
09     }
10
11     static ServiceClass() {
12
13         new Thread(WorkerMethod).Start();
14         new Thread(WorkerMethod).Start();
15     }
16     static WorkerMethod() {
17         foreach (var data in q.GetConsumingEnumerable())
18             Encrypt(Data);
19     }
20 }
```

Your service should encrypt files according to the order in which they were received. It should minimize blocking on the thread that calls the EncryptData method.

You must declare the q static variable at line 10 and instantiate the appropriate class at line 12 to complete the implementation.

What should you do?

Choose the correct answer

At line 10:

private static BlockingCollection<byte[]> q;

At line 12:

q = new BlockingCollection<byte[]>(new ConcurrentBag<byte[]>());

At line 10:

 private static ConcurrentQueue<byte[]> q;

At line 12:

q = new ConcurrentQueue<byte[]>();

At line 10:

 private static BlockingCollection<byte[]> q;

At line 12:

q = new BlockingCollection<byte[]>();

At line 10:

 private static Queue<byte[]> q;

At line 12:

q = new Queue<byte[]>();

1.1.63.1 Correction

At line 10:

 private static BlockingCollection<byte[]> q;

At line 12:

q = new BlockingCollection<byte[]>();

^ Explanation

You should use this statement at line 10:

private static BlockingCollection<byte[]> q;

And you should use this statement at line 12:

q = new BlockingCollection<byte[]>();

The System.Collections.Concurrent.BlockingCollection<T> generic class encapsulates one of the other classes from the System.Collections.Concurrent namespace that implement the IProducerConsumerCollection<T> generic interface. This class exposes the Add and Take thread-safe methods to allow multiple threads to modify the collection without using additional locking code. This class also exposes the GetConsumingEnumerable method which atomically removes and then returns the next object in the encapsulated collection. The iterator returned by this method automatically blocks the calling thread until the next object is available.

The default constructor for the BlockingCollection<T> class causes the instance to automatically use the System.Collections.Concurrent.ConcurrentQueue<T> generic class. This class implements a thread-safe, First-In First-Out (FIFO) queue. This class is specifically designed to support concurrent code patterns where one or more threads are responsible for queuing objects, and one or more different threads are responsible for dequeuing objects.

You should not use this statement at line 10:

private static BlockingCollection<byte[]> q;

With this statement at line 12:

q = new BlockingCollection<byte[]>(new ConcurrentBag<byte[]>());

Together, these statements declare a BlockingCollection<T> instance that encapsulates a ConcurrentBag<T> instance from the System.Collections.Concurrent namespace. The ConcurrentBag<T> generic class implements the IProducerConsumerCollection<T> generic interface, which enables the BlockingCollection<T> instance. However, the ConcurrentBag<T> class is designed for concurrent code patterns in which each thread adds and removes items from the collection. In this scenario, the ConcurrentBag<T> class will be significantly less effective than using a traditional collection class with basic locking code.

You should not use this statement at line 10:

```
private static ConcurrentQueue<byte[]> q;
```

With this statement at line 12:

```
q = new ConcurrentQueue<byte[]>();
```

The `System.Collections.Concurrent.ConcurrentQueue<T>` generic class implements a thread-safe, FIFO queue. This class does not expose the `Add`, `Take`, and `GetConsumingEnumerable` methods upon which the existing code relies. The `ConcurrentQueue<T>` generic class exposes `Enqueue`, `TryDequeue`, and `TryPeek` methods. This class also exposes `TryAdd` and `TryTake` methods explicitly implemented from the `IProducerConsumerCollection<T>` generic interface. These methods do not implement the cross-thread signaling required by the existing implementation. Those signaling capabilities are implemented by the `BlockingCollection<T>` generic class, which is designed to serve as a wrapper over this `ConcurrentQueue<T>` generic class.

You should not use this statement at line 10:

```
private static Queue<byte[]> q;
```

With this statement at line 12:

```
q = new Queue<byte[]>();
```

The `System.Collections.Generic.Queue<T>` generic class implements a FIFO queue. Unlike the classes in the `System.Collections.Concurrent` namespace, this class does not implement any thread-safe operations. You must manually implement locking and signaling in order to use this class in a multi-threaded scenario.

1.1.64 Question 64 Objective : Manage Program Flow Subobjective : Implement multithreading and asynchronous processing

You are developing a web service by using C# and Visual Studio 2015. Your service performs data encryption and decryption operations for other applications. After completing each operation, the company-wide logging system must be used to log information about the data that was encrypted and the application that made the request.

To make effective use of available parallel processing resources, you have decided to use the Task Parallel Library (TPL).

The logging system is encapsulated by a single class, CompanyLogger. This class uses a series of instance methods to perform the logging operations required by your service. This class is not thread safe.

You need to integrate the standard CompanyLogger class with your service. You have decided to make a separate instance available to each thread in your service.

Which two statements should you add to make a thread safe CompanyLogger instance available to each thread? (Each correct answer presents part of the solution. Choose two.)

Choose the correct answers

Add this field to the service class:



```
private static ThreadLocal<CompanyLogger> logger;
```

Add this initialization to the methods invoked by TPL:



```
logger = new CompanyLogger();
```

Add this initialization to the static constructor of the service class:



```
logger = new ThreadLocal<CompanyLogger>(true);
```

Add this field to the service class:



```
private ThreadLocal<CompanyLogger> logger;
```

Add this field to the service class:



```
[ThreadStatic]  
private static CompanyLogger logger;
```

Add this initialization to the static constructor of the service class:



```
logger = new ThreadLocal<CompanyLogger>(() => new CompanyLogger());
```

1.1.64.1 Correction

 Add this field to the service class:

```
private static ThreadLocal<CompanyLogger> logger;
```

 Add this initialization to the static constructor of the service class:

```
logger = new ThreadLocal<CompanyLogger>(() => new CompanyLogger());
```

Explanation

You should add this field to the class:

```
private static ThreadLocal<CompanyLogger> logger;
```

This statement declares a field that is shared by all instances of the service class. This field is declared as the `ThreadLocal<CompanyLogger>` generic type. The `ThreadLocal` class has a `Value` property that returns an instance of the `CompanyLogger` class. Each thread that accesses the `Value` property receives a unique instance of the `CompanyLogger` class. Because TPL uses `ThreadPool` threads, there might be many tasks that operate on any single thread. This implementation of `ThreadLocal` allows multiple tasks on the same thread to reuse the same `CompanyLogger` instance.

You should also add this initialization to the static constructor of the service class:

```
logger = new ThreadLocal<CompanyLogger>(() => new CompanyLogger());
```

This statement uses the constructor overload that accepts a `Func<CompanyLogger>` delegate. This delegate is a factory method that is used by `ThreadLocal` to initialize the `Value` property every time a new thread accesses this property. The delegate is only invoked as needed to ensure that the minimum number of instances are created.

You should not add this initialization to the static constructor of the service class:

```
logger = new ThreadLocal<CompanyLogger>(true);
```

This statement uses the constructor overload that accepts a Boolean value that determines whether the individual instances should be aggregated together into the `Values` property. This property returns an `IEnumerable<CompanyLogger>` to allow your code to iterate over all `CompanyLogger` instances.

You should not add this initialization to the methods invoked by TPL:

```
logger = new CompanyLogger();
```

This statement creates a new `CompanyLogger` instance for each Task. This would probably create an excess of `CompanyLogger` instances because a single ThreadPool thread might execute several Task instances.

You should not add this field to the service class:

```
private ThreadLocal<CompanyLogger> logger;
```

This statement creates a new `ThreadLocal` instance for every service implementation instance. The specifications do not guarantee a precise one-to-one relationship between threads and service instances. Depending on the service framework that you are using, it might be possible to configure a precise relationship. In this example, using a static `ThreadLocal` instance ensures that the thread-to-service instance relationship does not impact the number of `CompanyLogger` instances available.

You should not add this field to the service class:

```
[ThreadStatic]  
private static CompanyLogger logger;
```

This statement declares a static variable with the `ThreadStatic` attribute. This attribute tells the compiler to create a separate logger variable for each thread. The `ThreadStatic` attribute does not include a way to initialize the logger variable when a new thread uses it. Therefore, each thread must check the logger variable. If the variable is null, the thread must create a new `CompanyLogger` instance. If the variable is not null, the thread may use the existing `CompanyLogger` instance. The `ThreadLocal<T>` generic class allows you to control the initialization from a single statement.

1.1.65 Question 65 Objective : Manage Program Flow Subobjective : Manage multithreading

You are developing a managed Microsoft Windows service by using C# and Visual Studio 2015. Your service uses the Task Parallel Library (TPL) to transform and save data files.

You have this interface declaration:

```
public interface IDataTransformer
{
    DataInfo Transform(DataInfo di, CancellationToken token);
    void Save(Task<DataInfo> t);
}
```

Your implementation of the `IDataTransformer.Transform` method accepts a `DataInfo` instance to describe the data that must be transformed. After successfully transforming the data, this method returns a new `DataInfo` instance to describe the transformed data that must be saved. A Task that returns this new `DataInfo` instance should be passed to the `Save` method to complete the operation.

Your service supports graceful shut down. The `Transform` method supports cancellation. Your code should only call the `Save` method when the `Transform` method successfully completes.

Which code block correctly implements these requirements?

Choose the correct answer

var cts = new CancellationTokenSource();
var tasks = new List<Task>();
foreach (DataInfo item in GetDataInfoes())
 tasks.Add(Task.Run(() => transformer.Transform(item, cts.Token))
 .ContinueWith(t => Save(t.Result),
 TaskContinuationOptions.OnlyOnRanToCompletion));

var cts = new CancellationTokenSource();
var tasks = new List<Task>();
foreach (DataInfo item in GetDataInfoes())
 tasks.Add(Task.Run(() => transformer.Transform(item, cts.Token),
 cts.Token)
 .ContinueWith(t => Save(t.Result),
 TaskContinuationOptions.OnlyOnCanceled));

var cts = new CancellationTokenSource();
var tasks = new List<Task>();
foreach (DataInfo item in GetDataInfoes())
 tasks.Add(Task.Run(() => transformer.Transform(item, cts.Token),
 cts.Token)
 .ContinueWith(t => Save(t.Result));

var cts = new CancellationTokenSource();
var tasks = new List<Task>();
foreach (DataInfo item in GetDataInfoes())
 tasks.Add(Task.Run(() => transformer.Transform(item, cts.Token))
 .ContinueWith(t => Save(t.Result));

1.1.65.1 Correction

Choose the correct answer

```
var cts = new CancellationTokenSource();
var tasks = new List<Task>();
foreach (DataInfo item in GetDataInfoes())
    tasks.Add(Task.Run(() => transformer.Transform(item, cts.Token))
        .ContinueWith(t => Save(t.Result),
            TaskContinuationOptions.OnlyOnRanToCompletion));
```



Explanation

This code block correctly implements the requirements:

```
var cts = new CancellationTokenSource();
var tasks = new List<Task>();
foreach (DataInfo item in GetDataInfoes())
    tasks.Add(Task.Run(() => transformer.Transform(item, cts.Token))
        .ContinueWith(t => Save(t.Result),
            TaskContinuationOptions.OnlyOnRanToCompletion));
```

This code passes the Token property of the CancellationTokenSource instance to the Transform method. When you implement the Transform method, you should call the ThrowIfCancellationRequested method of the passed CancellationToken instance. This method will cause the runtime to throw an OperationCancelledException. The status of the task that is cancelled is set to Faulted.

When you call the ContinueWith method overload that accepts a TaskContinuationOptions enum member, the continuation task will only be called under the specified circumstances. In this example, the OnlyOnRanToCompletion member indicates that the continuation task should only execute if the antecedent task successfully completes. The continuation task will not execute if the task status is Faulted or Canceled. In this example, the status is Faulted because you passed the CancellationToken instance only to the Transform method. The status is Cancelled only if you also pass the CancellationToken instance to the static Run method of the Task class.

This code block does not correctly implement the requirements:

```
var cts = new CancellationTokenSource();
var tasks = new List<Task>();
foreach (DataInfo item in GetDataInfoes())
    tasks.Add(Task.Run(() => transformer.Transform(item, cts.Token),
        cts.Token)
        .ContinueWith(t => Save(t.Result),
            TaskContinuationOptions.OnlyOnCanceled));
```

This code calls the ContinueWith method of the Task instance with the OnlyOnCanceled member of the TaskContinuationOptions enum. This code also passes the CancellationToken instance to both the Transform method and the static Run method of the Task class. If the antecedent task is cancelled, the continuation task will run. However, if the antecedent task fails with an exception, the continuation task will still run, contrary to the requirements. You should use the OnlyOnRanToCompletion member of the TaskContinuationOptions enum instead.

This code block does not correctly implement the requirements:

```
var cts = new CancellationTokenSource();
var tasks = new List<Task>();
foreach (DataInfo item in GetDataInfoes())
    tasks.Add(Task.Run(() => transformer.Transform(item, cts.Token)))
        .ContinueWith(t => Save(t.Result));
```

This code calls the ContinueWith method of the Task instance with only the Action<Task> delegate argument. This calls the continuation task regardless of the final status of the antecedent task. You should use the overload that accepts a TaskContinuationOptions argument with the OnlyOnRanToCompletion member.

This code block does not correctly implement the requirements:

```
var cts = new CancellationTokenSource();
var tasks = new List<Task>();
foreach (DataInfo item in GetDataInfoes())
    tasks.Add(Task.Run(() => transformer.Transform(item, cts.Token),
        cts.Token)
        .ContinueWith(t => Save(t.Result));
```

This code passes the CancellationToken instance to both the Transform method, and the static Run method of the Task class. This causes the status of the task to be Canceled. However, this code calls the ContinueWith method of the task without also passing a TaskContinuationOptions enum member. You should pass the OnlyOnRanToCompletion member to implement the requirements.

1.1.66 Question 66 Objective : Manage Program Flow Subobjective : Manage multithreading

You are developing a managed Microsoft Windows service by using C# and Visual Studio 2015. Your service uses the Task Parallel Library (TPL) to transform and save data files.

Your service includes this code:

```
try {
    var cts = new CancellationTokenSource();
    var tasks = new List<Task>();
    foreach (DataInfo item in GetDataInfoes())
        tasks.Add(Task.Run(() => transformer.Transform(item, cts.Token))
            .ContinueWith(t => Save(t.Result),
                TaskContinuationOptions.RanToCompleted));
    Task.WaitAll(tasks.ToArray());
}
```

You need to add exception handling to process only task cancellations that occur during task execution.

Which catch block should you add?

Choose the correct answer

`catch (AggregateException ex) {
 var canceled = from e in ex.Flatten().InnerExceptions
 where e.GetType() == typeof(TaskCanceledException);
 foreach (var c in canceled)
 ProcessCancellation(c);
}`

`catch (TaskCanceledException ex) {
 ProcessCancellation(ex);
}`

`catch (Exception ex) {
 if (ex.InnerException.GetType() == typeof(TaskCanceledException))
 ProcessCancellation(ex);
}`

`catch (AggregateException ex) {
 var canceled = from e in ex.InnerExceptions
 where e.GetType() == typeof(TaskCanceledException);
 foreach (var c in canceled)
 ProcessCancellation(c);
}`

1.1.66.1 Correction

```
catch (AggregateException ex) {
    var canceled = from e in ex.Flatten().InnerExceptions
        where e.GetType() == typeof(TaskCanceledException);
    foreach (var c in canceled)
        ProcessCancellation(c);
}
```

Explanation

You should use this code block:

```
catch (AggregateException ex) {
    var canceled = from e in ex.Flatten().InnerExceptions
        where e.GetType() == typeof(TaskCanceledException);
    foreach (var c in canceled)
        ProcessCancellation(c);
}
```

All exceptions that are thrown by tasks are wrapped by the TPL in an AggregateException. This exception is thrown when you wait on a task or access its result property. You also retrieve an instance when you access a task's Exception property. The AggregateException class exposes an InnerExceptions property to contain the enumerable list of exceptions that were wrapped by TPL. AggregateException also exposes a Flatten method because some of the inner exceptions might also be instances of AggregateException. The general pattern is to first flatten the aggregate exception before processing the individual exceptions.

You should not use this code block:

```
catch (AggregateException ex) {
    var canceled = from e in ex.InnerExceptions
        where e.GetType() == typeof(TaskCanceledException);
    foreach (var c in canceled)
        ProcessCancellation(c);
}
```

This code does not call the Flatten method of the AggregateException class. You cannot be sure that the InnerExceptions property will not contain additional AggregateException instances based on the available source code. Flattening an aggregate exception is generally simpler than recursively processing nested aggregate exceptions.

You should not use this code block:

```
catch (TaskCanceledException ex) {
    ProcessCancellation(ex);
}
```

Exceptions that are encountered during task execution are not directly thrown to the waiting code. Task Parallel Library wraps all thrown exceptions in an AggregateException, even for a single exception thrown by a single task. You must find the thrown exception in the InnerExceptions property of the aggregate exception.

1.1.67 Question 67 Objective : Manage Program Flow Subobjective : Manage multithreading

You are developing suite of applications by using C# and Microsoft Visual Studio 2015. These applications share a central web service. End users might open and use any combination of these applications on their system. The web service can only support three simultaneous connections from any one client system.

You need to ensure that each client application waits to perform service requests until there are fewer than three pending service operations.

Which class should you use?

Choose the correct answer

- Mutex
- Semaphore
- ReaderWriterLockSlim
- SemaphoreSlim

1.1.67.1 Correction

^ Explanation

You should use the `Semaphore` class. The `Semaphore` class allows you to restrict access to limited resources. When you declare a semaphore, you provide a maximum resource capacity argument and a reserved capacity argument. Some overloads to the `Semaphore` constructor allow you to provide a name. When you pass a name argument, a system-level semaphore is created. Multiple applications that require access to the same limited resource can create `Semaphore` instances that wrap the same system-level semaphore. When the capacity of the semaphore is depleted, each call to the `Semaphore.WaitOne` method will block the thread until a different thread releases the resource by calling the `Semaphore.Release` method.

You should not use the `SemaphoreSlim` class. This class allows you to restrict access to limited resources. When you declare a semaphore, you provide a maximum resource capacity argument and a reserved capacity argument. Unlike the `Semaphore` class, the `SemaphoreSlim` class cannot be used to create a named, system-level semaphore. This means that a `SemaphoreSlim` instance can only synchronize access to resources in a single process. You cannot use this class to synchronize multiple applications in the suite.

You should not use the `Mutex` class. This class synchronizes access to a single resource across multiple threads or processes. This class is similar to a semaphore with a capacity of one.

You should not use the `ReaderWriterLockSlim` class. This class synchronizes access to multiple threads that require mostly read-only access to a resource. The `ReaderWriterLockSlim` class can atomically upgrade a shared-read lock to an exclusive-write lock.

1.1.68 Question 68 Objective : Manage Program Flow Subobjective : Manage multithreading

You are developing an application by using C# and Microsoft Visual Studio 2015. You are writing a class that needs to track the number of instances created since startup. The application that uses this class uses many multithreading techniques.

You have this code:

```
public class CountedClass {  
    private static int _instances = 0;  
    public static int Instances {  
        get { return _instances; }  
    }  
    public CountedClass() {  
    }  
}
```

Which code block should you add to the default constructor to track the number of instances?

Choose the correct answer

lock((object)_instances) {
 _instances++;
}

lock(_instances) {
 _instances++;
}

Interlocked.Add(ref _instances, 1);

_instances++;

1.1.68.1 Correction

Explanation

You should use this code block:

```
Interlocked.Add(ref _instances, 1);
```

The Interlocked class performs atomic operations on value types. This Add method accepts a reference to an int variable, followed by a value. The Interlocked class obtains an exclusive lock on the variable, adds the second argument to the first, and stores the results in the referenced variable.

You should not use this code block:

```
lock(_instances) {  
    _instances++;  
}
```

This code uses the lock statement with the _instances variable. This code does not compile because the lock statement requires a reference type argument. The _instances variable is a value type argument when you declare it as an integer. The compiler exits with the error "int' is not a reference type as required by the lock statement."

You should not use this code block:

```
lock((object)_instances) {  
    _instances++;  
}
```

This code uses the lock statement with a boxed integer. When you box a value type by casting to an object, the runtime creates a separate object in memory in which a copy of the original value is stored. This boxing operation occurs every time the constructor is called, creating a new box object each time. The lock statement only works when you provide a reference to the same object. This code is not thread safe and is subject to thread race conditions.

You should not use this code block:

```
_instances++;
```

This code directly increments the _instances variable. This variable is a static variable, which means that all instances of the class must share a single variable. Code that uses this variable should obtain exclusive access to it because the application is multithreaded. If the operating system preempts a thread after it reads the value, then another thread might execute code that changes the value. Then the first thread reads stale data, and any attempt to make program flow decisions or modify the variable will be based on invalid data. This is a basic race condition that should be avoided by using locks or the Interlocked class.

1.1.69 Question 69 Objective : Manage Program Flow Subobjective : Manage multithreading

You are developing a managed Microsoft Windows service by using C# and Microsoft Visual Studio 2015. This service generates and caches large report objects. Other applications on the system access these objects through a named pipe to the service.

You want the service to signal other processes when a report is ready to access.

Which class should you use?

Choose the correct answer

- AutoResetEvent
- ManualResetEventSlim
- CountdownEvent
- ManualResetEvent

1.1.69.1 Correction



ManualResetEvent

^ Explanation

You should use the `ManualResetEvent` class. This class allows one thread to block all threads that wait on the event until it becomes signaled. If the `ManualResetEvent` instance is named, any thread in any process that creates a new, named instance will obtain a reference to the same instance. To create a named `ManualResetEvent`, call the constructor overload that accepts a Boolean argument and a string argument. The Boolean argument should be false to instantiate the event in a non-signaled state. The string should be a unique name that is shared among all processes that should be signaled by the same event instance.

You should not use the `ManualResetEventSlim` class. This class is similar to a `ManualResetEvent`. However, `ManualResetEventSlim` cannot be used to signal across process boundaries.

You should not use the `AutoResetEvent` class. A single waiting thread is unblocked each time the `Set` method is called. In this example, the managed service does not know exactly how many threads are waiting on the resource, and cannot determine the precise number of times to call the `Set` method.

You should not use the `CountdownEvent` class. This class unblocks waiting threads after the `Signal` method is called a predetermined number of times. This class is useful when one thread needs to wait on multiple other threads.

1.1.70 Question 70 Objective : Manage Program Flow Subobjective : Manage multithreading

You are modifying a C# application by using Microsoft Visual Studio 2015. This application makes database queries in background threads. The background threads perform compute-intensive operations on the results of the queries. You have determined that other threads in your application and in other applications on the system are often starved for processor time.

The background threads process results in a loop. You have decided to pause the process after each iteration to execute other threads that are ready to run. You want to pause only if threads are ready on the same logical processor as the background thread that pauses.

Which statement should you use?

Choose the correct answer

- Thread.Sleep(0);
- Thread.SpinWait(1);
- Thread.Yield();
- Thread.Sleep(1);

1.1.70.1 Correction

^ Explanation

You should use this statement:

```
Thread.Yield();
```

This statement uses the static Yield method of the Thread class. This method releases the current thread's time slice only if another thread is ready on the same processor. If there are no waiting threads on the processor, the current thread continues its time slice. If there are waiting threads of any priority on the same processor, the time slice is yielded and the operating system reschedules the current thread according to its priority.

You should not use this statement:

```
Thread.Sleep(0);
```

This statement uses the static Sleep method of the Thread class. This method accepts an integer argument for the number of milliseconds to wait. When you pass zero to the Sleep method, the operating system only schedules threads of an equal or higher priority. Lower priority threads that experienced starvation before the modification will probably continue to experience starvation after inserting Thread.Sleep(0) into the method.

You should not use this statement:

```
Thread.Sleep(1);
```

This statement uses the static Sleep method of the Thread class. This method accepts an integer argument for the number of milliseconds to wait. When you pass any non-zero integer argument, the thread unconditionally yields its time slice. Your application possibly underutilizes its processor resources because the operating system must switch thread context and reschedule the worker thread even if there are no waiting threads.

You should not use this statement:

```
Thread.SpinWait(1);
```

This statement uses the static SpinWait method of the Thread class. This method accepts an integer argument. The SpinWait method executes an empty, or no-op, loop with the number of iterations determined by the argument. This method is used to occupy a processor for extremely short durations of time when you do not want a thread to yield its time slice but you need to wait for another process to complete. The execution time for the SpinWait method depends on the number of iterations and the processor speed.

1.1.71 Question 71 Objective : Manage Program Flow Subobjective : Manage multithreading

You are troubleshooting a report-retrieval class written in C# by using Microsoft Visual Studio 2015. The class is supposed to generate report objects, cache the report in memory, and return the cached object from the property's get accessor.

Server logs show that, under heavy load, the multithreaded applications that use this class make multiple attempts to generate the report. Reports are computationally expensive to create. Therefore, you need to modify the get accessor to reduce the server load.

You have this code:

```
private Report _report
public Report CurrentReport {
    get {
        if (_report.IsOld)
            _report = GetFreshReport();
        return _report;
    }
}
```

You need to modify this code to call the GetFreshReport method exactly once to create current report objects.

Which code blocks should you use? (Each correct answer presents a complete solution. Choose two.)

Choose the correct answers

`private Lazy _report =
 new Lazy(() => GetFreshReport());
public Report CurrentReport {
 get {
 if (_report.Value.IsOld)
 _report = new Lazy(() => GetFreshReport());
 return _report.Value;
 }
}`

`private Report _report;
private object _locker;
public Report CurrentReport {
 get {
 lock(_locker) {
 if (_report.IsOld)
 _report = GetFreshReport();
 }
 return _report;
 }
}`

private Report _report;
private object _locker;
public Report CurrentReport {
 get {
 Monitor.Enter(_locker);
 try {
 if (_report.IsOld)
 _report = GetFreshReport();
 } finally {
 Monitor.Exit(_locker);
 }
 }
}



private Report _report;
public Report CurrentReport {
 get {
 if (_report.IsOld) {
 _report == null;
 _report = GetFreshReport();
 }
 return _report;
 }
}



1.1.71.1 Correction

```
private Report _report;
private object _locker;
public Report CurrentReport {
    get {
        lock(_locker) {
            if (_report.IsOld)
                _report = GetFreshReport();
        }
        return _report;
    }
}
```

```
private Report _report;
private object _locker;
public Report CurrentReport {
    get {
        Monitor.Enter(_locker);
        try {
            if (_report.IsOld)
                _report = GetFreshReport();
        } finally {
            Monitor.Exit(_locker);
        }
    }
}
```

Explanation

You should use this code block:

```
private Report _report;
private object _locker;
public Report CurrentReport {
    get {
        lock(_locker) {
            if (_report.IsOld)
                _report = GetFreshReport();
        }
        return _report;
    }
}
```

This code uses the lock statement to restrict access to the first thread to enter the code block. The expression in parentheses is a reference to an object that might be accessed by multiple threads. The contents of the code include any statements that involve a read operation that might lead to a write operation. This statement prevents other threads from preempting the first thread after the read operation but before the write operation.

The problem described is known as a race condition. The first thread to read an old report must then generate a new report. However, under heavy load it is possible for other threads to preempt the first thread after it queries the IsOld property but before creating the new report. Each of these threads continue

through the if statement and generate a new report. The lock statement permits only a single thread to execute the code within the lock statement.

Alternatively, you could use this code block:

```
private Report _report;
private object _locker;
public Report CurrentReport {
    get {
        Monitor.Enter(_locker);
        try {
            if (_report.IsOld)
                _report = GetFreshReport();
        } finally {
            Monitor.Exit(_locker);
        }
    }
}
```

The `Monitor` class provides the static `Enter` and `Exit` methods to prevent race conditions. The `Monitor.Enter` method causes the current thread to wait for exclusive access to the object reference passed as the argument. The thread continues executing the remaining code only after exclusive access is granted by the .NET common language runtime (CLR). When the method has completed the operations that require exclusive access to the object, the `Monitor.Exit` method releases the exclusive lock. The C# compiler rewrites the lock statement to use the `Monitor` class.

You should not use this code block:

```
private Lazy<Report> _report =
    new Lazy<Report>(() => GetFreshReport());
public Report CurrentReport {
    get {
        if (_report.Value.IsOld)
            _report = new Lazy<Report>(() => GetFreshReport());
        return _report.Value;
    }
}
```

The generic `Lazy` class is designed to create objects only when requested. The constructor accepts a factory method delegate, usually a lambda expression. The `Lazy` class uses optimized thread-safe code call the delegate the first time the `Value` property is accessed by any thread. This class is ideal when an object is expensive to create, might be accessed by multiple threads, and might not be accessed at all during the lifetime of the owning object. In the example above, the `Report` class is expensive to create and is accessed by multiple threads. However, several report objects are created in the lifetime of the owning object. You should use a lock statement or the `Monitor` class instead.

You should not use this code block:

```
private Report _report;
public Report CurrentReport {
    get {
        if (_report.IsOld) {
            _report == null;
            _report = GetFreshReport();
        }
        return _report;
    }
}
```

This code is subject to a race condition, but with a different outcome from the original scenario. In this code block, if the first thread is preempted after the `IsOld` property returns true but before assigning null to the `_report` field, multiple threads might still call the `GetFreshReport` method. After assigning null to the `_report` field, any other thread that attempts to read the `IsOld` property throws a `NullReferenceException`. You should use a lock statement or the `Monitor` class instead.

1.1.72 Question 72 Objective : Manage Program Flow Subobjective : Manage multithreading

You are writing a C# application by using Visual Studio 2015. The application processes payments. You want to check the balance before the payment is taken from the account.

You need to ensure that the block of code to check the balance can only be accessed by a single thread at one time.

Which block of code should you write?

Choose the correct answer

```
public class Pay
{
    private float balance;
    public Object l = new Object();
    public void Payment(float amount)
    {
        lock (l)
        {
            if(amount < balance)
            {
                balance -= amount;
            }
        }
    }
}
```



```
public class Pay
{
    private float balance;
    private Object l = new Object();
    public void Payment(float amount)
    {
        lock (l)
        {
            if(amount < balance)
            {
                balance -= amount;
            }
        }
    }
}
```



class Pay

```
class Pay
{
    private float balance;
    private string l = "lock";
    public void Payment(float amount)
    {
        lock (l)
        {
            if(amount < balance)
            {
                balance -= amount;
            }
        }
    }
}
```

public class Pay

```
public class Pay
{
    private float balance;
    public void Payment(float amount)
    {
        lock (this)
        {
            if(amount < balance)
            {
                balance -= amount;
            }
        }
    }
}
```

1.1.72.1 Correction

```
public class Pay
{
    private float balance;
    private Object l = new Object();
    public void Payment(float amount)
    {
        lock (l)
        {
            if(amount < balance)
            {
                balance -= amount;
            }
        }
    }
}
```



^ Explanation

You should use the block of code below:

```
public class Pay
{
    private float balance;
    private Object l = new Object();
    public void Payment(float amount)
    {
        lock (l)
        {
            if(amount < balance)
            {
                balance -= amount;
            }
        }
    }
}
```

This code properly creates the lock as a private object and then uses the lock on the block of code that needs to have exclusive access.

You should not use the block of code below:

```
class Pay
{
    private float balance;
    private string l = "lock";
    public void Payment(float amount)
    {
        lock (l)
        {
            if(amount < balance)
            {
                balance -= amount;
            }
        }
    }
}
```

This code uses a literal string for the lock. This is not good practice because literal strings are intrinsically created as global objects in C#. If another block of code placed a lock on the same literal string, a deadlock condition could result.

You should not use the block of code below:

```
public class Pay
{
    private float balance;
    public void Payment(float amount)
    {
        lock (this)
        {
            if(amount < balance)
            {
                balance -= amount;
            }
        }
    }
}
```

This code places a lock on a public object (the Pay class), which is not a good practice because other code would be allowed to place a simultaneous lock on the same Pay object, causing a deadlock condition.

You should not use the block of code below:

```
public class Pay
{
    private float balance;
    public Object l = new Object();
    public void Payment(float amount)
    {
        lock (l)
        {
            if(amount < balance)
            {
                balance -= amount;
            }
        }
    }
}
```

This code places a lock on a public object (the l object), which is not a good practice because other code would be allowed to place a simultaneous lock on the l object, causing a deadlock condition.

1.1.73 Question 73 Objective : Create and Use Types Subobjective : Find, execute, and create at runtime by using reflection.

You are developing a class library by using C# and Microsoft Visual Studio 2015. You are building a class that dynamically generates Comparison methods. These methods are tailored to each entity class by using reflection to discover custom attributes on the entity class.

An Employee entity class has been completed with this declaration:

```
[Comparable]
public class Employee
{
    [CompareBy(3, descending = true)]
    public int ID { get; set; }
    [CompareBy(2)]
    public string FirstName { get; set; }
    [CompareBy(1)]
    public string LastName { get; set; }
}
```

You need to implement the class that supports the [CompareBy(...)] annotation.

Complete the class definition by dragging statement fragments from the left to the appropriate location on the right. You may use items more than once, and you do not have to use each item.

Drag and drop the answers

```
[AttributeUsage(AttributeTargets.Property)]
public class CompareByAttribute : Attribute
{
    public [ ] 
    {
        this.[ ]
    }
    public [ ]
    private [ ]
}
```

CompareByAttribute(bool descending)	CompareByAttribute(int order)	descending = descending;
order = order;	bool descending { get; set; }	int order { get; set; }
bool descending;	int order;	

1.1.73.1 Correction

Drag and drop the answers

```
[AttributeUsage(AttributeTargets.Property)]
public class CompareByAttribute : Attribute
{
    public CompareByAttribute(int order)
    {
        this.order = order;
    }
    public bool descending { get; set; }
    private int order;
}
```

CompareByAttribute(bool descending)	CompareByAttribute(int order)	descending = descending;
order = order;	bool descending { get; set; }	int order { get; set; }
bool descending;	int order;	

Explanation

You should use this code:

```
[AttributeUsage(AttributeTargets.Property)]
public class CompareByAttribute : Attribute
{
    public CompareByAttribute(int order)
    {
        this.order = order;
    }
    public bool descending { get; set; }
    private int order;
}
```

To create a custom attribute class to be used for **annotating** (also called **decorating**) a class, you should create a class that inherits from the `System.Attribute` base class. An Attribute-derived class is generally a very simple class whose definition is determined by the intended usage.

The bare minimum annotation for any attribute requires square brackets surrounding the class name. Because attribute classes generally include the suffix `Attribute` in the class name, the compiler permits you to omit the suffix for brevity. Assume your class has this definition:

```
public class CompareByAttribute : System.Attribute {}
```

With this definition, the compiler permits this annotation:

```
[CompareBy]
```

Many attributes require positional parameters when using annotation syntax. Positional parameters are always declared as constructor parameters. Your annotation must include arguments for every parameter of one of the accessible constructors. Assume your desired annotation follows:

```
[CompareBy(1)]
```

This annotation implies that the CompareByAttribute class has a constructor that declares a single integer parameter. To implement this attribute, your class might have this definition:

```
public class CompareByAttribute : System.Attribute
{
    public CompareByAttribute(int order)
    {
        this.order = order;
    }
    private int order;
}
```

If you want to have optional named arguments when constructing your annotation, your attribute class needs to have a public readable and writeable property with the desired name. For instance, assume you wanted to use this annotation:

```
[CompareBy(1, descending = true)]
```

This desired annotation implies that your attribute should be defined as follows:

```
public class CompareByAttribute : System.Attribute
{
    public CompareByAttribute(int order)
    {
        this.order = order;
    }
    private int order;
    public bool descending { get; set; }
}
```

There are many other options available when you create an attribute class. One of the most common options is to limit which types of declarations may include your attribute as an annotation. The AttributeUsageAttribute class allows you to combine the bitwise flags in the AttributeTargets enumeration to limit the usage of your attribute. For instance, you can indicate that your attribute may only annotate property declarations by passing the AttributeTargets.Property positional parameter.

Objective:

Create and Use Types

Subobjective:

Find, execute, and create types at runtime by using reflection.

1.1.74 Question 74 Objective : Create and Use Types Subobjective : Find, execute, and create at runtime by using reflection.

You are developing a class library by using C# and Visual Studio 2015. You need to generate and compile a Comparison<CustomClass> method at runtime. This method compares two instances of CustomClass based on an IEnumerable of PropertyInfo instances. You need to be able to generate and save the source code to this method.

You have already used reflection to generate the list of properties by which you can compare two instances of CustomClass: (Line numbers are included for reference purposes only.)

```
01 public Comparison<CustomClass> CreateMethod(IEnumerable<PropertyInfo> props)
02 {
03     Comparison<CustomClass> comp;
04
05     comp = CompileAndLoadMethod(dynamicMethod);
06
07     return comp;
07 }
```

Which line of code when inserted at line 04 could be used to generate the Comparison<CustomClass> method, save the C# source code, and compile and load the method?

Choose the correct answer

- DynamicMethod dynamicMethod = CreateCustomMethod(props);
- Expression dynamicMethod = CreateCustomMethod(props);
- MulticastDelegate dynamicMethod = CreateCustomMethod(props);
- CodeCompileUnit dynamicMethod = CreateCustomMethod(props);

1.1.74.1 Correction



```
CodeCompileUnit dynamicMethod = CreateCustomMethod(props);
```

^ Explanation

You should use this line of code at line 04:

```
CodeCompileUnit dynamicMethod = CreateCustomMethod(props);
```

The `CodeCompileUnit` class is one of the main classes used by the `CodeDOM` library. `CodeDOM` is a code-generation capability built into the .NET Framework Class Library to support runtime generation of source code. You can use the `CodeCompileUnit` class to create an object model (or code graph) that represents the hierarchy of code blocks found in .NET applications. This object can be converted to source code and compiled by using the `CSharpCodeProvider` class. Once compiled, the resulting assembly can be loaded into memory using the `Reflection API`.

You should not use this line of code at line 04:

```
DynamicMethod dynamicMethod = CreateCustomMethod(props);
```

The `DynamicMethod` class is part of the `Reflection API`. You can generate (emit) Common Intermediate Language (CIL) code by using this class. The resulting CIL code can be just-in-time (JIT) compiled and loaded into memory. The `Reflection API` does not support generating source code in C# because CIL code supports a superset of the C# language features.

You should not use this line of code at line 04:

```
Expression dynamicMethod = CreateCustomMethod(props);
```

The `Expression` class is part of the Language-Integrated Query (LINQ) API. Instances of `Expression` may be compiled to delegates, or they may be read and modified as data. The `Expression` class does not support generating C# source code.

You should not use this line of code at line 04:

```
MulticastDelegate dynamicMethod = CreateCustomMethod(props);
```

The `MulticastDelegate` class allows a single invocation of a delegate to execute one or more methods. For methods that return an expression, the return value of the invocation is the return value only of the last delegate in the list. You should not use the `Delegate` or `MulticastDelegate` classes because they do not provide built-in support for dynamically generating code of any kind.

1.1.75 Question 75 Objective : Create and Use Types Subobjective : Find, execute, and create at runtime by using reflection.

You are creating a C# application by using Visual Studio 2015. You have created a custom attribute using the code below.

```
[System.AttributeUsage(System.AttributeTargets.Class |  
System.AttributeTargets.Struct)]  
public class Documentation: System.Attribute  
{  
    private string Author;  
    public string Modified;  
    public Documentation(string Author)  
    {  
        this.Author = Author;  
        Modified = DateTime.Now.ToString();  
    }  
}
```

You need to apply the attribute.

Which two lines of code could you write? (Each correct answer presents a complete solution. Choose two.)

Choose the correct answers

- [Documentation(Author = "Jay Adams", Modified = "12/01/2015")]
- [Documentation("Jay Adams", Modified = "12/01/2015")]
- [Documentation("Jay Adams", "12/01/2015")]
- [Documentation("Jay Adams")]
- [Documentation(Author = "Jay Adams")]

1.1.75.1 Correction

Explanation

You could use the following line of code:

```
[Documentation("Jay Adams", Modified = "12/01/2015")]
```

This code correctly uses "Jay Adams" as a positional parameter for the Author member of the Documentation attribute, and "12/01/2015" as a named parameter for the Modified member of the Documentation attribute. Because Author is a private member used in the constructor, it must be used as a positional parameter and cannot be used as a named parameter. Because Modified is a public member that is not listed as a parameter of the constructor, it must be called as a named parameter and cannot be used as a positional parameter.

As an alternative, you could use the following line of code:

```
[Documentation("Jay Adams")]
```

This code correctly uses "Jay Adams" as a positional parameter for the Author member of the Documentation attribute. A parameter for Modified is not required. Modified is initialized in the constructor.

You should not use the following line of code:

```
[Documentation(Author = "Jay Adams", Modified = "12/01/2015")]
```

This code incorrectly uses "Jay Adams" as a named parameter for the Author member of the Documentation attribute. Because Author is a private member assigned a value in the constructor, it must be used as a positional parameter and cannot be used as a named parameter.

You should not use the following line of code:

```
[Documentation("Jay Adams", "12/01/2015")]
```

This code incorrectly uses "12/01/2015" as a positional parameter for the Modified member of the Documentation attribute. Because Modified is a public member and is not passed as an argument to the constructor, it must be called as a named parameter and cannot be used as a positional parameter. You should not use the following line of code:

```
[Documentation(Author = "Jay Adams")]
```

This code incorrectly uses "Jay Adams" as a named parameter for the Author member of the Documentation attribute. Because Author is a private member used in the constructor, it must be used as a positional parameter and cannot be used as a named parameter.

- -

1.1.76 Question 76 Objective : Create and Use Types Subobjective : Find, execute, and create at runtime by using reflection.

You are creating a C# application by using Visual Studio 2015. You have created a custom attribute using the code below:

```
[System.AttributeUsage(System.AttributeTargets.Class | System.AttributeTargets.Struct)]
public class Documentation: System.Attribute
{
    private string Author;
    public string Modified;
    public Documentation(string Author)
    {
        this.Author = Author;
        Modified = DateTime.Now.ToString();
    }
}
```

You need to apply the attribute.

For each of the following statements, select Yes if the statement is true or No if the statement is false.

Statements	yes	no
Author can be used as a positional parameter.	<input type="radio"/>	<input type="radio"/>
Modified can be used as a positional parameter.	<input type="radio"/>	<input type="radio"/>
Author can be used as a named parameter.	<input type="radio"/>	<input type="radio"/>
Modified can be used as a named parameter.	<input type="radio"/>	<input type="radio"/>
The Documentation attribute can be applied to a method.	<input type="radio"/>	<input type="radio"/>

1.1.76.1 Correction

Statements	yes	no
Author can be used as a positional parameter.	<input checked="" type="radio"/>	<input type="radio"/>
Modified can be used as a positional parameter.	<input type="radio"/>	<input checked="" type="radio"/>
Author can be used as a named parameter.	<input type="radio"/>	<input checked="" type="radio"/>
Modified can be used as a named parameter.	<input checked="" type="radio"/>	<input type="radio"/>
The Documentation attribute can be applied to a method.	<input type="radio"/>	<input checked="" type="radio"/>

^ Explanation

You should choose Yes for Author can be used as a positional parameter. Private data members that are included in the constructor of custom attributes are always used as a positional parameter and they cannot be used as named parameters.

You should choose No for Modified can be used as a positional parameter. Public data members of a custom attribute are always used as named parameters and cannot be used as positional parameters.

You should choose No for Author can be used as a named parameter. Private data members that are included in the constructor of custom attributes are always used as positional parameters and they cannot be used as named parameters.

You should choose Yes for Modified can be used as a named parameter. Public data members of a custom attribute are always used as named parameters and cannot be used as positional parameters.

You should choose No for The Documentation attribute can be applied to a method. The System.AttributeTargets.Class | System.AttributeTargets.Struct usage attributes specify that this attribute can only be applied to a class or struct. To target methods, System.AttributeTargets.Method would have to be added to acceptable usage attributes.

1.1.77 Question 77 Objective : Create and Use Types Subobjective : Enforce encapsulation

You are developing a class library by using C# and Visual Studio 2015. You need to complete a class to represent employee data. Your application stores employee data in a database. One of the tables that stores employee data includes a DateOfBirth field. Your application must be able to calculate employees' ages from this field.

You have already completed this code:

```
public class Employee
{
    private DateTime _dob;
    public DateTime DateOfBirth
    {
        get { return _dob; }
        set
        {
            this._dob = value;
            this.Age = (int)((DateTime.Now - value).TotalDays / 365.25);
        }
    }
}
```

You need to implement the Age member in the employee class. Because you cannot store the age in the database, the Age member must be read-only.

Which declaration correctly implements the Age member?

```
public int Age {get ; private set;}

public int Age {get ; set;}

private int Age;

public readonly int Age;
```

1.1.77.1 Correction

^ Explanation

You should use this declaration:

```
public int Age { get; private set; }
```

This declares a public property of Employee instances named Age. The data type of this property is an integer. When the get and set keywords are followed by semi-colons instead of code blocks, the compiler automatically creates a private backing variable that is returned from the property getter, and stored in the property setter.

The compiler allows you to use a different access modifier for either the get or set keywords. In this case, the set keyword is private. This makes the property read-only from the perspective of any code outside of the Employee class declaration. Inside the class block, however, the property may still be assigned. This is used by the set block of the DateOfBirth property.

You should not use this declaration:

```
public int Age { get; set; }
```

This declares a public automatic property. The set block is callable from code outside the Employee class declaration because the access modifier applies to both the get and set keywords. This fails the requirement that the Age should be read-only.

You should not use following declaration:

```
private int Age;
```

This declares a private integer variable called Age. This variable cannot be accessed outside of the Employee class block. It is a best practice to declare instance variables as private. However, code outside of the Employee class block is not be able to retrieve the calculated Age.

You should not use this declaration:

```
public readonly int Age;
```

This declares a public integer variable called Age. This variable may only be initialized from a constructor in the Employee class because of the readonly keyword. After this variable is initialized, it may not be assigned a new value, even from within the Employee class block. The compiler will not permit the assignment in the DateOfBirth property set block.

1.1.78 Question 78 Objective : Manage Program Flow Subobjective : Manage multithreading

You are writing a C# application by using Visual Studio 2015. The application creates an audit log of transactions.

You need to ensure that the block of code to write to the log file can only be accessed by a single thread at one time.

Which block of code should you write?

Choose the correct answer

```
public class Log<T>
{
    private List<T> list = new List<T>();
    public void Add(T entry)
    {
        try
        {
            Monitor.Enter(list);
        }
        catch
        {
            Monitor.Exit(list);
        }
        list.Add(entry);
    }
}
```



```
public class Log<T>
{
    private List<T> list = new List<T>();
    public void Add(T entry)
    {
        Monitor.Enter(list);
        try
        {
            list.Add(entry);
        }
        finally
        {
            Monitor.Exit(list);
        }
    }
}
```



 ✓

```
public class Log<T>
{
    private List<T> list = new List<T>();
    public void Add(T entry)
    {
        Monitor.Enter(list);
        try
        {
            list.Add(entry);
        }
        catch(Exception e)
        {
            Monitor.Exit(list);
        }
    }
}
```

 ✓

```
public class Log<T>
{
    private List<T> list = new List<T>();
    public void Add(T entry)
    {
        try
        {
            Monitor.Enter(list);
        }
        finally
        {
            Monitor.Exit(list);
        }
        list.Add(entry);
    }
}
```

1.1.78.1 Correction

```
public class Log<T>
{
    private List<T> list = new List<T>();
    public void Add(T entry)
    {
        Monitor.Enter(list);
        try
        {
            list.Add(entry);
        }
        finally
        {
            Monitor.Exit(list);
        }
    }
}
```

^ Explanation

You should use the block of code below:

```
public class Log<T>
{
    private List<T> list = new List<T>();
    public void Add(T entry)
    {
        Monitor.Enter(list);
        try
        {
            list.Add(entry);
        }
        finally
        {
            Monitor.Exit(list);
        }
    }
}
```

This block of code incorrectly places the Monitor.Exit call in the catch block. This means the Monitor will only exit if there is an error. Instead, the Monitor.Exit call should be placed in a finally block to ensure that it is called whether an exception is thrown or not.

You should not use the block of code below:

```
public class Log<T>
{
    private List<T> list = new List<T>();
    public void Add(T entry)
    {
        try
        {
            Monitor.Enter(list);
        }
        finally
        {
            Monitor.Exit(list);
        }
        list.Add(entry);
    }
}
```

This block of code incorrectly places the Monitor.Enter call in a try block, and it leaves the code to be locked outside the try block.

When you use the Monitor class to lock code, you must ensure that the monitor explicitly exists both when the code works and when there is an error. Therefore, the block of code to be locked must be placed in a try block just after the Monitor.Enter call. Then, placing the Monitor.Exit call in the finally block ensures that it is called whether an exception is thrown or not.

You should not use the block of code below:

```
public class Log<T>
{
    private List<T> list = new List<T>();
    public void Add(T entry)
    {
        Monitor.Enter(list);
        try
        {
            list.Add(entry);
        }
        catch(Exception e)
        {
            Monitor.Exit(list);
        }
    }
}
```

You should not use the block of code below:

```
public class Log<T>
{
    private List<T> list = new List<T>();
    public void Add(T entry)
    {
        try
        {
            Monitor.Enter(list);
        }
        catch
        {
            Monitor.Exit(list);
        }
        list.Add(entry);
    }
}
```

This block of code incorrectly places the `Monitor.Enter` call in a try block, and it leaves the code to be locked outside the try block. The code also places the `Monitor.Exit` call in a catch block, which means that it will only be called if there is an error. Instead, the `Monitor.Exit` call should be placed in a finally block.

1.1.79 Question 79 Objective : Manage Program Flow Subobjective : Manage multithreading

You are writing a C# application by using Visual Studio 2015. The application processes long calculations on multiple threads.

You write the code below (line numbers added for reference):

```
01 static WaitHandle[] handles = new WaitHandle[]
02 {
03     new AutoResetEvent(false),
04     new AutoResetEvent(false),
05     new AutoResetEvent(false)
06 };
07 ThreadPool.QueueUserWorkItem(new WaitCallback(Calculate), waitHandles[0]);
08 ThreadPool.QueueUserWorkItem(new WaitCallback(Calculate), waitHandles[1]);
09
10 Summarize();
```

The Calculate function processes the calculations. After all of the calculations are completed, you want the Summarize function to run.

Which line of code should you write at line 09?

Choose the correct answer

- `WaitHandle.WaitAll(waitHandles);`
- `WaitHandle.WaitAny(waitHandles);`
- `WaitHandle.SignalAndWait(waitHandles[0],waitHandles[1]);`
- `handles.ToList().ForEach(h => h.WaitOne());`

1.1.79.1 Correction

^ Explanation

You should use the line of code below:

```
WaitHandle.WaitAll(waitHandles);
```

The WaitAll method instructs the thread management to wait until all of the wait handles in the waitHandles array have completed before moving on to the next statement.

You should not use the line of code below:

```
WaitHandle.WaitAny(waitHandles);
```

The WaitAny method instructs the thread management to move on to the next statement once any one of the wait handles in the waitHandles array has completed. This may run the Summarize function before all of the calculations are complete.

You should not use the line of code below:

```
WaitHandle.SignalAndWait(waitHandles[0],waitHandles[1]);
```

The SignalAndWait method instructs the thread management to move on to wait for the second wait handle once the first wait handle has completed. This means the next statement will be executed once wait handle 1 and 2 have been completed, but it will not wait until the third wait handle has completed. This may run the Summarize function before all of the calculations are complete.

You should not use the line of code below:

```
handles.ToList().ForEach(h => h.WaitOne());
```

The WaitOne method instructs the thread management to move on to the next statement once the current wait handle has completed. This may run the Summarize function before all of the calculations are complete.

1.1.80 Question 80 Objective : Debug Applications and Implement Security Subobjective : Implement diagnostics in an application

You use Microsoft .NET Framework 4.5 to create an application. A custom event log named AppCenter exists.

You need to write the message Invalid to the custom event log.

Which code segment should you use?

Choose the correct answer

EventLog eventLog = new EventLog();
if (!EventLog.SourceExists("MySource"))
{
 eventLog.CreateEventSource("MySource", "AppCenter");
}
eventLog.WriteEntry("Invalid");

if (!EventLog.SourceExists("MySource"))
{
 EventLog.CreateEventSource("MySource", "AppCenter");
}
EventLog.WriteEntry(null, "Invalid");

if (!EventLog.SourceExists("MySource"))
{
 EventLog.CreateEventSource("MySource", "AppCenter");
}
EventLog eventLog = new EventLog("AppCenter");
eventLog.Source="MySource";
eventLog.WriteEntry("Invalid");

EventLog eventLog = new EventLog("AppCenter");
eventLog.WriteEntry("Invalid");

1.1.80.1 Correction

```
if (!EventLog.SourceExists("MySource"))
{
    EventLog.CreateEventSource("MySource", "AppCenter");
}
EventLog eventLog = new EventLog("AppCenter");
eventLog.Source="MySource";
eventLog.WriteEntry("Invalid");
```

^ Explanation

You should use this code segment:

```
if (!EventLog.SourceExists("MySource"))
{
    EventLog.CreateEventSource("MySource", "AppCenter");
}
EventLog eventLog = new EventLog("AppCenter");
eventLog.Source="MySource";
eventLog.WriteEntry("Invalid");
```

This code first calls the `SourceExists` method of the `EventLog` class to determine whether or not an event source exists. Messages must be associated with event sources. If it does not exist, it calls the `CreateEventSource` method to create an event source in the `AppCenter` custom event log. It then creates an `EventLog` instance, passing to its constructor the name of the custom event log to retrieve. It sets the `Source` property to the event source, and then calls the `WriteEntry` method to write the message `Invalid`. The message is written to the `AppCenter` event log for the source `MySource`.

You should not use this code segment:

```
if (!EventLog.SourceExists("MySource"))
{
    EventLog.CreateEventSource("MySource", "AppCenter");
}
EventLog.WriteEntry(null, "Invalid");
```

This code correctly creates an event source, but it passes a null reference as the first parameter to the static `WriteEntry` method of the `EventLog` class. You must pass the name of the source as the first parameter. Passing a null reference causes an exception to be thrown. It does not write a message to the event log.

You should not use this code segment:

```
EventLog eventLog = new EventLog("AppCenter");
eventLog.WriteEntry("Invalid");
```

You must set the `Source` property before calling the `WriteEntry` method. Not doing so causes an exception to be thrown, so a message is not written to the event log.

You should not use this code segment:

```
EventLog eventLog = new EventLog();
if (!EventLog.SourceExists("MySource"))
{
    eventLog.CreateEventSource("MySource", "AppCenter");
}
eventLog.WriteEntry("Invalid");
```

This code does not specify the name of the event log as the parameter to the EventLog constructor. Because you are writing to a custom log, you must specify its name. You must also set the Source property before calling the WriteEntry method. Not doing so causes an exception to be thrown, so a message is not written to the event log.

1.1.81 Question 81 Objective : Debug Applications and Implement Security Subobjective : Validate application input

You use Microsoft .NET Framework 4.5 to create an application. Your application receives this data from a Web service:

```
{"ID":"Item1","ProviderParameters":[{"Key":"RateRequestType","Value":"Rate"}],  
"ShipFromAddress":  
{"Street":"10 M Street Street","City":"Kennesaw","PostalCode":"30144","State":"GA"},  
"ShipToAddress":  
{"Street":"1 Uptown","City":"Redmond","PostalCode":"98052","State":"WA"},  
"Size":{"Height":3,"Length":1,"Unit":0,"Width":2},  
"Weight":{"Unit":0,"Value":5}}
```

You need to ensure that this data can be converted to an appropriate type in your application.

Which class should you use?

Choose the correct answer

[XmlSerializer](#)

[UriTypeConverter](#)

[TargetConverter](#)

[JavaScriptSerializer](#)

1.1.81.1 Correction



JavaScriptSerializer

^ Explanation

You should use the `JavaScriptSerializer` class. This class allows you to serialize and deserialize Java Script Object Notation (JSON) data. JSON is a data format that allows you to represent an object graph as a string literal. Within the literal object graph, each object is enclosed in curly braces {}, and each property-value pair is separated by a colon. For example, this subset of the JSON data represents a `ShipFromAddress` property that is set to an object instance that has `Street`, `City`, `PostalCode`, and `State` properties:

```
"ShipFromAddress":  
{"Street":"10 M Street Street","City":"Kennesaw","PostalCode":"30144","State":"GA"}
```

You should not use the `XmlSerializer` class. This class allows you to serialize and deserialize XML data. In this scenario, the data is not represented as XML.

You should not use the `TargetConverter` class. The ASP.NET Hyperlink control uses this class to convert a collection of target URLs to a string.

You should not use the `UriTypeConverter` class. This class allows you to perform conversions between `Uri` instances and string representations of URLs.

1.1.82 Question 82 Objective : Create and Use Types Subobjective : Enforce encapsulation

You are writing a C# application by using Visual Studio 2015. You are designing a base class that will be inherited by several other classes.

The class hierarchy must meet the following requirements:

- * You need to define some methods in the base class that can be overridden, while other methods cannot be overridden.
- * You need to define some methods in the base class that do not have an implementation. These classes must be implemented by the classes that inherit from the base class.

You write the following code (line number are included for reference only):

```
01 public class Weapons
02 {
03     protected int damage;
04     protected int range;
05     public int CalculateDamage(int modifier)
06     {
07         return damage * modifier;
08     }
09     public bool HasHit(int roll, int threshold);
10 }
11 public class Sword : Weapons
12 {
13     private int size;
14     public int CalculateDamage(int modifier)
15     {
16         return damage * size * modifier;
17     }
18     public bool HasHit(int roll, int threshold)
19     {
20         return roll * size < threshold;
21     }
22 }
```

You need to choose the correct access modifier for each method.

Which access modifier should you use for each method? Place the correct access modifier next to each of the line numbers. A modifier may be used once, more than once, or not at all.

Drag and drop the answers

Modifier	Line Number
	Line 01
	Line 05
	Line 09
	Line 14
	Line 18

 abstract sealed virtual override

1.1.82.1 Correction

Modifier	Line Number
abstract	Line 01
virtual	Line 05
abstract	Line 09
override	Line 14
override	Line 18

abstract

sealed

virtual

override

Explanation

You should use the abstract modifier on line 01. The abstract modifier is used to define a class that contains abstract methods. An abstract method is a method that does not contain an implementation and must be overridden and implemented by any class that inherits the abstract class.

You should use the virtual modifier on line 05. The virtual modifier is used to indicate that a method in a base class can be overridden in a class that inherits the base class. The method that overrides the virtual method must be preceded by the override modifier. Because the CalculateDamage method also appears in the Sword class, it must be declared as virtual in the Weapon class.

You should use the abstract modifier on line 09. The abstract modifier is used to define a method that contains has no implementation. Abstract methods can only be defined in an abstract class. Any class that inherits the base class must override the method and provide an implementation for it. The method that overrides the virtual method must be preceded by the override modifier. An abstract method is automatically treated as a virtual method. Because the HasHit method does not contain an implementation, it should be declared as abstract to force the Sword class to provide an implementation.

You should use the override modifier on line 14 and line 18. The override modifier indicates that a method in the current class is overriding a method that was declared as abstract or virtual in the parent class. Both CalculateDamage and HasHit in the Sword class override the respective methods in the Weapons class.

You should not use the sealed class on any of the lines. The sealed modifier is applied to an entire class and indicates that none of the methods in that class can be overridden. Since you want to override methods defined in the Weapon class, you cannot declare the Weapon class as sealed.

1.1.83 Question 83 Objective : Implement Data Access Subobjective : Query and manipulate data and object by using LINQ

You use Microsoft .NET Framework 4.5 to create an application. The application must process this XML data:

```
<Reservation>
  <Passenger Name="John Doe" FrequentFlyerNumber="12345"/>
</Reservation>
```

This method exists:

```
public XElement CreateReservation(string passengerName, string frequentFlyerNumber)
{
  XElement xml = null;
  return xml;
}
```

You must implement this method to set the Name attribute to the value of passengerName and the FrequentFlyerNumber attribute to the value of frequentFlyerNumber.

You need to use LINQ to XML to create an instance of XElement that represents the previous XML data.

Which code segment should you use?

Choose the correct answer

XElement xml =
 new XElement("Reservation", null),
 new XElement("Passenger", null),
 new XAttribute("Name", passengerName),
 new XAttribute("FrequentFlyerNumber", frequentFlyerNumber)
);

XElement xml =
 new XElement("Reservation",
 new XElement("Passenger",
 new XAttribute("Name", passengerName),
 new XAttribute("FrequentFlyerNumber", frequentFlyerNumber)
)
);

XElement xml =
 new XElement("Reservation",
 new XElement("Passenger",
 "Name=passengerName FrequentFlyerNumber=frequentFlyerNumber"
)
);

XElement xml = new XElement("",
 new XElement("Reservation", null),
 new XElement("Passenger",
 new XAttribute("Name", passengerName),
 new XAttribute("FrequentFlyerNumber", frequentFlyerNumber)
));

1.1.83.1 Correction

```
 XElement xml =  
    new XElement("Reservation",  
        new XElement("Passenger",  
            new XAttribute("Name", passengerName),  
            new XAttribute("FrequentFlyerNumber", frequentFlyerNumber)  
        )  
    );
```

^ Explanation

You should use this code segment:

```
XElement xml =  
    new XElement("Reservation",  
        new XElement("Passenger",  
            new XAttribute("Name", passengerName),  
            new XAttribute("FrequentFlyerNumber", frequentFlyerNumber)  
        )  
    );
```

The XElement class represents an XML element and its child nodes and attributes. Its two-parameter constructor accepts an XName instance as the first parameter, and a variable number of Object instances as the second parameter. XName can be cast to and from strings. If you specify a string, it becomes the name of the XML element. For the second parameter, you can specify a string to represent the value of the element, another XElement instance to represent a child element, or an XAttribute instance to represent an XML attribute.

You should not use this code segment:

```
XElement xml = new XElement("",  
    new XElement("Reservation", null),  
    new XElement("Passenger", null),  
    new XAttribute("Name", passengerName),  
    new XAttribute("FrequentFlyerNumber", frequentFlyerNumber)  
);
```

This code attempts to create a root XML element with no name, as specified by the empty string.

You should not use this code segment:

```
XElement xml = new XElement("",  
    new XElement("Reservation", null),  
    new XElement("Passenger",  
        new XAttribute("Name", passengerName),  
        new XAttribute("FrequentFlyerNumber", frequentFlyerNumber)  
    ));
```

This code attempts to create a root XML element with no name and two child XML elements named Passenger and Reservation.

You should not use this code segment:

```
XElement xml =  
    new XElement("Reservation",  
        new XElement("Passenger",  
            "Name=passengerName FrequentFlyerNumber=frequentFlyerNumber"  
        )  
    );
```

This code would generate this XML, which does not match the XML in the scenario:

```
<Reservation>  
  <Passenger>  
    Name=passengerName FrequentFlyerNumber=frequentFlyerNumber  
  </Passenger>  
</Reservation>
```

1.1.84 Question 84 Objective : Debug Application and Implement Security Subobjective : Perform symmetric and asymmetric encryption.

You use Microsoft .NET Framework 4.5 to create an application. This code exists to encrypt data:

```
string original = GetString();
byte[] key = GetKey();
byte[] iv = GetIV();
byte[] encryptedData = null;
using (Rijndael r = Rijndael.Create())
{
    r.Key = key;
    r.IV = iv;
    ICryptoTransform encryptor = r.CreateEncryptor(key, iv);
    using (MemoryStream ms = new MemoryStream())
    {
        using (CryptoStream cs = new CryptoStream(ms, encryptor, CryptoStreamMode.Write))
        {
            using (StreamWriter writer = new StreamWriter(cs))
            {
                writer.WriteLine(original);
            }
            encryptedData = ms.ToArray();
        }
    }
}
```

This code exists to decrypt the data (line numbers are included for reference only):

```
01 using (Rijndael r = Rijndael.Create())
02 {
03     ICryptoTransform decryptor = r.CreateDecryptor();
04     using (MemoryStream ms = new MemoryStream(encryptedData))
05     {
06         using (CryptoStream cs = new CryptoStream(ms, decryptor, CryptoStreamMode.Read))
07         {
08             using (StreamReader reader = new StreamReader(cs))
09             {
10                 string decryptedString = reader.ReadToEnd();
11             }
12         }
13     }
14 }
```

When you run the previous code, you discover that the encrypted data is not decrypted correctly.

You need to solve this problem.

What should you do?

Choose the correct answer

Replace line 04 with this code:



```
using (MemoryStream ms = new MemoryStream())
```

Replace line 03 with this code:



```
ICryptoTransform decryptor = r.CreateDecryptor(key, iv);
```

Replace line 08 with this code:



```
using (StreamWriter reader = new StreamWriter(cs))
```

Replace line 06 with this code:



```
using (CryptoStream cs = new  
CryptoStream(ms, decryptor, CryptoStreamMode.Write))
```

1.1.84.1 Correction



Replace line 03 with this code:

```
ICryptoTransform decryptor = r.CreateDecryptor(key, iv);
```

^ Explanation

You should replace line 03 with this code:

```
ICryptoTransform decryptor = r.CreateDecryptor(key, iv);
```

This code passes the key and initialization vector to the CreateDecryptor method. This is necessary so that the same key and initialization vector that is used to encrypt the data can be used to decrypt the data. The RijndaelManaged class performs symmetric encryption, which uses the same key to encrypt and decrypt data.

You should not replace line 04 with this code:

```
using (MemoryStream ms = new MemoryStream())
```

When decrypting data, you must pass the encrypted byte array to the `MemoryStream` constructor because you must read from the underlying `MemoryStream` instance. A `CryptoStream` instance is created from the `MemoryStream` instance. As you read from the `CryptoStream` instance, it internally reads from the `MemoryStream` instance, decrypting its contents in the process.

You should not replace line 06 with this code:

```
using (CryptoStream cs = new CryptoStream(ms, decryptor, CryptoStreamMode.Write))
```

You must pass `CryptoStreamMode.Read` to the `CryptoStream` instance instead of `CryptoStreamMode.Write`. You should use `CryptoStreamMode.Write` when you encrypt data and `CryptoStreamMode.Read` when you decrypt data.

You should not replace line 08 with this code:

```
using (StreamWriter reader = new StreamWriter(cs))
```

You should use the `StreamReader` class because you must read from a stream to decrypt its contents. You should use the `StreamWriter` class when you encrypt data because you are encrypting its contents as you write to the stream.

1.1.85 Question 85 Objective : Create and Use Types Subobjective : Manage the object life cycle

You are developing an application by using C# and Visual Studio 2015. You have an array initialization method with this declaration:

```
unsafe private void InitializeArrayOfInts(int* firstElement, int numElements)
```

You need to use this method to initialize a new array of integers.

Which code block should you use to perform the initialization?

Choose the correct answer

```
private void SomeMethod()
{
    var arrayOfInts = new int[2073600];
    fixed(int* firstElement = &arrayOfInts[0])
    {
        InitializeArrayOfInts(firstElement, arrayOfInts.Length);
    }
}
```



```
unsafe private void SomeMethod()
{
    var arrayOfInts = new int[2073600];
    fixed(int* firstElement = &arrayOfInts[0])
    {
        InitializeArrayOfInts(firstElement, arrayOfInts.Length);
    }
}
```



```
private void SomeMethod()
{
    var arrayOfInts = new int[2073600];
    unchecked
    {
        int* firstElement = &arrayOfInts[0];
        InitializeArrayOfInts(firstElement, arrayOfInts.Length);
    }
}
```



```
unsafe private void SomeMethod()
{
    var arrayOfInts = new int[2073600];
    unchecked
    {
        int* firstElement = &arrayOfInts[0];
        InitializeArrayOfInts(firstElement, arrayOfInts.Length);
    }
}
```



1.1.85.1 Correction

```
unsafe private void SomeMethod()
{
    var arrayOflnts = new int[2073600];
    fixed(int* firstElement = &arrayOflnts[0])
    {
        InitializeArrayOflnts(firstElement,, arrayOflnts.Length);
    }
}
```

Explanation

You should use this code block:

```
unsafe private void SomeMethod()
{
    var arrayOflnts = new int[2073600];
    fixed(int* firstElement = &arrayOflnts[0])
    {
        InitializeArrayOflnts(firstElement,, arrayOflnts.Length);
    }
}
```

The unsafe keyword in the method declaration indicates to the compiler that the method might include direct memory manipulation. Unsafe operations require special Code Access Security privileges and are not allowed for all applications.

The fixed block indicates to the .NET Common Language Runtime (CLR) that the Garbage Collector is not permitted to move the ArrayOflnts object while the code block executes. The fixed block initializer statements indicate that the address of (& operator) the first element of the array will be directly referred to by an integer pointer (int* type) named firstElement. The fixed block is required any time unsafe operations are performed against managed objects.

You should not use this code block because it is missing the required unsafe keyword on the method declaration:

```
private void SomeMethod()
{
    var arrayOfInts = new int[2073600];
    fixed(int* firstElement = &arrayOfInts[0])
    {
        InitializeArrayOfInts(firstElement,, arrayOfInts.Length);
    }
}
```

You should not use this code block because it uses an unchecked block instead of a fixed block:

```
unsafe private void SomeMethod()
{
    var arrayOfInts = new int[2073600];
    unchecked
    {
        int* firstElement = &arrayOfInts[0];
        InitializeArrayOfInts(firstElement,, arrayOfInts.Length);
    }
}
```

The unchecked keyword is used to prevent the compiler from performing overflow-checking.

You should not use this code block because it is missing the required unsafe keyword and because it uses an unchecked block instead of an unsafe block:

```
private void SomeMethod()
{
    var arrayOfInts = new int[2073600];
    unchecked
    {
        int* firstElement = &arrayOfInts[0];
        InitializeArrayOfInts(firstElement,, arrayOfInts.Length);
    }
}
```

1.1.86 Question 86 Objective : Create and Use Types Subobjective : Manage the object life cycle

You are developing an application by using C# and Visual Studio 2015. You are using an unmanaged buffer for image processing. You have this code: (Line numbers are included for reference purposes only.)

```
01 IntPtr buffer = IntPtr.Zero;
02 int bufferSize = 2073600 * sizeof(int);
03 try
04 {
05     buffer = Marshal.AllocHGlobal(bufferSize);
06
07     ... // Use the buffer
08 }
09 finally
10 {
11     if (buffer == IntPtr.Zero)
12     {
13         Marshal.FreeHGlobal(buffer);
14
15     }
16 }
```

You need to inform the Garbage Collector of your unmanaged memory usage so it can determine when to collect memory.

Which code fragments should you add to your code? (Each correct answer presents part of the solution. Choose two.)

Choose the correct answers

Add this code at Line 06:

GC.Collect();

Add this code at Line 14:

GC.ReRegisterForFinalize(buffer);

Add this code at Line 06:

GC.AddMemoryPressure(bufferSize);

Add this code at Line 06:

GC.SuppressFinalize(buffer);

Add this code at Line 14:

GC.RemoveMemoryPressure(bufferSize);



Add this code at Line 14:

```
GC.KeepAlive(buffer);
```

1.1.86.1 Correction

Add this code at Line 14:



```
GC.RemoveMemoryPressure(bufferSize);
```

Add this code at Line 14:



```
GC.ReRegisterForFinalize(buffer);
```

Add this code at Line 06:



```
GC.AddMemoryPressure(bufferSize);
```

Explanation

To correctly inform the Garbage Collector that you have allocated a large amount of unmanaged memory, you should add this line of code to Line 06:

```
GC.AddMemoryPressure(bufferSize);
```

To correctly inform the Garbage Collector that you have freed a large amount of unmanaged memory, you should add this line of code to Line 14:

```
GC.RemoveMemoryPressure(bufferSize);
```

You should not use this code at Line 06 that forces an immediate collection because it does not include information about your unmanaged allocation:

```
GC.Collect();
```

You should not use this code at Line 14 that incorrectly instructs the Garbage Collector to keep a boxed IntPtr structure in memory, even if it should have been reclaimed due to a lack of managed object references:

```
GC.KeepAlive(buffer);
```

You should not use this code at Line 06 that incorrectly instructs the Garbage Collector to immediately reclaim a boxed IntPtr instead of adding it to the finalizer queue:

```
GC.SuppressFinalize(buffer);
```

You should not use this code at Line 14 that incorrectly instructs the Garbage Collector to add a boxed IntPtr to the finalizer queue before the memory is reclaimed:

```
GC.ReRegisterForFinalize(buffer);
```

1.1.87 Question 87 Objective : Create and Use Types Subobjective : Manage the object life cycle

You are developing an application by using C# and Visual Studio 2015. You have a class with this definition:

```
public class ExplicitlyDisposable : IDisposable
{
    void IDisposable.Dispose()
    {
        System.Diagnostics.Debug.WriteLine("Disposal was called");
    }
    public void DoWork()
    {
        // Do some work that might throw an exception
    }
}
```

You need to ensure that `Dispose()` is called whether the class throws an exception or performs its work successfully.

Which code blocks could you use to satisfy this requirement? (Each correct answer presents a complete solution. Choose two.)

Choose the correct answers

var target = new ExplicitlyDisposable();
try
{
 target.DoWork();
}
finally
{
 if (target != null) target.Dispose();
}

using (var target = new ExplicitlyDisposable())
{
 target.DoWork();
 // Do other work
}

 var target = new ExplicitlyDisposable();
target.DoWork();
// Do some more work...
((IDisposable)target).Dispose();

 var target = new ExplicitlyDisposable();
try
{
 target.DoWork();
 // Do other work
}
finally
{
 if (target != null) ((IDisposable)target).Dispose();
}

 var target = new ExplicitlyDisposable();
try
{
 target.DoWork();
 // Do some more work...
}
finally
{
 ((IDisposable)target).Dispose();
}

1.1.87.1 Correction

```
using (var target = new ExplicitlyDisposable())
{
    target.DoWork();
    // Do other work
}
```

```
✓ var target = new ExplicitlyDisposable();
target.DoWork();
// Do some more work...
((IDisposable)target).Dispose();
```

```
✓ var target = new ExplicitlyDisposable();
try
{
    target.DoWork();
    // Do other work
}
finally
{
    if (target != null) ((IDisposable)target).Dispose();
}
```

When a class implements the IDisposable system interface, it is important to ensure the Dispose() method is called before the object is released to the .NET Garbage Collector.

You could use this code to ensure Dispose() is called:

```
var target = new ExplicitlyDisposable();
try
{
    target.DoWork();
    // Do other work
}
finally
{
    if (target != null) ((IDisposable)target).Dispose();
}
```

The finally block contains code that executes whether the code in the try block completes successfully or terminates with an exception. It is important to check the object reference for null so the code in the finally block does not throw a NullReferenceException that would hide the original error.

This line from the ExplicitlyDisposable class indicates that the Dispose() method is explicitly implemented:

```
void IDisposable.Dispose()
```

This hides the Dispose() method from the compiler unless you first cast the object reference to the IDisposable type.

This line from the `ExplicitlyDisposable` class indicates that the `Dispose()` method is explicitly implemented:

```
void IDisposable.Dispose()
```

This hides the `Dispose()` method from the compiler unless you first cast the object reference to the `IDisposable` type.

As an alternative, you could also use this code to ensure `Dispose()` is called:

```
using (var target = new ExplicitlyDisposable())
{
    target.DoWork();
    // Do other work
}
```

The `using` statement is a shortened syntax that ensures the `Dispose()` method is called when the code block completes successfully or with an exception. This statement also ensures that a `NullReferenceException` is not thrown.

You should not use this code because it does not cast the object reference to the `IDisposable` type before calling the `Dispose()` method:

```
var target = new ExplicitlyDisposable();
try
{
    target.DoWork();
}
finally
{
    if (target != null) target.Dispose();
}
```

You should not use this code block because it does not guard against exceptions:

```
var target = new ExplicitlyDisposable();
target.DoWork();
// Do some more work...
((IDisposable)target).Dispose();
```

You should not use this code block because it does not guard against a `NullReferenceException` in the `finally` block:

```
var target = new ExplicitlyDisposable();
try
{
    target.DoWork();
    // Do some more work...
}
finally
{
    ((IDisposable)target).Dispose();
}
```

1.1.88 Question 88 Objective : Create and Use Types Subobjective : Manage the object life cycle

You are developing an application by using C# and Microsoft Visual Studio 2015. You create a class that should implement the IDisposable interface. This class already contains the following definition:

```
public class Subscriber : IDisposable
{
    private MyApp.Publisher pub;
    private IDisposable ownedChild;
    public Subscriber(Publisher pub) {
        this.pub = pub;
        pub.Published += this.PublishedHandler;
        this.ownedChild =
            new System.Data.SqlClient.SqlConnection();
    }
    ...
}
```

Drag the code fragments to the appropriate location in the class definition given to implement IDisposable in the Subscriber class. You may use items more than once, and you do not have to use each item.

Drag and drop the answers

```
private bool isDisposed = false;
public void Dispose() {
    [REDACTED]
}
~Subscriber() {
    [REDACTED]
}
protected virtual void Dispose(bool disposing) {
    if (isDisposed) return;
    isDisposed = true;
    if (disposing)
    {
        [REDACTED]
        [REDACTED]
    }
}
```

`this.Dispose(true);`

`this.Dispose(false);`

`this.pub.Published -= this.PublishedHandler;`
`this.pub = null;`

`this.ownedChild.Dispose();`
`this.ownedChild = null;`

`this.Dispose(false);`
`GC.SuppressFinalize(this);`

`this.Dispose(true);`
`GC.SuppressFinalize(this);`

`if (this.ownedChild != null) {
 this.ownedChild.Dispose();
 this.ownedChild = null; }`

`if (this.pub != null) {
 this.pub.Published -= this.PublishedHandler;
 this.pub = null; }`

1.1.88.1 Correction

```
private bool isDisposed = false;
public void Dispose() {
    this.Dispose(true);
    GC.SuppressFinalize(this);
}

~Subscriber() {
    this.Dispose(false);
}

protected virtual void Dispose(bool disposing) {
    if (isDisposed) return;
    isDisposed = true;
    if (disposing)
    {
        if (this.pub != null) {
            this.pub.Published -= this.PublishedHandler;
            this.pub = null; }
        if (this.ownedChild != null) {
            this.ownedChild.Dispose();
            this.ownedChild = null; }
    }
}
```

this.Dispose(true);	this.Dispose(false);
this.pub.Published -= this.PublishedHandler; this.pub = null;	this.ownedChild.Dispose(); this.ownedChild = null;
this.Dispose(false); GC.SuppressFinalize(this);	this.Dispose(true); GC.SuppressFinalize(this);
if (this.ownedChild != null) { this.ownedChild.Dispose(); ...}	if (this.pub != null) { this.pub.Published -= this.PublishedHandler; ...}

^ Explanation

Because the Garbage Collector runs infrequently on a difficult-to-predict basis, developers cannot rely on finalizers (or destructors) to release critical resources in a more-timely manner. The `IDisposable` interface declares a single method, `Dispose()`, to provide consistency among system classes and custom classes.

Even though the basic purpose of the `Dispose()` method is to release resources, there are additional expectations that have resulted in a consistent Disposal Pattern with which it is often used, sometimes with slight modification.

The first requirement is that the cleanup code must be called by the finalizer because the system cannot guarantee that a developer calls the `Dispose()` method. The Garbage Collector does not guarantee that it will collect unused resources in any particular order. Therefore, your disposal code should only refer to managed objects if it is called explicitly from the `Dispose()` method. You need to use the overload that declares a Boolean parameter, which is generally named `disposing`. The argument passed to this parameter should be `true` if the disposal code is explicitly invoked by the `Dispose()` method, and `false` if it is implicitly invoked through the finalizer. To make inheritance easier, this overload is generally declared as a protected void method.

There is another expectation (but not a requirement) for classes that implement `IDisposable`. The instance should throw an `ObjectDisposedException` if any other method or property is used after disposal. To satisfy this expectation, the Disposal Pattern includes a private Boolean variable to track whether the object has already been disposed. The exception to this is the `Dispose()` method itself, which should not throw an exception if invoked more than once.

There are many different kinds of cleanup that might be programmed into the disposal routine. This example shows two. Under most circumstances, an object that defines event handlers in code should remove the handlers in the disposal routine. Also, if an object instantiates a long-lived child object that implements IDisposable, the child object's Dispose() method should be called in the parent object's disposal routine. In both of these examples, because they involve managed code, the disposal should be called from Dispose(), but not the finalizer.

1.1.89 Question 89 Objective : Create and Use Types Subobjective : Manage the object life cycle

Example: WeakReference

You are developing an application by using C# 5.0 and Visual Studio 2012. You create a class that includes this code: (Line numbers are included for reference purposes only.)

```
01 public class MyClass
02 {
03     private IDisposable dependentObject;
04     public MyClass (IDisposable dependency)
05     {
06         this.dependentObject = dependency;
07         ...
08     }
09 }
10 ...
11
12 }
```

You need to change this code so that the object reference in MyClass does not keep the Garbage Collector from reclaiming the object.

What should you do? (Each correct answer presents part of the solution. Choose all that apply.)

Choose the correct answers

Change Line 01 to read:

```
public class MyClass : IDisposable
```

Add code at Line 11 to read:

```
public void Dispose()
{
    this.dependentObject = null;
}
```

Change Line 06 to read:

```
this.dependentObject = new WeakReference(dependency);
```

Change Line 03 to read:

```
private WeakReference dependentObject;
```

Change Line 06 to read:

```
using (this.dependentObject = dependency)
```

```
{
```

Add code at Line 08 to read:

```
}
```

1.1.89.1 Correction

Change Line 06 to read:

`this.dependentObject = new WeakReference(dependency);`

Change Line 03 to read:

`private WeakReference dependentObject;`

Explanation

To ensure that the object reference in `MyClass` does not keep the Garbage Collector from reclaiming the object, you should use the `WeakReference` class. This class exposes a property called `Target` that contains the object reference. It also exposes a property called `IsAlive`, a Boolean value that is true if the `Target` property is usable, and false if the object has been collected.

If a candidate object is referenced (rooted) only by the `Target` property of one or more `WeakReference` objects, the Garbage Collector collects the candidate object.

To use a `WeakReference`, you should change Line 03 to read:

`private WeakReference dependentObject;`

You should also change Line 06 to read:

`this.dependentObject = new WeakReference(dependency);`

You should not change Line 06 to read:

```
using (this.dependentObject = dependency)  
{
```

And you should not add code at Line 08 to read:
}

This code change would result in calling the `Dispose` method on the `dependency` argument to the constructor. Under most circumstances, it would be incorrect for a class to call the `Dispose` method on an object that the class does not instantiate. Additionally, the `dependency` object would be unusable to instances of `MyClass` and to the rest of the application.

You should not change Line 01 to read:

`public class MyClass : IDisposable`

And you should not add code at Line 11 to read:

```
public void Dispose()  
{  
    this.dependentObject = null;  
}
```

Together, these two code changes create a basic implementation of `IDisposable`. The `IDisposable` interface is designed to make it convenient to release resources before the Garbage Collector runs. Implementing `IDisposable` does not permit the Garbage Collector to reclaim objects that still have living references.

1.1.90 Question 90 Objective : Create and Use Types Subobjective : Manage the object life cycle

You are creating a C# application by using Visual Studio 2015. Your application creates unmanaged resources. You have created the code below. (Line numbers are included for reference only.)

```
01 public class Unmanaged: IDisposable
02 {
03
04     public Unmanaged()
05     {
06         this.resource = ... // allocate the resource
07     }
08     public void Dispose()
09     {
10
11     }
12     protected virtual void Dispose(bool disposing)
13     {
14         if (disposing)
15         {
16
17         }
18     }
19 }
20 }
```

You need to complete the code at lines 03, 10, 11, and 17 to properly dispose of the resources.

What should you do?

Create a list in the correct order

Possible steps

- Dispose(true);
- GC.SuppressFinalize(this);
- Dispose(false);
- private Object resource;
- private SafeHandle resource;
- resource.Dispose();
- if (resource != null) resource.Dispose();

Steps in order



1.1.90.1 Correction

Create a list in the correct order

Possible steps

Dispose(false);

private Object resource;

resource.Dispose();

Steps in order

private SafeHandle resource;

Dispose(true);

GC.SuppressFinalize(this);

if (resource != null) resource.Dispose();



Explanation

You should write the following code in order:

- * private SafeHandle resource; for line 03
- * Dispose(true); for line 10
- * GC.SuppressFinalize(this); for line 11
- * if (resource != null) resource.Dispose(); for line 17

The code for line 03 declares resource as a safe handle, which provides support for a finalizer to properly handle disposition of the resource and prevent premature disposition.

You should add Dispose(true); to line 10 to call the Dispose method at line 13, setting the disposing parameter to true. This code will be executed when Dispose at line 08 is called by the finalizer and it is now appropriate to manually dispose of the resource.

GC.SuppressFinalize(this); is required after the call to Dispose at line 10 to tell the garbage collection routines that this resource has already been finalized and disposed and that finalize should not be called on it again.

if (resource != null) resource.Dispose(); is required to manually dispose of the resource. The resource != null check is required in case the Dispose method is called and the resource is no longer defined.

...

You should not use any the following line of code:

```
Dispose(false);
```

Although this line might appear to be the correct answer for line 10, it incorrectly passes a value of false for the disposing parameter. As a result, the resource would not be disposed.

You should not use any the following line of code:

```
private Object resource;
```

Although this line might appear to be the correct answer for line 03, it would incorrectly define the resource as a generic Object. The Object class has a Finalize method that can be overridden. However, it can only be called by the garbage collection process and you cannot control when it is called. The generic Object class does not have a Dispose method.

You should not use any the following line of code:

```
resource.Dispose();
```

Although this might appear to be the correct answer for line 17, it does not first check to see if the resource is null. If the resource has already been disposed, this call will fail and abort the program.

1.1.91 Question 91 Objective : Create and Use Types Subobjective : Manage the object life cycle

You are creating a C# application by using Visual Studio 2015. Your application creates unmanaged resources. You have started the following method:

```
protected virtual void Dispose(bool disposing)
```

You need to complete the method using the Dispose pattern.

For each of the following statements, select Yes if the statement is true or No if the statement is false.

Statements	yes	no
If disposing is false, the method has been called from the finalizer.	<input type="radio"/>	<input type="radio"/>
If disposing is false, it is safe to access unmanaged objects.	<input type="radio"/>	<input type="radio"/>
If disposing is true, it is safe for you to dispose of unmanaged objects.	<input type="radio"/>	<input type="radio"/>
This method should be used to centralize disposition of all unmanaged objects.	<input type="radio"/>	<input type="radio"/>

1.1.91.1 Correction

Statements	yes	no
If disposing is false, the method has been called from the finalizer.	<input checked="" type="radio"/>	<input type="radio"/>
If disposing is false, it is safe to access unmanaged objects.	<input type="radio"/>	<input checked="" type="radio"/>
If disposing is true, it is safe for you to dispose of unmanaged objects.	<input checked="" type="radio"/>	<input type="radio"/>
This method should be used to centralize disposition of all unmanaged objects.	<input checked="" type="radio"/>	<input type="radio"/>

^ Explanation

You should answer Yes for If disposing is false, the method has been called from the finalizer. If the method is called by a finalizer, then disposing is set to false, indicating that the System.Object finalizer invoked the Dispose method, so objects should not be accessed because they might have already been finalized. You cannot control the order in which objects are finalized by the finalizer.

You should answer No for If disposing is false, it is safe to access unmanaged objects. If disposing is false, then this method was called by the finalizer and the resource might no longer be allocated.

You should answer Yes for If disposing is true, it is safe for you to dispose of unmanaged objects. If disposing is set to true, then this method has been called by the IDisposable.Dispose method and it is now safe to dispose of the unmanaged resource being handled by this class.

You should answer Yes for This method should be used to centralize disposition of all unmanaged objects. All disposition of unmanaged objects should occur in a method with this signature. This ensures that unmanaged objects are not released prematurely and that duplicate calls to release the objects are handled correctly.

1.1.92 Question 92 Objective : Create and Use Types Subobjective : Manipulate string

The `PhoneNumber` class must store only the ten digits necessary for North American Numbering Plan (NANP) telephone numbers. Your library must also provide an additional class to format a `PhoneNumber` instance as `(XXX)XXX-XXXX` when using the `String.Format()` static method with any format specifier (such as `{0:G}`) in the format string.

You have already written this code: (Line numbers are included for reference purposes only.)

```
01 public class PhoneNumber
02
03 {
04     private string _digits;
05     public string Digits {
06         get { return this._digits; }
07         set {
08             if (value.Length != 10) throw new ArgumentException();
09             this._digits = value;
10         }
11
12
13 }
14 public class PhoneNumberFormatter
15
16 {
17
18
19 }
```

Choose the correct answers

Add this code at Line 18:

```
public string Format(string format, object arg, IFormatProvider provider)
{
    if (arg is FullPhoneNumber)
    {
        var num = (FullPhoneNumber)arg;
        return "(" + num.Digits.Substring(0, 3) + ")" + num.Digits.Substring(3, 3) +
               "-" + num.Digits.Substring(6, 4);
    }
    else if (string.IsNullOrEmpty(format)) return arg.ToString();
    else return String.Format('{0:' + format + "}', arg);
}
```

Add this code at Line 17:

```
public object GetFormat(type formatType)
{
    if (formatType == typeof(ICustomFormatter)) return this;
    else return null;
}
```

Add this code at Line 11:

 public object GetFormat(type formatType)
{
 if (formatType == typeof(ICustomFormatter)) return this;
 else return null;
}

Add this code at Line 02:

 : ICustomFormatter, IFormatProvider

Add this code at Line 15:

 : ICustomFormatter

Add this code at Line 15:

 : ICustomFormatter, IFormatProvider

Add this code at Line 02:

 : IFormatProvider

Add this code at Line 12:

 public string Format(string format, object arg, IFormatProvider provider)
{
 if (arg is FullPhoneNumber)
 {
 var num = (FullPhoneNumber)arg;
 return "(" + num.Digits.Substring(0, 3) + ")" + num.Digits.Substring(3, 3) +
 "-" + num.Digits.Substring(6, 4);
 }
 else if (string.IsNullOrEmpty(format)) return arg.ToString();
 else return String.Format("{0:" + format + "}", arg);
}

1.1.92.1 Correction

Add this code at Line 15:

: ICustomFormatter, IFormatProvider

Add this code at Line 18:

```
public string Format(string format, object arg, IFormatProvider provider)
{
    if (arg is FullPhoneNumber)
    {
        var num = (FullPhoneNumber)arg;
        return "(" + num.Digits.Substring(0, 3) + ")" + num.Digits.Substring(3, 3) +
               "-" + num.Digits.Substring(6, 4);
    }
    else if (string.IsNullOrEmpty(format)) return arg.ToString();
    else return String.Format("{0:" + format + "}", arg);
}
```

Add this code at Line 17:

```
public object GetFormat(type formatType)
{
    if (formatType == typeof(ICustomFormatter)) return this;
    else return null;
}
```

^ Explanation

When designing classes, it is desirable to separate the business logic from the presentation logic, which is the Separation of Concerns principle. To create formatting logic separate from your data class, you should implement the `IFormatProvider` and `ICustomFormatter` .NET system interfaces. In the sample code, these interfaces should both be implemented in the `PhoneNumberFormatter` class.

The basic process is as follows:

- 1) Call the `String.Format()` method, providing an instance of any class that implements `IFormatProvider`.
- 2) The `String.Format()` method calls the `GetFormat()` method of the `IFormatProvider` instance, passing the `ICustomFormatter` datatype as the argument.
- 3) If the `IFormatProvider` instance can provide an instance of the datatype argument, it returns an instance of that type; otherwise, it returns null. NOTE: The class that implements `IFormatProvider` is generally the same class that implements `ICustomFormatter`, so the `GetFormat()` method includes the line "return this;".
- 4) The `String.Format()` method calls the `Format()` method of the `ICustomFormatter` instance, passing the format string, the object to be formatted, and the `IFormatProvider` instance.
- 5) The `Format()` method of the `ICustomFormatter()` instance returns the string representation of the object to the `String.Format()` method. This string is incorporated into the larger format string of the call to `String.Format()` method.

You should add this code at Line 15 to implement the required interfaces in the PhoneNumberFormatter class:

: ICustomFormatter, IFormatProvider

You should add this code at Line 17 to implement the IFormatProvider.GetFormat() method in the PhoneNumberFormatter class:

```
public object GetFormat(type formatType)
{
    if (formatType == typeof(ICustomFormatter)) return this;
    else return null;
}
```

You should add this code at Line 18 to implement the ICustomFormatter.Format() method in the PhoneNumberFormatter class:

```
public string Format(string format, object arg, IFormatProvider provider)
{
    if (arg is FullPhoneNumber)
    {
        var num = (FullPhoneNumber)arg;
        return "(" + num.Digits.Substring(0, 3) + ")" + num.Digits.Substring(3, 3) +
            "-" + num.Digits.Substring(6, 4);
    }
    else if (string.IsNullOrEmpty(format)) return arg.ToString();
    else return string.Format("{0:" + format + "}", arg);
}
```

To use this code, you could call the String.Format() method as follows:

```
var num = new PhoneNumber {Digits="8115550100"};
var formattedString = String.Format(new PhoneNumberFormatter,
    "{0}", num);
```

You should not add this code at Line 02 that implements IFormatProvider and ICustomFormatter in the PhoneNumber class and violates the requirement to keep the presentation logic in a separate class:

: ICustomFormatter, IFormatProvider

You should not add this code at Line 02 that only implements IFormatProvider in the PhoneNumber class:

: IFormatProvider

You should not add this code at Line 15 that implements ICustomFormatter, but not IFormatProvider in the PhoneNumberFormatter class:

: ICustomFormatter

You should not add this code at Line 11 that implements the `IFormatProvider.GetFormat()` method in the `PhoneNumber` class:

```
public object GetFormat(type formatType)
{
    if (formatType == typeof(ICustomFormatter)) return this;
    else return null;
}
```

You should not add this code at Line 12 that implements the `ICustomFormatter.Format()` method in the `PhoneNumber` class:

```
public string Format(string format, object arg, IFormatProvider provider)
{
    if (arg is FullPhoneNumber)
    {
        var num = (FullPhoneNumber)arg;
        return "(" + num.Digits.Substring(0, 3) + ")" + num.Digits.Substring(3, 3) +
            "-" + num.Digits.Substring(6, 4);
    }
    else if (string.IsNullOrEmpty(format)) return arg.ToString();
    else return String.Format("{0:" + format + "}", arg);
}
```

NOTE: For a simple custom formatting solution that violates the Separation of Concerns principle, see the `IFormattable` interface.

<https://docs.microsoft.com/en-us/dotnet/api/system.iformattable?redirectedfrom=MSDN&view=netframework-4.7.2>

1.1.93 Question 93 Objective : Create and Use Types Subobjective : Manipulate string

You are developing a data-processing application by using C# and Visual Studio 2015. You need to construct a single string as a comma-delimited list of five-character codes. These codes are stored one-per-row in a database. You have this method stub:

```
public string BuildString(System.Data.DbDataReader reader)
{
}
```

Which code block should you use to complete the BuildString() method most efficiently?

Choose the correct answer

`string retVal = "";
while (reader.Read())
 retVal += reader.GetString(0) + ",";
return retVal.Substring(0, retVal.Length - 1);`

`string retVal = "";
while (reader.Read())
 retVal = String.Concat(retVal, reader.GetString(0), ",");
return retVal.Substring(0, retVal.Length - 1);`

`var retVal = new System.Text.StringBuilder();
while (reader.Read())
 retVal.Append(reader.GetString(0) + ",");
retVal.Remove(retVal.Length - 2, 1);
return retVal.ToString();`

`char[] tmp, buffer = new char[5], retVal = new char[0];
while(reader.Read())
{
 tmp = retVal;
 reader.GetChars(0, 0, buffer, 5);
 retVal = new char[tmp.Length + 6];
 tmp.CopyTo(retVal, 0);
 retVal[tmp.Length] = ',';
 buffer.CopyTo(retVal, tmp.Length + 1);
}
return new String(retVal, 1, retVal.Length - 1);`

1.1.93.1 Correction

```
var retVal = new System.Text.StringBuilder();
while (reader.Read())
    retVal.Append(reader.GetString(0) + ",");
retVal.Remove(retVal.Length - 2, 1);
return retVal.ToString();
```

Explanation

You should use this code block:

```
var retVal = new System.Text.StringBuilder();
while (reader.Read())
    retVal.Append(reader.GetString(0) + ",");
retVal.Remove(retVal.Length - 2, 1);
return retVal.ToString();
```

This code block uses an instance of the `StringBuilder` class to assemble the string. The `StringBuilder` class is designed to efficiently allocate memory for a steadily growing string. Because of the reduced complexity compared to using basic string concatenation, the performance difference in both execution time and memory allocation becomes measurable around 10,000 iterations, and noticeable to a user around 50,000 iterations. Even when your application constructs smaller strings in a loop, the reduced memory allocation can help reduce the number of garbage collections.

You should not use this code block that allocates a new temporary character array for each iteration, which causes exponential growth in execution time and memory usage with increased iterations:

```
char[] tmp, buffer = new char[5], retVal = new char[0];
while(reader.Read())
{
    tmp = retVal;
    reader.GetChars(0, 0, buffer, 5);
    retVal = new char[tmp.Length + 6];
    tmp.CopyTo(retVal, 0);
    retVal[tmp.Length] = ',';
    buffer.CopyTo(retVal, tmp.Length + 1);
}
return new String(retVal, 1, retVal.Length - 1);
```

You should not use this code block that uses the `String.Concat()` static method. This method is more convenient than using character arrays, but also causes exponential growth in execution time and memory usage with increased iterations:

```
string retVal = "";
while (reader.Read())
    retVal = String.Concat(retVal, reader.GetString(0), ",");
return retVal.Substring(0, retVal.Length - 1);
```

You should not use this code block that uses the + operator. This technique is almost identical to String.Concat() and causes exponential growth in execution time and memory usage with increased iterations:

```
string retVal = "";
while (reader.Read())
    retVal += reader.GetString(0) + ",";
return retVal.Substring(0, retVal.Length - 1);
```

1.1.94 Question 94 Objective : Create and Use Types Subobjective : Manipulate string

You are developing a C# application using Visual Studio 2015. You need to use a logging class with this public declarations:

```
public class Logger
{
    public Logger(TextWriter logDestination) { ... }
    public Log(string message) { ... }
}
```

You need to store log messages in memory and readable as a string.

Which code block correctly implements this requirement?

Choose the correct answer

var destination = new MemoryStream();
var writer = new System.IO.StreamWriter(destination);
var logger = new Logger(writer);

var destination = TextWriter.Null;
var logger = new Logger(writer);

var destination = new String();
var writer = new System.IO.StreamWriter(destination);
var logger = new Logger(writer);

var destination = new StringBuilder();
var writer = new System.IO.StringWriter(destination);
var logger = new Logger(writer);

1.1.94.1 Correction

```
✓ var destination = new StringBuilder();
var writer = new System.IO.StringWriter(destination);
var logger = new Logger(writer);
```

^ Explanation

You should use this code block to provide a `TextWriter` implementation that stores data as a string in memory:

```
var destination = new StringBuilder();
var writer = new System.IO.StringWriter(destination);
var logger = new Logger(writer);
```

The `StringBuilder` class exposes a `ToString()` method that concatenates all child strings into a single string object. The `StringWriter` class uses the `Append()` method of the `StringBuilder` argument when writing messages (for instance, when using the `WriteLine()` method).

You should not use this code block that passes a string to the `StreamWriter` class:

```
var destination = new String();
var writer = new System.IO.StreamWriter(destination);
var logger = new Logger(writer);
```

The `StreamWriter` constructor overload that accepts a single string argument uses the argument as a file path. The `StreamWriter` instance will not write the stream contents into the string.

You should not use this code block that passes a string to the `StreamWriter` class:

```
var destination = new String();
var writer = new System.IO.StreamWriter(destination);
var logger = new Logger(writer);
```

The `StreamWriter` constructor overload that accepts a single string argument uses the argument as a file path. The `StreamWriter` instance will not write the stream contents into the string.

You should not use this code block that stores written messages in a `MemoryStream`:

```
var destination = new MemoryStream();
var writer = new System.IO.StreamWriter(destination);
var logger = new Logger(writer);
```

A `MemoryStream` instance stores data internally in a byte array buffer, not as a string. This buffer can be accessed through `GetBuffer()`. This buffer can be converted to a string with the use of a class that inherits from `System.Text.Encoding`. For instance, you could use the `GetString()` method of the `System.Text.Encoding.Default` static property or the `System.Text.UnicodeEncoding` class. However, the returned string will likely include null characters ('\0') padding out the string to match the buffer size. Therefore, it requires additional processing to generate the expected string.

You should not use this code block that creates a `TextWriter` instance because it does not actually do anything when used:

```
var destination = TextWriter.Null;
var logger = new Logger(writer);
```

The `TextWriter` class is the abstract base class for a variety of `XXXWriter` classes that all have particular use-case scenarios. This class provides the base functionality that developers can use for stream-based write operations.

<https://docs.microsoft.com/en-us/dotnet/api/system.io.stringwriter?redirectedfrom=MSDN&view=netframework-4.7.2>

1.1.95 Question 95 Objective : Create and Use Types Subobjective : Create and implement a class hierarchy

You are developing an application by using C# and Visual Studio 2015. You are implementing INotifyDataErrorInfo in a class that will be used for data binding.

You have these classes: (Line numbers are for reference only.)

```
01 public class Customer
02 {
03     public decimal CreditLimit { get; set; }
04     public decimal Type { get; set; }
05     public IEnumerable GetErrors()
06     {
07         return new CreditLimitValidator(this);
08     }
09 }
10 public class CreditLimitValidator
11 { }
```

You need to complete the CreditLimitValidator class.

Which code blocks correctly complete the CreditLimitValidator class? (Each correct answer presents part of the solution. Choose all that apply.)

Choose the correct answers

Add to the block at line 11:

```
public void Reset() { // omitted }
```

Change line 10 to read:

```
public class CreditLimitValidator : IEnumerable
```

Add to the block at line 11:

```
public boolean MoveNext() { // omitted }
```

Add to the block at line 11:

```
public object Current { get { // omitted } }
```

 Add to the block at line 11:

```
public IEnumerator GetEnumerator() { // omitted }
```

 Change line 10 to read:

```
public class CreditLimitValidator : IEnumerator
```

1.1.95.1 Correction

Change line 10 to read:



```
public class CreditLimitValidator : IEnumerable
```

Add to the block at line 11:



```
public IEnumerator GetEnumerator() { // omitted }
```

Explanation

You should change line 10 to read:

```
public class CreditLimitValidator : IEnumerable
```

This implements the System.Collections.IEnumerable interface in the CreditLimitValidator class. The IEnumerable interface is used by the foreach loop to facilitate iterating through a forward-only, read-only collection of elements.

You should add to the block at line 11:

```
public IEnumerator GetEnumerator() { // omitted }
```

The GetEnumerator method is the only method declared in the IEnumerable interface. This method returns an instance of IEnumerator. The IEnumerator interface describes a class that exposes the methods necessary to start, stop, reset, and move progressively through a forward-only, read-only collection. Instances of IEnumerator are used by the compiled code that results from using a foreach loop.

You should not change line 10 to read:

```
public class CreditLimitValidator : IEnumerator
```

This implements the System.Collections.IEnumerator interface in the CreditLimitValidator class. Instances of IEnumerator are not intrinsically compatible with the IEnumerable type. The return type of the GetErrors method is declared as IEnumerable on line 05, but the return expression is an instance of CreditLimitValidator. You should conclude from this that CreditLimitValidator must implement IEnumerable.

You should not add to the block at line 11:

```
public object Current { get { // omitted } }
```

This declares a read-only property called `Current` that returns an object. This property could be used in instances of the `IEnumerator` interface to retrieve the current object while iterating through a collection. This property is not relevant to the `IEnumerable` interface.

You should not add to the block at line 11:

```
public boolean MoveNext() { // omitted }
```

This declares a method called `MoveNext`. This method could be used in instances of the `IEnumerator` interface to move to the first, then to each subsequent object while iterating through a collection. This property is not relevant to the `IEnumerable` interface.

You should not add to the block at line 11:

```
public void Reset() { // omitted }
```

This declares a method called `Reset` that restarts an iteration. This method could be used in instances of the `IEnumerator` interface to reset the `Current` property to the first object in a series for any class that is not forward-only. This property is not relevant to the `IEnumerable` interface.

<https://docs.microsoft.com/en-us/dotnet/api/system.collections.ienumerable?redirectedfrom=MSDN&view=netframework-4.7.2>

1.1.96 Question 96 Objective : Create and Use Types Subobjective : Create and implement a class hierarchy

You are creating a C# application by using Visual Studio 2015. You have created the code below:

```
public class Plant
{
    public virtual void Draw()
    {
        DrawPlant();
    }
}
```

You need to implement a class called Tree that has its own Draw method. When a consumer of Tree calls its Draw method, the Draw method for Plant should also be called.

Which code should you write?

Choose the correct answer

`public class Tree: Plant
{
 public override void Draw()
 {
 DrawTree();
 Plant.Draw();
 }
}`

`public class Tree: Plant
{
 public override void Draw()
 {
 DrawTree();
 DrawPlant();
 }
}`

public class Tree: Plant
{
 public override void Draw()
 {
 DrawTree();
 base.Draw();
 }
}

public class Tree: Plant
{
 public override void Draw()
 {
 DrawTree();
 Draw();
 }
}

1.1.96.1 Correction

```
public class Tree: Plant
{
    public override void Draw()
    {
        DrawTree();
        base.Draw();
    }
}
```



Explanation

You should use the code below:

```
public class Tree: Plant
{
    public override void Draw()
    {
        DrawTree();
        base.Draw();
    }
}
```

This code properly uses `base.Draw` to call the `Draw` method from the `Plant` class, which is the base class for `Tree`.

You should not use the code below:

```
public class Tree: Plant
{
    public override void Draw()
    {
        DrawTree();
        Plant.Draw();
    }
}
```

This code incorrectly uses `Plant.Draw()` in an attempt to call the `Draw` method of the `Plant` class. The correct line of code is `base.Draw()` to call the `Draw` method of the base class.

You should not use the code below:

```
public class Tree: Plant
{
    public override void Draw()
    {
        DrawTree();
        DrawPlant();
    }
}
```

This code incorrectly attempts to directly call the `DrawPlant` function of the `Plant` class. However, the requirement is to call the `Draw` method of the `World` class, and the correct line of code would be `base.Draw()`.

You should not use the code below:

```
public class Tree: Plant
{
    public override void Draw()
    {
        DrawTree();
        Draw();
    }
}
```

This code calls the `Draw` method, which calls the `Draw` method of the `Tree` class, resulting in an endless recursive call. The correct line of code is `base.Draw()` to call the `Draw` method of the `Plant` class.

1.1.97 Question 97 Objective : Create and Use Types Subobjective : Create and implement a class hierarchy

You are developing a scheduling application by using C# and Visual Studio 2015. Your application has this type declaration (Line numbers are included for reference purposes only.):

```
01 public enum DaysOfWeek : short  
02 {  
03     None = 0,  
04     Sunday = 1,  
05     Monday = 2,  
06     Tuesday = 4,  
07     Wednesday = 8,  
08     Thursday = 16,  
09     Friday = 32,  
10     Saturday = 64  
11 }
```

In a single instance of the DaysOfWeek type, the application should be able to store all of the days of the week on which an event is held. Additionally, a call to the `ToString()` method should result in a comma-delimited list of day names. Finally, calling the static `Enum.Parse<DayOfWeek>(...)` generic method should be able to convert from a comma-delimited list of day names to an instance of DaysOfWeek.

Which code change should you make to Line 01 to allow the DaysOfWeek enumeration to satisfy all of these requirements?

Choose the correct answer

[Serializable] public enum DaysOfWeek : short

public enum DaysOfWeek : string

[Flags] public enum DaysOfWeek : short

[DataContract] public enum DaysOfWeek : long

1.1.97.1 Correction



[Flags] public enum DaysOfWeek : short

^ Explanation

To change the enumeration to support the desired `ToString()` behavior, you should annotate the enumeration with the `Flags` attribute:

```
[Flags] public enum DaysOfWeek : short
```

You should not annotate the enumeration with the `Serializable` attribute that indicates to the .NET serialization system that the enumeration may be converted to XML because it does not affect the behavior of the `ToString()` method:

```
[Serializable] public enum DaysOfWeek : short
```

You should not change the base type to `System.String`. In the .NET framework, reference types (such as strings) are not supported as enumeration base types:

```
public enum DaysOfWeek : string
```

You should not use the `DataContract` attribute to annotate the enumeration. This instructs the `DataContractSerializer` how to create an XML (or other format) representation of instances of the annotated class:

```
[DataContract] public enum DaysOfWeek : long
```

The default behavior of an enumeration assumes that each member is a mutually exclusive option. To allow a single instance to store an on or off value for each member of the enumeration, you should manually assign powers of two as the value of each member (for example: 1, 2, 4, 8, 16, ...). The only limit is that the count of enumeration members must be less than or equal to the number of bits in the base type. In the sample code, the base type is `short`, which is a 16-bit integer that is sufficient for the seven days of the week (plus a named zero value) and does not need to be changed.

The static `Enum.Parse<Tenum>(...)` and `Enum.TryParse<Tenum>(...)` generic methods can automatically parse enumeration values from a comma-delimited list of strings. An optional overload to these methods allow for case-insensitive matching. In the sample code, the powers-of-two values for each member are correct.

The default behavior of the `Enum.ToString()` instance method (inherited by all types declared with the C# `enum` keyword) is to convert the given value to a string only if that value is declared as a named member of the enumeration. Using the sample enumeration, if an appointment was scheduled for Tuesdays and Thursdays, the code might appear as follows:

```
var days = DaysOfWeek.Tuesday | DaysOfWeek.Thursday;
```

But the numeric value would be 20 (4 + 16). When the default `ToString()` method runs, it does not see this value in the enumeration (the `Enum.IsDefined()` method would return `false`), so it prints the number 20 as characters.

<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/enum>

1.1.98 Question 98 Objective : Create and Use Types Subobjective : Create types

You are developing a business data library by using C# and Visual Studio 2015. Other developers use this library, and your classes should be built so others can inherit from them. Your classes should follow the Don't Repeat Yourself (DRY) principle. Your class fields must always be correctly initialized. You should not add unnecessary methods to the public signature of your classes.

You have defined this Employee class (Line numbers are included for reference purposes only.):

```
01 public class Employee
02 {
03
04     public Employee(string firstName, string lastName)
05     { _firstName = firstName;
06         _lastName = lastName; }
07     // additional details omitted
08}
```

Other developers create classes that could include this Manager class (Line numbers are included for reference purposes only.):

```
09 public class Manager : Employee
10 {
11     protected Manager() {}
12     public Manager(string firstName, string lastName)
13     : base(firstName, lastName) { }
14     // additional details omitted
15 }
```

Which constructor should you add to Line 03 to meet the requirements and complete the Employee class?

Choose the correct answer

private Employee()
 : this(string.Empty, string.Empty)
{}

private Employee()
{
 this._firstName = string.Empty;
 this._lastName = string.Empty;
}

protected Employee()
 : this(string.Empty, string.Empty)
{}

protected Employee()
{
 this._firstName = string.Empty;
 this._lastName = string.Empty;
}

1.1.98.1 Correction

```
✓ protected Employee()
  : this(string.Empty, string.Empty)
{}
```

Explanation

To satisfy all requirements, you should add this constructor to Line 03:

```
protected Employee()
  : this(string.Empty, string.Empty)
{}
```

To satisfy the Don't Repeat Yourself (DRY) principle, it is important that all initialization is declared in a single constructor routine. In C#, you can reuse constructor logic by using constructor chaining. The `this` keyword is used to chain to other constructors in the current class. The `base` keyword is used to chain to constructors in a base class.

The C# compiler automatically chains all non-chained constructors of derived classes to the parameter-free constructor of the base class. If you do not explicitly declare a parameter-free constructor in any class, the compiler generates one for you.

A simple solution to maintain the DRY principle is to always include a public or protected default (parameter-free) constructor, even when the class would not otherwise require a default constructor. This constructor would then chain to the constructor with the most parameters, passing default values as the arguments.

You should not add this constructor to Line 03 because it violates the DRY principle by duplicating the logic that initializes the private fields of the `Employee` class:

```
protected Employee()
  : this(string.Empty, string.Empty)
{}
```

You should not add this constructor to Line 03 because it uses the `private` keyword to hide the constructor from inherited classes:

```
private Employee()
  : this(string.Empty, string.Empty)
{}
```

You should not add this constructor to Line 03 because it violates the DRY principle by duplicating the logic that initializes the private fields of the `Employee` class and because it uses the `private` keyword to hide the constructor from inherited classes:

```
private Employee()
{
  this._firstName = string.Empty;
  this._lastName = string.Empty;
}
```

1.1.99 Question 99 Objective : Create and Use Types Subobjective : Create types

You are developing an application by using C# and Visual Studio 2015. You have a type with this definition (Line numbers are included for reference purposes only.):

```
01 public struct Point3d
02 {
03
04     public Point3d (double x, double y, double z)
05     {
06         this.X = x;
07         this.Y = y;
08         this.Z = z;
09     }
10 }
```

You need to implement the data members of the struct at Line 03.

Which code block should you use?

Choose the correct answer

public double X;
public double Y;
public double Z;

public double X = 0;
public double Y = 0;
public double Z = 0;

private double _x;
private double _y;
private double _z;
public double X {
 get { return this._x; }
 private set { this._x = value; }
}
public double Y {
 get { return this._y; }
 private set { this._y = value; }
}
public double Z {
 get { return this._z; }
 private set { this._z = value; }
}

public double X { get; private set; }
public double Y { get; private set; }
public double Z { get; private set; }

1.1.99.1 Correction



```
public double X;  
public double Y;  
public double Z;
```

^ Explanation

You should use this code block that declares three fields (or type-level variables):

```
public double X;  
public double Y;  
public double Z;
```

When you declare fields in a struct, the compiler automatically adds a default (or parameter-free) constructor that sets all fields to binary zero. When creating a struct in C#, the compiler requires definite assignment. This means that the compiler must be able to analyze the source code and guarantee that all fields have been initialized before any methods (including property setters and getters) may be called. Because the compiler does not require the use of any constructors when using structs, you must be diligent to ensure that these fields are initialized.

You should not use this code block that uses automatically-implemented properties that do not have backing fields addressable from within the source code. They cannot be assigned before using the property setter to initialize the values:

```
public double X { get; private set; }  
public double Y { get; private set; }  
public double Z { get; private set; }
```

Note: It is possible to use automatically-implemented properties if you chain all of your custom constructors back to the compiler-generated default constructor by using this suffix to your constructor declarations:

You should not use this code block that uses automatically-implemented properties that do not have backing fields addressable from within the source code. They cannot be assigned before using the property setter to initialize the values:

```
public double X { get; private set; }  
public double Y { get; private set; }  
public double Z { get; private set; }
```

Note: It is possible to use automatically-implemented properties if you chain all of your custom constructors back to the compiler-generated default constructor by using this suffix to your constructor declarations:

: this()

You should not use this code block that initializes the fields as part of the declaration. This syntax is not supported within a struct declaration:

```
public double X = 0;  
public double Y = 0;  
public double Z = 0;
```

You should not use this code block that uses manually-implemented properties with a backing field unless you also changed the constructor to initialize the backing field instead of using the property setter:

```
private double _x;  
private double _y;  
private double _z;  
public double X {  
    get { return this._x; }  
    private set { this._x = value; }  
}  
public double Y {  
    get { return this._y; }  
    private set { this._y = value; }  
}  
public double Z {  
    get { return this._z; }  
    private set { this._z = value; }  
}
```

1.1.100 Question 100 Objective : Create and Use Types Subobjective : Create types

You are developing a Microsoft Windows Presentation Foundation (WPF) application by using C# and Microsoft Visual Studio 2015. The standard WPF assembly PresentationFramework.dll defines a class System.Windows.Window that you must derive.

Your derived class must set the DataContext property, which is defined in the Window base class as a System.Object, to an instance of a class that implements INotifyPropertyChanged. The instance might be passed to the constructor, but if not, it should be instantiated in the constructor. Your class must be able to work with any implementation of INotifyPropertyChanged.

Finish the definition for the MainView derived class by dragging the items from the left to the appropriate location on the right. You may use items more than once, and you do not have to use every item.

Drag and drop the answers

```
public class MainView
{
    public MainView()
    {
        this.DataContext = new T();
    }
    public MainView(T viewModel)
    {
        this.DataContext = viewModel;
    }
}
```

where T : class

where T : class, new()

where T : INotifyPropertyChanged

where T : INotifyPropertyChanged, class

where T : INotifyPropertyChanged, new()

where T : new()

<T>

: System.Windows.Window

1.1.100.1 Correction

Drag and drop the answers

```
public class MainView
<T>
: System.Windows.Window
where T : INotifyPropertyChanged, new()
{
    public MainView()
    {
        this.DataContext = new T();
    }
    public MainView(T viewModel)
    {
        this.DataContext = viewModel;
    }
}
```

where T : class	where T : class, new()	where T : INotifyPropertyChanged
where T : INotifyPropertyChanged, class	where T : INotifyPropertyChanged, new()	where T : new()
<T>	: System.Windows.Window	

Explanation

The final code sample should appear as follows:

```
public class MainView
<T>
: System.Windows.Window
where T : INotifyPropertyChanged, new()
{
    public MainView()
    {
        this.DataContext = new T();
    }
    public MainView(T viewModel)
    {
        this.DataContext = viewModel;
    }
}
```

Other constraints available that should not be used in this case:

class - requires the type parameter to be a reference type.

struct - requires the type parameter to be a value type.

<class name> - requires the type parameter to inherit the specified class.

<type parameter> - requires one type parameter to inherit a different type parameter.

There are several requirements for the `MainView` class. Some of these requirements could be implemented using several techniques, others require very specific C# techniques to fulfill.

The first requirement is that the `MainView` class derives the `System.Windows.Window` base class. This class provides the basic functionality for a visual window in WPF. This requirement leads to the following code:

```
public class MainView
    : System.Windows.Window
{
}
```

The next requirement is that the `MainView` constructor must accept an instance of a class that implements the `INotifyPropertyChanged` system interface in the `System.ComponentModel` namespace. This could be implemented by using the following constructor definition:

```
public MainView(INotifyPropertyChanged viewModel)
{
    this.DataContext = viewModel;
}
```

However, the next requirement makes the previous code block unacceptable. The final requirement is that if an instance of a class that implements `INotifyPropertyChanged` is not passed as an argument to the constructor, the `MainView` class must create a new instance. Interface definitions, such as `INotifyPropertyChanged`, cannot specify whether the class has a public constructor. Using generic type definitions satisfies the previous requirement and this requirement.

To use a generic type, you must create a type parameter. The type parameter is a custom name that serves as a placeholder for a data type. Conventionally, the type parameter is named `T` if there is a single type parameter, or prefixed with `T` if there are more than one type parameters. In the same way that value parameters are declared as a comma-delimited list in parentheses, type parameters are declared as a comma-delimited list in angle brackets immediately following the class (or other reference type) name.

The type parameter may be used within the class declaration anywhere a data type is required. When the compiler encounters a generic type used in executable code, it creates a special class that replaces the type parameter with the actual data type used in code. As long as the generic class can use any potential class with the same pattern of code, the executable code can use any data type as the type parameter.

In this example, we still have the requirement that the `DataContext` property must be set to an instance of a class that implements `INotifyPropertyChanged`. When you use generic types, you may add a `where` expression to establish a constraint on the type parameters. To guarantee that the class used implements `INotifyPropertyChanged`, you would add the constraint `where T : INotifyPropertyChanged` to the end of the class declaration after inheritance and interface implementations have been declared.

This changes the class declaration to the following code:

```
public class MainView<T>
    : System.Windows.Window
    where T : INotifyPropertyChanged
{
    public MainView(T viewModel)
    {
        ...
    }
}
```

```
        this.DataContext = viewModel;
    }
}
```

After all of these changes, you still need to add a constructor with the following definition to satisfy the still unfulfilled requirement to create a new instance of the type if one is not passed to the constructor:

```
public MainView()
{
    this.DataContext = new T();
}
```

This code requires that the data type passed as a type parameter has a public default (or parameter-free) constructor. This is a common requirement when using generic syntax, so a special constraint may be added: `new()`. You may declare a comma-separated list of constraints for each type parameter. The `new()` constraint must be the final constraint in the comma-delimited list.

Generics : <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/generics/index>

1.1.101 Question 101 Objective: Create and Use Types Subobjective: Create types

You are writing a C# application by using Visual Studio 2015.

You need to create a data type to hold only one of three possible categories. The data type should meet the following requirements:

- * Each category should be a value.
- * One value must be "Unknown".
- * One value, "Count", should be equal to the total number of categories, and this value should automatically update if a new category is added.

You need to complete the code. Line numbers are included for reference.

Which code should you write? Use the drop-down lists to correctly complete the code.

Choose the correct options

```
01  Categories
02 {
03  = ,
04 Plant,
05 Animal,
06 Mineral,
07 
08 }
```

1.1.101.1 Correction

Choose the correct options

```
01 enum Categories
02 {
03 Unknown = -1,
04 Plant,
05 Animal,
06 Mineral,
07 Count
08 }
```

Explanation

You should choose enum for line 01. An enum is a data structure that allows you to use names as if they were integers. Each name in the enum is assigned an integer, starting with zero, and each additional name is assigned the next higher integer.

You should choose Unknown for the first target in line 03. Unknown needs to be the first declaration because it will be set to -1. This will make Plant = 0, Animal = 1, Mineral = 2, and Count = 3 (which is the total number of categories).

You should choose -1 for the second target in line 03. This sets Unknown to -1 and makes Plant = 0, Animal = 1, Mineral = 2, and Count = 3 (which is the total number of categories).

You should choose Count for line 07. By placing Count as the last item, and setting Unknown to -1, Count will always be the number of categories, even when you add new categories to the list (as long as they are placed between Unknown and Count).

You should not choose class for Line01. A class is a data structure that is used to define data and methods that act on the data. In this case you need an enum, which is a data structure that defines a sequence of integers paired with names.

You should not choose struct for Line01. A struct is a data structure that is used to define data and optionally methods that act on the data. In this case you need an enum, which is a data structure that defines a sequence of integers paired with names.

You should not choose Count for the first target in Line 03. This would set Count = -1, which would not accurately reflect the number of categories. Instead, place Count as the last item. Then by setting Unknown to -1 and as the first item, Count will always be the number of categories, even when you add new categories to the list (as long as they are placed between Unknown and Count).

You should not choose 0 for the second target in Line 03. This sets Unknown to 0 and makes Plant = 1, Animal = 2, Mineral = 3, and Count = 4, which means that count does not represent the number of categories. Instead, Unknown should be set to -1.

You should not choose 1 for the second target in Line 03. This sets Unknown to 1 and makes Plant = 2, Animal = 3, Mineral = 4, and Count = 5, which means that count does not represent the number of categories. Instead, Unknown should be set to -1.

You should not choose Unknown for Line 07. This places Unknown as the last element in the list, but you need Unknown to be set to -1 and Count to be at the end of the list. This will make Plant = 0, Animal = 1, Mineral = 2, and Count = 3 (which is the total number of categories).

1.1.102 Question 102 Objective : Create and Use Types Subobjective : Create types

You are writing a C# application by using Visual Studio 2015. You have created the following function:

```
int CalcArea(int length, int width)
{
    return length * width;
}
```

Which three lines of code can properly call the function? (Each correct answer presents a complete solution. Choose three.)

Choose the correct answers

- CalcArea(20, width: 30);
- CalcArea(width: 30, length: 20);
- CalcArea(length: 20, width: 30);
- CalcArea(length:20, 30);
- CalcArea(width:30, 20);
- CalcArea(30, length:20);

1.1.102.1 Correction



CalcArea(20, width: 30);



CalcArea(width: 30, length: 20);



CalcArea(length: 20, width: 30);

^ Explanation

The following lines of code would properly call the function:

```
CalcArea(length: 20, width: 30);
CalcArea(width: 30, length: 20);
CalcArea(20, width: 30);
```

C# allows the parameters in a function to also be used as named parameters. Because the signature for the CalcArea function is (int length, int width), both length and width can be used as named parameters. The advantage of using named parameters is that they can be called in any order. It is also legal to mix a positional parameter with a named parameter as long as the positional parameter is specified first. The function calls above are equivalent to CalcArea(20, 30).

You should not choose CalcArea(length:20, 30);. The positional parameter must be specified first.

You should not use CalcArea(30, length:20);. This function call would not be allowed because the first parameter is length (which will be set to 30), and attempting to then set the named parameter for length would not be allowed.

You should not use CalcArea(width:30, 20);. In this function the positional parameter is named first, and this is not allowed.

1.1.103 Question 103 Objective : Create and Use Types Subobjective : Create types

You are developing a data transfer library by using C# and Visual Studio 2015. Your application must write a custom binary format to a file. The data to be serialized is contained in an ArrayList of objects of different types. Most of these objects do not share a common base class other than System.Object.

You have this code:

```
public void SerializeObjects (ArrayList list, BinaryWriter writer)
{
    var serializer = new CustomSerializer { Writer = writer };
    foreach (var o in list)
        serializer.Serialize(o);
}
```

You need to write a CustomSerializer class that can serialize each data type to the custom binary format. It will need to use some property getters, and some method return values for the output, depending on the exact object type. You begin the class definition with this code:

```
public class CustomSerializer
{
    public BinaryWriter Writer { get; set; }
    public void SerializeClass1 (Class1 object) { // implementation omitted }
    public void SerializeClass2 (Class2 object) { // implementation omitted }
    // Additional SerializeXXXX methods as needed
}
```

You need to define the public Serialize method.

Which method definition correctly implements the Serialize method to satisfy the requirements?

Choose the correct answer

`public void Serialize(var o)`
 {
 Serialize(o);
 }

`public void Serialize(DynamicObject o)`
 {
 Serialize(o);
 }

`public void Serialize(ExpandoObject o)`
 {
 Serialize(o);
 }

`public void Serialize(dynamic o)`
 {
 Serialize(o);
 }

1.1.103.1 Correction

```
public void Serialize(dynamic o)
{
    Serialize(o);
}
```

Explanation

You should use this method:

```
public void Serialize(dynamic o)
{
    Serialize(o);
}
```

This method uses the dynamic keyword to provide runtime-binding of method calls. The compiler treats the dynamic keyword the same as object when determining the type of input expressions. However, the compiler allows method calls to the object that cannot be verified by static evaluation of the source code. When dynamic expressions are used as parameters to overloaded methods, the compiler does not bind to a specific method. Instead, the compiler uses all available information to determine a candidate list of methods. The .NET Common Language Runtime selects one method from this list when the statement is executed and the exact data type of the dynamic expression is known.

In this example, the Serialize method accepts one parameter of any data type. When the code is evaluated statically at compile time, the dynamic keyword on the parameter is treated as a synonym for object. When the dynamic expression o is used in a method call, the compiler does not bind to any particular overload of the private Serialize methods. At runtime, the data type of the o expression is determined, and only then does the runtime select the correct method from the overloads.

You should not use this method:

```
public void Serialize(var o)
{
    Serialize(o);
}
```

This method incorrectly uses the var keyword on a parameter declaration. The var keyword permits the compiler to determine the data type of a variable when the code is evaluated statically. The var keyword may only be used when a local variable is declared and initialized in a single statement. The var keyword is never valid when used to declare parameters.

You should not use this method:

```
public void Serialize(ExpandoObject o)
{
    Serialize(o);
}
```

This method incorrectly uses the ExpandoObject data type for the parameter. The ExpandoObject class is a special kind of dynamic class that permits developers to use property syntax on the object without declaring actual property get and set blocks. The ExpandoObject type intercepts these method calls and converts the property names into keys to store and lookup values in an internal Dictionary object. The ExpandoObject type is not compatible with existing, conventional classes.

You should not use this method:

```
public void Serialize(DynamicObject o)
{
    Serialize(o);
}
```

This method incorrectly uses the DynamicObject data type for the parameter. The DynamicObject class is a special kind of class that developers may use to create custom dynamic classes similar to ExpandoObject. The ExpandoObject type is not compatible with existing, conventional classes.

1.1.104 Question 104 Objective : Create and Use Types Subobjective : Consume types

You are writing a C# application by using Visual Studio 2015. You are simulating a game of rock/paper/scissors. In each round, each player chooses rock, paper, or scissors. You want to count the number of times each move has been picked by the player.

You define the following enum:

```
enum CategoryIndex
{
    Rock,
    Paper,
    Scissors,
}
```

You define the following array:

```
int[] categories = { 0, 0, 0 };
```

Which block of code should you write?

Choose the correct answer

void AddCategories(CategoryIndex ci)
{
 categories[(int)ci - 1]++;
}

void AddCategories(CategoryIndex ci)
{
 categories[ci - 1]++;
}

void AddCategories(CategoryIndex ci)
{
 categories[(int)ci]++;
}

void AddCategories(CategoryIndex ci)
{
 categories[ci]++;
}

1.1.104.1 Correction

```
void AddCategories(CategoryIndex ci)
{
    categories[(int)ci]++;
}
```

Explanation

You should use the block of code below:

```
void AddCategories(CategoryIndex ci)
{
    categories[(int)ci]++;
}
```

This block of code correctly casts `CategoryIndex`, an enum, to be an integer, so that it can be used as an index for the `categories` array. Because C# is a strongly typed language, you must explicitly cast an enum to the integer data type to use it as an integer.

You should not use the block of code below:

```
void AddCategories(CategoryIndex ci)
{
    categories[ci]++;
}
```

This block of code attempts to use an enum as if it were an integer. Although enum data types are assigned an integer value, an enum data type is not an integer data type. Because C# is a strongly typed language, you must explicitly cast an enum to the integer data type to use it as an integer, as in `(int)ci`.

You should not use the block of code below:

```
void AddCategories(CategoryIndex ci)
{
    categories[(int)ci - 1]++;
}
```

Because the first value defined in the `CategoryIndex` enum is assigned a value of 0, subtracting 1 from this enum value would result in an index of -1, which is an invalid array index. In this case, you can simply cast the `CategoryIndex` as an integer, and then use that value, as in `categories[(int)ci]`.

You should not use the block of code below:

```
void AddCategories(CategoryIndex ci)
{
    categories[ci - 1]++;
}
```

This block of code attempts to use an enum as if it were an integer. Although enum data types are assigned an integer value, an enum data type is not an integer data type. Because C# is a strongly typed language, you must explicitly cast an enum to the integer data type to use it as an integer, as in `(int)ci`. Furthermore, this code would then subtract 1 from the index (assuming it worked), and this could produce an illegal index of -1. The correct expression is `categories[(int)ci]`.

1.1.105 Question 105 Objective : Create and Use Types Subobjective : Consume types

You are writing a C# application by using Visual Studio 2015. You need write a function that will calculate the numeric sum of two numbers that are stored as strings.

Which code should you write?

Choose the correct answer

float SumStrings(string a, string b)
{
 return (float)a + (float)b;
}

float SumStrings(string a, string b)
{
 return float.Parse(a + b);
}

float SumStrings(string a, string b)
{
 return float.Parse(a) + float.Parse(b);
}

float SumStrings(string a, string b)
{
 return a + b;
}

1.1.105.1 Correction

^ Explanation

You should use the block of code below:

```
float SumStrings(string a, string b)
{
    return float.Parse(a) + float.Parse(b);
}
```

This block of code correctly uses the `float.Parse` method to explicitly convert each string to a float before adding them together. The result will be a float that is the sum of the two numbers.

You should not use the block of code below:

```
float SumStrings(string a, string b)
{
    return a + b;
}
```

This block of code does not convert the strings to a float before adding them. This would result in the two strings being concatenated. Additionally, the return would fail because the function is supposed to return a float, but the return expression is evaluated as a string.

You should not use the block of code below:

```
float SumStrings(string a, string b)
{
    return (float)a + (float)b;
}
```

This block of code attempts to cast the string parameters to the float data type. However, there is no cast from string to float. Instead, you must convert the strings to floats by using the `float.Parse` method. The correct expression would be: `return float.Parse(a) + float.Parse(b);`

You should not use the block of code below:

```
float SumStrings(string a, string b)
{
    return float.Parse(a + b);
}
```

This block of code incorrectly places both of the strings inside the parentheses of the `float.Parse` method. As a result, the strings would first be concatenated together, and then an attempt would be made to convert the result to a float. The correct expression would be: `return float.Parse(a) + float.Parse(b);`

1.1.106 Question 106 Objective : Create and Use Types Subobjective : Enforce encapsulation

You are developing a class library by using C# and Microsoft Visual Studio 2015. You want to implement a Product class. This class is used in many other applications.

The Product class should encapsulate all data behind method calls. No other code blocks should have direct access to the storage for these method calls. You have decided to use property declarations to simplify the syntax of these methods.

Complete the Product class implementation by dragging the keywords from the left to the appropriate location on the right. You may use items more than once, and you do not have to use each item.

Drag and drop the answers

```
public class Product
{
     int _id;
     int Id
    {
        get { return _id; }
        set { _id = value; }
    }
     string _firstName;
     string FirstName
    {
        get { return _firstName; }
        set { _firstName = value; }
    }
}
```

internal private protected protected internal public

1.1.106.1 Correction

```
public class Product
{
    [private] int _id;
    [public] int Id
    {
        get { return _id; }
        set { _id = value; }
    }
    [private] string _firstName;
    [public] string FirstName
    {
        get { return _firstName; }
        set { _firstName = value; }
    }
}
```

^ Explanation

You should use this code:

```
public class Product
{
    private int _id;
    public int Id
    {
        get { return _id; }
        set { _id = value; }
    }
    private string _firstName;
    public string FirstName
    {
        get { return _firstName; }
        set { _firstName = value; }
    }
}
```

You can use access modifiers to control which blocks of code may refer to a declaration. The access modifier precedes the declaration that it modifies. You may choose from among these access modifier keywords: public, private, internal, protected, and protected internal.

The public keyword is the least restrictive access modifier. This keyword indicates that any code block that can reference the containing code block may refer to the declaration. For example, if a type declaration includes the public keyword, then any code that references the containing assembly may use the type. If a type member declaration includes the public keyword, any code that can access the type may refer to the type member.

The private keyword is the most restrictive access modifier. This keyword indicates that the only code that can refer to a private declaration must be within the same containing code block. For example, if a nested type declaration includes the private keyword, then only the containing type declaration may directly refer to it. If a type member includes the private keyword, then only the type that declares the member may refer to the type member.

The internal keyword is more restrictive than the public access modifier, but less restrictive than the private access modifier. This keyword indicates that other code within the containing assembly may use the declaration as if it uses the public keyword. However, code outside the containing assembly is restricted as if the declaration uses the private keyword. This is the default access for any declaration that omits access modifier keywords.

The protected keyword is a special access modifier that relates to inheritance. This keyword indicates that the declaration should be restricted as if it uses the private keyword. However, any code block that directly, or indirectly, inherits from the containing type can also refer to the declaration. This keyword is not limited by whether the inheriting type is declared in the containing assembly.

The protected and internal keywords, when used together, declare the union of the protected and internal access modifiers. This access modifier specifies that the declaration is considered public within the containing assembly. Code that inherits from the containing type may also refer to the declaration, even if the inheriting code is in a different assembly.

1.1.107 Question 107 Objective : Debug Application and Implement Security Subobjective : Validate application input

You use Microsoft .NET Framework 4.5 to create an application. You must create a generic class named Builder that must only support the Vehicle type or types derived from Vehicle.

You need to define the class.

Which code segment should you use?

Choose the correct answer

- public class Builder<T,Vehicle> where T : struct {}
- public class Builder {}
- public class Builder<T> where T : Vehicle {}
- public class Builder<T,Vehicle> where T : class {}

1.1.107.1 Correction



```
public class Builder<T> where T : Vehicle {}
```

^ Explanation

You should use this code segment:

```
public class Builder<T> where T : Vehicle {}
```

This class represents a generic Builder class. The angle bracket identifies the name of the type parameter that is used throughout the class definition. In this case, the name of the generic type is T. The where keyword identifies a generic constraint. In this scenario, the constraint ensures that other code can only specify Vehicle (or one of its derived classes) as the type argument for T.

You should not use this code segment:

```
public class Builder<Vehicle> {}
```

Although this class represents a generic class, the angle bracket identifies the name of the type parameter. Therefore, Vehicle in this class does not represent the Vehicle type. Instead, the name Vehicle serves as a placeholder for types that are constructed later. Code can use any type as a type argument to this class because this class does not specify a generic constraint.

You should not use this code segment:

```
public class Builder<T,Vehicle> where T : class {}
```

This class uses two type parameters: T and Vehicle. The angle bracket identifies the name of the type parameter. Therefore, Vehicle in this class does not represent the Vehicle type. Instead, the name Vehicle serves as a placeholder for types that are constructed later. Code can use any class as a type argument to this class because the constraint specifies that the type argument must be a class.

You should not use this code segment:

```
public class Builder<T,Vehicle> where T : struct {}
```

This class uses two type parameters: T and Vehicle. The angle bracket identifies the name of the type parameter. Therefore, Vehicle in this class does not represent the Vehicle type. Instead, the name Vehicle serves as a placeholder for types that are constructed later. Code can use any struct as a type argument to this class because the constraint specifies that the type argument must be a struct.

1.1.108 Question 108 Objective : Debug Application and Implement Security Subobjective : Validate application input

You are creating a C# application by using Visual Studio 2015. You are writing a function to validate email addresses that are entered in your application. You have found the following regular expression by using a Bing search.

`^[a-zA-Z](\.[a-zA-Z_-]{0,})@[a-zA-Z]+\.(a-zA-Z){1,6}\.[a-zA-Z]{2,6}$`

You need to ensure that this regular expression will correctly validate an email address.

For each of the following statements, select Yes if the statement is true or No if the statement is false.

Statements	yes	no
The dollar sign (\$) indicates that the search should start at the beginning of the line of input.	<input type="radio"/>	<input type="radio"/>
You must use RegexOptions.IgnoreCase for this expression to work for upper and lower case.	<input type="radio"/>	<input type="radio"/>
This regular expression allows numbers in the top-level domain.	<input type="radio"/>	<input type="radio"/>
This regular expression allows an underline (_) and a period(.) in the part of the email address before the at sign (@).	<input type="radio"/>	<input type="radio"/>
This regular expression requires that the top-level domain must have 3 characters.	<input type="radio"/>	<input type="radio"/>

1.1.108.1 Correction

Statements	yes	no
The dollar sign (\$) indicates that the search should start at the beginning of the line of input.	<input type="radio"/>	<input checked="" type="radio"/>
You must use RegexOptions.IgnoreCase for this expression to work for upper and lower case.	<input checked="" type="radio"/>	<input type="radio"/>
This regular expression allows numbers in the top-level domain.	<input type="radio"/>	<input checked="" type="radio"/>
This regular expression allows an underline (_) and a period(.) in the part of the email address before the at sign (@).	<input checked="" type="radio"/>	<input type="radio"/>
This regular expression requires that the top-level domain must have 3 characters.	<input type="radio"/>	<input checked="" type="radio"/>

Explanation

You should answer No to The dollar sign (\$) indicates that the search should start at the beginning of the line of input. The dollar sign (\$) in a regular expression indicates that the search should stop at the end of the item, not at the beginning. The caret (^) indicates that the search should start at the beginning.

You should answer Yes to You must use RegexOptions.IgnoreCase for this expression to work for upper and lowercase. The current regex statement uses the a-z construct to find text characters. However, this will only find lowercase characters. To insure that uppercase characters are also included, you must use the RegexOptions.IgnoreCase enumeration value. If the regex construct had been a-zA-z, then this would not be necessary.

You should answer No to This regular expression allows numbers in the top-level domain. The expression for the top-level domain is [a-z]{2,6}, which allows two to six letters between a and z and no numbers.

You should answer Yes to This regular expression allows an underline _ and a period(.) in the part of the email address before the at sign (@). The expression before the at sign (@) is (\.?|[a-zA-Z_-])\{0,\}. This allows zero or one periods(.) followed by any number of letters, numbers, underscores(_), and dashes(-).

You should answer No to This regular express requires that the top-level domain must have 3 characters. The expression for the top-level domain is [a-zA-Z]{2,6}, which allows two to six letters between a and z.

1.1.109 Question 109 Objective : Debug Application and Implement Security Subobjective : Validate application input

You use Microsoft .NET Framework 4.5 to create an application. You set a breakpoint at the beginning of a method named ProcessRequest. This is the only method in the enclosing class.

You need to configure the debugger to break at the breakpoint the fifth time the ProcessRequest method is called.

What should you do?

Choose the correct answer

- Set the Hit Count property to break when the hit count is five.
- Set the Condition property to count() == 5.
- Set the Filter property to Count=5.
- Set the Filter property to ProcessRequest.Count() == 5.

1.1.109.1 Correction



Set the Hit Count property to break when the hit count is five.

^ Explanation

You should set the Hit Count property to break when the hit count is five. The Hit Count property tells the debugger when it should break at a breakpoint. By setting it to the number five, you allow it to break only at the fifth time the ProcessRequest method is called.

You should not set the Filter property to Count=5. The Filter property allows you to configure the debugger to break only at certain processes and threads. You cannot use the Filter property to tell the debugger to break after a specific number of times a method is called.

You should not set the Condition property to count() == 5. The Condition property allows you to configure the debugger to break when a certain condition is met. The count() == 5 expression tells the debugger to break when a method named count returns a value of 5. This does not cause the debugger to break the fifth time the ProcessRequest method is called unless the count method keeps track of the number of times the ProcessMethod is called. In this scenario, however, the ProcessRequest method is the only one that exists.

You should not set the Filter property to ProcessRequest.Count() == 5. The Filter property allows you to configure the debugger to break only at certain processes and threads. You cannot use the Filter property to tell the debugger to break after a specific number of times a method is called.

1.1.110 Question 110 Objective : Debug Application and Implement Security Subobjective : Validate application input

You use Microsoft .NET Framework 4.5 to create a multi-threaded application. You are debugging the application. You set a breakpoint at a method named ImportImage.

You need to ensure that the breakpoint is hit only when the thread's name is Thread1.

Which two debugger properties allow you to accomplish your goal?

Choose the correct answer

- Breakpoint Location or Breakpoint Filter
- Breakpoint Filter or Breakpoint Condition
- Breakpoint Hit Count or Breakpoint Location
- Breakpoint Condition or Breakpoint Hit Count

1.1.110.1 Correction



Breakpoint Filter or Breakpoint Condition

^ Explanation

You should set the Breakpoint Filter property. The Filter property allows you to configure the debugger to break only at certain processes and threads. In this Breakpoint Filter dialog box, you would specify ThreadName="Thread1".

Alternatively, you can set the Breakpoint Condition property. This property allows you to configure the debugger to break when a certain condition is met. For example, you can configure the debugger to break when a variable is equal to a certain value. In this scenario, you can set the condition to Thread.CurrentThread.Name == "Thread1".

You should not set the Breakpoint Location property. This property specifies the line number and character where the breakpoint should be hit. In this scenario, the breakpoint already exists, which means that the Breakpoint Location property is already set.

You should not set the Breakpoint Hit Count property. This property tells the debugger when it should break at a breakpoint.

1.1.111 Question 111 Objective : Debug Application and Implement Security Subobjective : Debug an application

You use Microsoft .NET Framework 4.5 to create an application. This code exists:

```
private void GenerateImage(Chart chart)
{
    if (chart == null) return;
    using (FileStream stream = File.OpenWrite("Chart.img"))
    {
        chart.SaveImage(stream);
    }
}
```

Before you release the application, you need to ensure that the chart instance is never set to a null reference. You want to be notified with a message box if it is. For the release version of the application, the message box should not display, and execution should continue uninterrupted.

You need to make changes to the code to accomplish your goal.

What should you do?

Choose the correct answer

Add this statement to the beginning of the method:

Debug.Assert(chart != null);

Remove the first line of code from the method.

Add this statement to the beginning of the method:

Trace.Assert(chart != null);

Replace the first line of code with this statement:

if (chart == null) throw new ArgumentNullException();

1.1.111.1 Correction

Add this statement to the beginning of the method:



```
Debug.Assert(chart != null);
```

Explanation

You should add this statement to the beginning of the method:

```
Debug.Assert(chart != null);
```

The Assert method of the Debug class tests whether the specified condition is true. If it is false, it halts execution and displays a message box. However, this only happens in a debug configuration. In a release configuration, the Assert method of the Debug class has no effect.

Replace the first line of code with this statement:

```
if (chart == null) throw new ArgumentNullException();
```

This code throws an exception, which occurs during release and debug configurations. This would not meet the requirement to allow the application to execute uninterrupted.

Add this statement to the beginning of the method:

```
Trace.Assert(chart != null);
```

The Assert method of the Trace class works the same as that of the Debug class, except that it works in a release configuration also. This would not meet the requirement to allow the application to execute uninterrupted in a release configuration.

You should not remove the first line of code from the method. This would cause an exception to be thrown if the chart instance is null when the SaveImage method is called. This would not meet the requirement to allow the application to execute uninterrupted in a release configuration.

1.1.112 Question 112 Objective : Debug Application and Implement Security Subobjective : Debug an application

You are creating a C# application by using Visual Studio 2015. You are using the Spy++ tool for debugging your application.

You need to ensure that you use the tool correctly.

For each of the following statements, select Yes if the statement is true or No if the statement is false.

Statements	yes	no
You must use Spy++ x64 if you are using a 64-bit CPU.	<input type="radio"/>	<input checked="" type="radio"/>
You must keep Visual Studio open while you use Spy++.	<input checked="" type="radio"/>	<input type="radio"/>
Spy++ allows you to view the status of each thread.	<input checked="" type="radio"/>	<input type="radio"/>
Spy++ allows you to view the Window messages handled by each process.	<input checked="" type="radio"/>	<input type="radio"/>

1.1.112.1 Correction

Statements	yes	no
You must use Spy++ x64 if you are using a 64-bit CPU.	<input type="radio"/>	<input checked="" type="radio"/>
You must keep Visual Studio open while you use Spy++.	<input type="radio"/>	<input checked="" type="radio"/>
Spy++ allows you to view the status of each thread.	<input checked="" type="radio"/>	<input type="radio"/>
Spy++ allows you to view the Window messages handled by each process.	<input checked="" type="radio"/>	<input type="radio"/>

^ Explanation

You should answer No for You must use Spy++ x64 if you are using a 64-bit CPU. The 32-bit version of Spy++ can be used to debug on 64-bit CPUs because Visual Studio is a 32-bit product. You would need to use the 64-bit version if the debug output was being redirected to a process running in a 64-bit window.

You should answer No to You must keep Visual Studio open while you use Spy++. Once Spy++ is started, Visual Studio can be closed. Spy++ can even be started from the command line without ever opening Visual Studio.

You should answer Yes to Spy++ allows you to view the status of each thread. One of the core features of Spy++ is the ability to see every thread being used by the application.

You should answer Yes to Spy++ allows you to view the Windows messages handled by each process. One of the core features of Spy++ is the ability to see all Windows messages sent to each process in the application.

1.1.113 Question 113 Objective : Debug Application and Implement Security Subobjective : Perform symmetric and asymmetric encryption

You use Microsoft .NET Framework 4.5 to create an application. The application must authenticate users by using passwords. You do not want to store the plaintext passwords in the database, but you want to store some type of data in the database to represent the passwords. However, there must be no way for the passwords to be decrypted.

You need to choose a class to accomplish your goal.

Which class should you use?

Choose the correct answer

- DSACryptoServiceProvider
- RijndaelManaged
- RSACryptoServiceProvider
- SHA1Managed

1.1.113.1 Correction



SHA1Managed

Explanation

You should use the SHA1Managed class. This class uses the Secure Hash Algorithm (SHA) 1 to compute a unique hash of data. Hashing is one way, and it cannot be reversed. If you hash the passwords and store the hashed data in the database, there would be no way for any application to reverse the hashing process to reveal the passwords. When you authenticate the user, you can hash the user's input and compare it to the hashed data in the database. If the two hash values are equal, then the passwords match and the user is authenticated.

You should not use RSACryptoServiceProvider. This class implements public key or asymmetric encryption by using the Rivest, Shamir, Adleman (RSA) algorithm. With asymmetric encryption, a public key is used to encrypt the data, but a private key is used to decrypt the data. The public key is known to all entities that must be able to encrypt the data. The private key is known only to the entity that should decrypt the data. The public and private key pair cannot be used with other public or private keys. Any application or user that has access to the private key can decrypt the passwords, which violates the requirement in this scenario.

You should not use DSACryptoServiceProvider. Like RSACryptoServiceProvider, this class implements asymmetric encryption. It uses the Digital Signature Algorithm (DSA). Any application or user that has access to the private key can decrypt the passwords, which violates the requirement in this scenario.

You should not use RijndaelManaged. This class implements shared-key, or symmetric encryption. With symmetric encryption, anyone that has access to the shared key can decrypt the passwords, which violates the requirement in this scenario.

1.1.114 Question 114 Objective : Debug Application and Implement Security Subobjective : Perform symmetric and asymmetric encryption

You use Microsoft .NET Framework 4.5 to create application. This code exists:

```
[PrincipalPermission(SecurityAction.Demand, Role="BUILTIN\Users")]
static void SaveData()
{
}

static void Main()
{
    SaveData();
}
```

When the SaveData method is called, an exception of type SecurityException is thrown with the message: Request for principal permission failed.

You ensure that the user running the application is a member of the BUILTIN\Users group.

You need to solve this problem.

What should you do?

Choose the correct answer

Replace SecurityAction.Demand with SecurityAction.Assert.

Add this code before the call to SaveData:

PrincipalPermission p = new PrincipalPermission(null, "BUILTIN\Users");
p.Demand();

Create an instance of WindowsPrincipal and call the IsInRole method.

Add this code before the call to SaveData:

AppDomain.CurrentDomain.SetPrincipalPolicy(PrincipalPolicy.WindowsPrincipal);

1.1.114.1 Correction

Add this code before the call to SaveData:

```
AppDomain.CurrentDomain.SetPrincipalPolicy(PrincipalPolicy.WindowsPrincipal);
```

Explanation

You should add this code before the call to SaveData:

```
AppDomain.CurrentDomain.SetPrincipalPolicy(PrincipalPolicy.WindowsPrincipal);
```

The SetPrincipalPolicy method of the AppDomain class allows you to specify how the principal associated with the application should be created. A principal is simply a user with a set of roles. By specifying PrincipalPolicy.WindowsPrincipal as the parameter, you instruct the runtime to construct a Windows user with roles that represent the user's groups. This principal is then attached to the current thread. All requests for a specific principal examine the current thread for the attached principal. If you do not call the SetPrincipalPolicy method, the runtime automatically creates a generic principal of type GenericPrincipal with no roles assigned.

You should not replace SecurityAction.Demand with SecurityAction.Assert. The Demand value causes the security check to go up the call stack to ensure that all code is granted the permission that is demanded. If all code in the call stack is granted the permission, the demand succeeds. Otherwise it fails. The Assert value causes the security check only on the current stack frame. The check does not go up the call stack. You should use the Assert value when you are sure that your code is safe and that stack walk checks are not important. The problem is that the code is not granted the principal permission because the principal policy is not correctly set.

You should not create an instance of WindowsPrincipal and call the IsInRole method. Internally, the PrincipalPermission attribute does the exact same thing. Therefore, this would not solve the problem.

You should not add this code before the call to SaveData:

```
PrincipalPermission p = new PrincipalPermission(null, "BUILTIN\\Users");
p.Demand();
```

This code performs the same functionality as the PrincipalPermission attribute in this scenario. This code represents an imperative security check, whereas the attribute represents a declarative security check. The problem is that the code is not granted the principal permission because the principal policy is not correctly set.

1.1.115 Question 115 Objective : Create and Use Types Subobjective : Create and implement a class hierarchy

You are developing a class library by using C# and Microsoft Visual Studio 2015. You need to develop a hierarchy of classes to represent different kinds of customers. These classes will be used in other applications.

You need to declare the Customer class. This class serves as the base class for all Customer classes. You should not be able to create new instances of the Customer class. The Customer-derived classes you declare in this library must not be visible to other assemblies. Other developers derive the Customer class in their own assemblies.

Which declaration satisfies these requirements?

Choose the correct answer

- internal abstract class Customer { /* implementation omitted */ }
- internal sealed class Customer { /* implementation omitted */ }
- public abstract class Customer { /* implementation omitted */ }
- public sealed class Customer { /* implementation omitted */ }

1.1.115.1 Correction



```
public abstract class Customer { /* implementation omitted */ }
```

Explanation

You should use this declaration:

```
public abstract class Customer { /* implementation omitted */ }
```

The public access modifier allows all assemblies that take a reference to your library to access the Customer type. The abstract keyword causes the compiler to reject all attempts to create new instances of the Customer class, even in your own code. The only way to create a new instance that uses the Customer type is to use the new keyword with a class that inherits, directly or indirectly, from Customer.

You should not use this declaration:

```
internal abstract class Customer { /* implementation omitted */ }
```

The internal access modifier incorrectly limits the visibility of the Customer class only to your library assembly. No other assemblies have access to the Customer class. This is incorrect, because other developers inherit from this class in their own libraries. You should use the public access modifier to allow other developers to derive Customer.

You should not use this declaration:

```
public sealed class Customer { /* implementation omitted */ }
```

The public access modifier allows all assemblies that take a reference to your library to access the Customer type. The sealed keyword causes the compiler to reject all attempts to derive the Customer class. You should use the abstract keyword instead, because you have Customer-derived classes in your own assembly, and other developers have Customer-derived classes in their assemblies.

You should not use this declaration:

```
internal sealed class Customer { /* implementation omitted */ }
```

The internal access modifier incorrectly limits the visibility of the Customer class only to your library assembly. You should use the public access modifier instead, because other developers must have access to the Customer type to derive from it. The sealed keyword causes the compiler to reject all attempts to derive the Customer class. You should use the abstract keyword instead, because you have Customer-derived classes in your own assembly.

1.1.115.2 Feedback sent

- I have a doubt about one of your answer: could you please answer me?

The question is:

You need to develop a hierarchy of classes to represent different kinds of customers. The classes will be used in other applications.

*You need to declare the Customer class. (...) **The Customer-derived classes you declare in this library must not be visible to other assemblies.***

Your correct answer is:

Public abstract class Customer / implementation omitted */*

My answer was:

Internal abstract class Customer / implementation omitted */*

The internal access modifier limits the visibility of the Customer class only to the library assembly that is what it is required in the question « the class (...) must NOT be visible to other assemblies. ».

Thanks, you for your consideration and correction.

I would appreciate being able to use your site longer than 30 days if you could agree to a commercial gesture.

Regards,

1.1.116 Question 116 Objective : Create and Use Types Subobjective : Create types

You are developing a utility class library by using C# and Microsoft Visual Studio 2015. Your application has performance problems when you use the built-in collection classes in System.Collections.Generic. After thorough analysis, you have decided to implement an Intrusive Linked List. Because this data structure is a pattern that will be used in many classes, you have created the following interface:

```
public interface IIntrusiveList<T>
{
    IIntrusiveList<T> Next { get; set; }
    IIntrusiveList<T> Previous { get; set; }
    T Self { get; }
}
```

The implementation for common methods such as Add(), Remove(), and Count() can use the same implementation for any class that implements IIntrusiveList<T>. You have decided to implement these three methods as extensions to the IIntrusiveList<T> type.

Complete the implementation of the IntrusiveListExtensions class by dragging the keywords on the left to the appropriate location on the right. You may use items more than once, and you do not have to use each item.

Drag and drop the answers

```
public [ ] class IntrusiveListExtensions
{
    public [ ] void Add<T> ([ ] IIntrusiveList<T> node,
                           T item)...
    public [ ] void Remove<T>([ ] IIntrusiveList<T> node)...
    public [ ] long Count<T> ([ ] IIntrusiveList<T> node)...
}
```

[Extension] [base] [static] [T] [this]

1.1.116.1 Correction

Drag and drop the answers

```
public static class IntrusiveListExtensions
{
    public static void Add<T> (this IIntrusiveList<T> node,
        T item)...
    public static void Remove<T>(this IIntrusiveList<T> node)...
    public static long Count<T> (this IIntrusiveList<T> node)...
}
```

[Extension] [base] [static] [T] [this]

Explanation

An extension method is a static method with a more convenient syntax. For instance, you could write a method that extends the string type to determine a file type based on the file extension:

```
public static StringExtensions
{
    public static bool IsImage(string fileName)
    { // implementation omitted }
}
```

You would then invoke this method by using the following method call (assume a variable called `fileName` of type `string`):

```
bool result = StringExtensions.IsImage(fileName);
```

An extension method simplifies the method call to:

```
bool result = fileName.IsImage();
```

To create an extension method, you must create a static method in a static class. The class name is unimportant, as long as it is accessible and in scope (by adding the namespace to the code file with the `using` directive). The data type of the first parameter determines the data type to be extended. The other important detail that turns this method from a standard static method to an extension method is the `this` keyword that prefixes the first parameter:

```
public static StringExtensions
{
    public static bool IsImage(this string fileName)
    { // implementation omitted }
}
```

The compiler automatically annotates the method, the class, and the containing assembly with the `ExtensionAttribute`. This allows the compiler to assign the value on the left of the dot (.) operator to the first parameter of the method.

Because the compiler does not require more elaborate design, such as the use of generics or inheritance, it is not necessary to use specialized keywords such as `base` or generic type parameters like `T`.

1.1.117 Question 117 Objective : Create and Use Types Subobjective : Create types.

You are developing an application by using C# and Visual Studio 2015. You have this class definition (Line numbers are provided for reference only.):

```
01 public class Manager  
02  
03 {  
04     private IList<Employee> _employees = new List<Employee>();  
05  
06 }
```

You want to add and retrieve employee objects from the internal collection by using the EmployeeID with index notation. For example, managerInstance[17] would return the Employee instance that has the ID property set to 17.

Which code block(s) should you use?

Choose the correct answer

At Line 02:

: IEnumerable

At Line 05:

```
public IEnumerator GetEnumerator()  
{ //implementation omitted }
```



At Line 05:

```
public Employee this[int employeeID] {  
    get { //implementation omitted }  
    set { //implementation omitted }  
}
```



At Line 02:

: IQueryable

At Line 05:

```
public IEnumerator GetEnumerator()  
{ //implementation omitted }  
public Type ElementType {  
    get { //implementation omitted }  
}  
public Expression Expression {  
    get { //implementation omitted }  
}  
public IQueryProvider Provider {  
    get { //implementation omitted }  
}
```



At Line 05:

```
public Employee index[int employeeID] {  
    get { //implementation omitted }  
}
```



1.1.117.1 Correction

At Line 05:

```
public Employee this[int employeeID] {  
    get { //implementation omitted }  
    set { //implementation omitted }  
}
```

Explanation

You should use this code block at Line 05 to implement index notation in a custom class:

```
public Employee this[int employeeID] {  
    get { //implementation omitted }  
    set { //implementation omitted }  
}
```

This syntax appears very similar to a standard property. An indexer includes a get block that accepts arguments of the types and names declared between the square brackets ([and]) and returns a value of the type declared before the this keyword. The set block accepts arguments of the types and names declared between the square brackets plus a special argument referenced by the value keyword which is of the type declared before the this keyword.

You should not use this code block at Line 05 that declares a read-only indexer:

```
public Employee index[int employeeID] {  
    get { //implementation omitted }  
}
```

Like property declarations, if an indexer does not contain a set block declaration, the indexer cannot be assigned to. If an indexer does not contain a get block declaration, the indexer cannot be used as an expression.

You should not use these code blocks at Line 02 and 05 that implement the System.IEnumerable interface:

At line 02:

```
: IEnumerable
```

At line 05:

```
public IEnumerator GetEnumerator()  
{ //implementation omitted }
```

The IEnumerable interface is used by the C# compiler to permit a class to be used after the `in` keyword when using a foreach loop. Many of the .NET Framework Class Library (FCL) classes that implement IEnumerable also implement one or more indexer properties. However, the IEnumerable interface and index notation are not technically related.

You should not use these code blocks at Line 02 and 05 that implement the System.Linq.IQueryable interface:

At line 02:

```
: IQueryable
```

At line 05:

```
public IEnumerator GetEnumerator()  
{ //implementation omitted }  
public Type ElementType {  
    get { //implementation omitted }  
}  
public Expression Expression {  
    get { //implementation omitted }  
}  
public IQueryProvider Provider {  
    get { //implementation omitted }  
}
```

The IQueryable interface is part of Language-Integrated Query (LINQ). This interface permits a query provider to process a LINQ query prior to execution. For instance, LINQ to SQL and LINQ to Entities are providers that can translate from a LINQ query to a Structured Query Language (SQL) query prior to execution.

1.1.118 Question 118 Objective : Create and Use Types Subobjective : Create types.

You are developing a product catalog application by using C# and Microsoft Visual Studio 2015. You are programming entity classes.

You have declared this class:

```
public partial class Product : INotifyPropertyChanged
{
    public double ListPrice
    {
        get { return this._listPrice; }
        set
        {
            if (value != this._listPrice)
            {
                this.OnListPriceChanging(value);
                this.OnPropertyChanging("ListPrice");
                this._listPrice = value;
            }
        }
    }
    public event PropertyChangedEventHandler PropertyChanging;
    protected virtual void OnPropertyChanging(string propertyName)
    {
        var ev = this.PropertyChanging;
        if (ev != null)
            this.PropertyChanging(this,
                new PropertyChangedEventArgs(propertyName));
    }
}
```

If another developer defines the `OnListPriceChanging` method in a separate file, you want to call it. If no one defines this method, you want the code to compile successfully.

Which declaration should you use to complete the `Product` class?

Choose the correct answer

- virtual void OnListPriceChanging(double value);
- protected void OnListPriceChanging(double value);
- new void OnListPriceChanging(double value);
- partial void OnListPriceChanging(double value);

1.1.118.1 Correction



`partial void OnListPriceChanging(double value);`

^ Explanation

You should use this declaration:

```
partial void OnListPriceChanging(double value);
```

The partial keyword on the class declaration causes the compiler to combine the declarations from multiple partial class declarations into a single class if they share the same fully-qualified class name. Within a partial class block, a partial method declaration causes the compiler to allow a partial class block to refer to methods that have been declared with a return type, name, and signature but without the implementation block. If another partial class block contains the implementation, then the compiler will use it. If no other partial class blocks implement the method, the compiler automatically implements an empty, or no-op, method block.

You should not use this declaration:

```
protected void OnListPriceChanging(double value);
```

The protected keyword allows other class declarations to access the member. When you declare a method with the protected keyword, you must implement the method block as part of the declaration. The protected keyword cannot be used in the same declaration as the partial keyword.

You should not use this declaration:

```
protected void OnListPriceChanging(double value);
```

The protected keyword allows other class declarations to access the member. When you declare a method with the protected keyword, you must implement the method block as part of the declaration. The protected keyword cannot be used in the same declaration as the partial keyword.

You should not use this declaration:

```
new void OnListPriceChanging(double value);
```

The new keyword enables member hiding. You can use this keyword when you inherit from a class. If the base class does not declare a member with the virtual keyword, your derived class can declare a replacement method by using the new keyword. When you declare a method with the new keyword, you must implement the method block as part of the declaration. The protected keyword cannot be used in the same declaration as the partial keyword.

You should not use this declaration:

```
virtual void OnListPriceChanging(double value);
```

The virtual keyword causes the common language runtime (CLR) to dynamically select the most-specific method declaration. You can create a derived class that declares a method with the override keyword to create a more-specific method in your derived class. When you declare a method with the virtual keyword you must implement the method block as part of the declaration. The virtual keyword cannot be used in the same declaration as the partial keyword.

1.1.119 Question 119 Objective : Create and Use Types Subobjective : Consume types

You are developing an application by using C# and Visual Studio 2015. You have these type declarations (Line numbers are included for reference purposes only.):

```
01 public struct Point2d
02 {
03     public double X { get; set; }
04     public double Y { get; set; }
05     public Point2d(double x, double y) : this()
06     // remaining logic omitted
07 }

08 public struct Point3d
09 {
10     public static explicit operator Point2d(Point3d value)
11     { return new Point2d(value.X, value.Y); }
12     public static implicit operator Point3d(Point2d value)
13     { return new Point3d(value.X, value.Y, 0d); }
14     public double X { get; set; }
15     public double Y { get; set; }
16     public double Z { get; set; }
17     public Point3d(double x, double y, double z) : this()
18     // remaining logic omitted
19 }
```

Which of these are valid statements? (Choose all that apply.)

Choose the correct answers

- Point3d p = new Point2d(3d, 2d);
- Point2d p = new Point3d(5d, 3d, 1d);
- Point2d p = (Point2d) new Point3d(1d, 2d, 3d);
- Point3d p = new Point2d(11d, 13d) as Point3d;
- Point2d p = new Point3d(1d, 1d, 1d) as Point2d;
- Point3d p = (Point3d) new Point2d(3d, 5d);

1.1.119.1 Correction

- Point3d p = new Point2d(3d, 2d);
- Point2d p = new Point3d(5d, 3d, 1d);
- Point2d p = (Point2d) new Point3d(1d, 2d, 3d);
- Point3d p = new Point2d(11d, 13d) as Point3d;
- Point2d p = new Point3d(1d, 1d, 1d) as Point2d;
- Point3d p = (Point3d) new Point2d(3d, 5d);

^ Explanation

These statements are valid statements:

```
Point2d p = (Point2d) new Point3d(1d, 2d, 3d);
Point3d p = new Point2d(3d, 2d);
Point3d p = (Point3d) new Point2d(3d, 5d);
```

To define a custom conversion, your type must declare the conversion method using a specialized syntax:

```
public static [implicit or explicit] operator [return type]([expression type] <identifier>)
```

The sample code given includes one implicit conversion and one explicit conversion.

You can use this code because of the explicit operator that performs a narrowing operation from the more complex Point3d to the simpler Point2d type:

```
Point2d p = (Point2d) new Point3d(1d, 2d, 3d);
```

When you create a custom type, the new type can only be automatically converted to other types through reference conversion (such as converting to a base type), boxing and unboxing conversions (such as converting from a struct to System.Object and back again), wrapping conversions (such as creating a Nullable<T> from a type T), and null type conversions (literally returning a null reference). To extend this behavior, a type may define custom conversions.

Custom conversions can be defined in one of two forms. An implicit conversion allows the compiler to assign an expression of one type to a variable of another type. This is similar to what is commonly termed a widening conversion, such as assigning a 32-bit signed integer to a 64-bit signed integer. An explicit conversion requires the developer to explicitly add a cast operator to the expression before assigning the value to a variable. This is similar to what is commonly termed a narrowing conversion, such as assigning a long to an int. In the .NET framework, the system only defines widening conversions implicitly, others are explicit only.

You can use this code because of the implicit operator that performs a widening operation from the Point2d type to the Point3d type:

```
Point3d p = new Point2d(3d, 2d);
```

You can use this code because of the implicit operator that the compiler allows you to use with an explicit cast:

```
Point3d p = (Point3d) new Point2d(3d, 5d);
```

You cannot use this code because there is not an implicit operator for the narrowing operation from the Point3d type to the Point2d type.

```
Point2d p = new Point3d(5d, 3d, 1d);
```

You cannot use either of these lines of code because the as operator does not execute custom conversions:

```
Point2d p = new Point3d(1d, 1d, 1d) as Point2d;  
Point3d p = new Point2d(11d, 13d) as Point3d;
```

The compiler translates the as keyword into code that checks the data type of the left-hand expression before performing the conversion. This code is functionally equivalent to the following:

```
(p1 is Point2d ? (Point2d)p1 : null)
```

1.1.120 Question 120 Objective : Create and Use Types Subobjective : Consume types

You are developing a class library by using C# and Visual Studio 2015. Developers who write Component Object Model (COM)-based applications need to use one of the classes in your library.

Your library includes this code:

```
[ComVisible(true)]
public class PolicyQuoter
{
    public decimal GetAutoQuote(int customerId) { // implementation omitted }
    public decimal GetLifeQuote(int customerId) { // implementation omitted }
    public decimal GetADDQuote(int customerId) { // implementation omitted }
}
```

You need to export only the PolicyQuoter class into a COM Type Library.

What should you do? (Choose all that apply.)

Choose the correct answers

Add this attribute to each method declaration:

```
[Guid("C358E7BE-7516-473C-BB4C-CC8953DAF437")]
```

Implement IUnknown in the PolicyQuoter class

Add this attribute to AssemblyInfo.cs

```
[assembly: Guid("C358E7BE-7516-473C-BB4C-CC8953DAF437")]
```

Add this attribute to AssemblyInfo.cs:

```
[assembly: ComVisible(false)]
```

Run this command after building the assembly:

```
regasm MyAssembly.dll
```

Implement IDispatch in the PolicyQuoter class

Add this attribute to AssemblyInfo.cs:

```
[assembly: ComVisible(true)]
```

Run this command after building the assembly:

```
tlbexp MyAssembly.dll
```

1.1.120.1 Correction

Add this attribute to AssemblyInfo.cs

[assembly: Guid("C358E7BE-7516-473C-BB4C-CC8953DAF437")]

Add this attribute to AssemblyInfo.cs:

[assembly: ComVisible(false)]

Run this command after building the assembly:

tlbexp MyAssembly.dll

Explanation

You should add this attribute to AssemblyInfo.cs:

[assembly: ComVisible(false)]

When the ComVisible attribute is applied to an assembly, the boolean argument is treated as the default for all types in the library. In this example, PolicyQuoter is the only class that should be exported to a COM Type Library. By setting ComVisible to false for the whole assembly, only types that are annotated with [ComVisible(true)] are exported.

The AssemblyInfo.cs file is the conventional file that contains all attributes that apply to the whole assembly.

You should add this attribute to AssemblyInfo.cs:

[assembly: Guid("C358E7BE-7516-473C-BB4C-CC8953DAF437")]

To expose types from a .NET assembly as COM interfaces requires that the assembly is uniquely identified with a Globally Unique ID (GUID). The Guid attribute should be applied to the assembly, and optionally to any type that will be visible to COM. If you do not explicitly annotate COM-visible types with the Guid attribute, the export process generates a Guid for you.

You should run this command after building the assembly:

tlbexp MyAssembly.dll

The Type Library Export utility (tlbexp.exe) is installed with Visual Studio 2010 or as part of the Windows Platform SDK. This utility will create a COM Type Library that exposes managed code to COM. In this

example, it will create a file named MyAssembly.tlb that can be registered with the Microsoft Windows operating system, and then called by a COM application.

You should not add this attribute to AssemblyInfo.cs:

```
[assembly: ComVisible(true)]
```

This attribute instructs tlbexp.exe to export all public classes in the managed assembly. In this example, you only want to export the PolicyQuoter class.

You should not add this attribute to each method declaration:

```
[Guid("C358E7BE-7516-473C-BB4C-CC8953DAF437")]
```

The Guid attribute may only be applied to assemblies and types. The Guid attribute must not be applied to individual methods, or the compiler will generate errors.

You should not run this command after building the assembly:

```
regasm MyAssembly.dll
```

The Assembly Registration Tool (regasm.exe) is used to register managed assemblies with the local operating system as a COM Type Library. In this example, the type library is used by other developers. Registering a type library on your local operating system will not be useful to other developers. The regasm utility does provide a /tlb flag that instructs regasm to first create a type library before registering the assembly.

You should not implement IDispatch in the PolicyQuoter class. This interface allows late-binding and dynamic languages to use COM interfaces. The tlbexp utility ensures that this interface is implemented in the appropriate COM coclasses in the type library.

You should not implement IUnknown in the PolicyQuoter class. This interface supports the point-counting memory management technique used by COM. The tlbexp utility ensures this interface is implemented in the appropriate COM coclasses in the type library.

<https://docs.microsoft.com/en-us/dotnet/framework/tools/tlbexp-exe-type-library-exporter>

<https://docs.microsoft.com/en-us/windows/desktop/midl/com-dcom-and-type-libraries>

<https://docs.microsoft.com/fr-fr/dotnet/framework/interop/marshaling-data-with-com-interop>

1.1.121 Question 121 Objective : Create and Use Types Subobjective : Consume types

You are developing an application by using C# and Microsoft Visual Studio 2015. Your application needs to log messages generated by several libraries. You do not have access to the source for these libraries.

Each message class declares a method called Log that takes zero arguments.

You need to write code that logs the message classes from each of three libraries.

Which code block should you use? (Each correct answer presents a complete solution. Choose two.)

Choose the correct answers

```
public interface ILoggable {  
    void Log();  
}  
 public class Logger {  
    public void LogMessage(ILoggable message)  
    {/* omitted */}  
}
```

```
 public class Logger {  
    public void LogMessage(Library1.MessageClass message)  
    {/* omitted */}  
    public void LogMessage(Library2.MessageClass message)  
    {/* omitted */}  
    public void LogMessage(Library3.MessageClass message)  
    {/* omitted */}  
}
```

```
 public class Logger {  
    public void LogMessage(dynamic message)  
    {/* omitted */}  
}
```

```
 public class Logger {  
    public void LogMessage(var message)  
    {/* omitted */}  
}
```

1.1.121.1 Correction

```
public class Logger {  
    public void LogMessage(Library1.MessageClass message)  
    { /* omitted */}  
    public void LogMessage(Library2.MessageClass message)  
    { /* omitted */}  
    public void LogMessage(Library3.MessageClass message)  
    { /* omitted */}  
}
```

```
public class Logger {  
    public void LogMessage(dynamic message)  
    { /* omitted */}  
}
```

^ Explanation

You should use this code block:

```
public class Logger {  
    public void LogMessage(dynamic message)  
    { /* omitted */}  
}
```

The dynamic keyword can be used in declarations where you know that specific properties and methods are present on objects that cannot be verified by the compiler. In this example, all of the Message classes support the same method signature. If you had access to the source code, you would extract an interface to represent the common method. In this example, you do not have control over the source. The dynamic keyword allows you to write a single method that can call the common method on these unrelated classes.

Or, you could use this code block:

```
public class Logger {  
    public void LogMessage(Library1.MessageClass message)  
    { /* omitted */}  
    public void LogMessage(Library2.MessageClass message)  
    { /* omitted */}  
    public void LogMessage(Library3.MessageClass message)  
    { /* omitted */}  
}
```

You can use overloaded methods to perform similar operations by processing different arguments. The C# compiler allows you to create several methods that share the same name if the count or respective data type of the arguments are unique for each method. In this example, each library declares a unique `MessageClass` type that does not share a common type, except `object`, with the `MessageClass` types from the other libraries. The compiler distinguishes between the methods based on the data type of the passed argument.

You should not use this code block:

```
public interface ILoggable {  
    void Log();  
}  
public class Logger {  
    public void LogMessage(ILoggable message)  
    {/* omitted */}  
}
```

You should not create an interface. An interface is a data type that describes a class or struct that contains specified methods. Multiple interfaces can be declared in a single data type. Declaring interfaces and then implementing interfaces in multiple unrelated classes is a good way to create loosely-coupled code. In this example, you do not have access to the source code for the class libraries. Therefore, you are unable to implement the `ILoggable` interface in the different `MessageClass` types. The C# compiler does not infer interface types from classes that match the signature but do not implement the interface.

You should not use this code block:

```
public class Logger {  
    public void LogMessage(var message)  
    { /* omitted */}  
}
```

This code incorrectly uses the `var` keyword to declare a parameter. The `var` keyword is used to instruct the compiler to infer the data type of a local variable. The compiler requires `var` declarations to include an initialization expression, from which the data type is inferred. The `var` keyword is not valid as part of a parameter declaration.

1.1.122 Question 122 Objective : Create and Use Types Subobjective : Find, execute, and create types at

runtime by using reflection

You are developing an application by using C# and Visual Studio 2015. You are using .NET Reflection to dynamically load an Assembly and instantiate types discovered within it.

You have this code: (Line numbers are for reference purposes only.)

```
01 var assemblyPath = "C:\someAssembly.dll";
02
03 foreach (Type publicType in assembly.ExportedTypes)
04     InstantiateTypeByUsingReflection(publicType); // Defined elsewhere
```

You need to complete the implementation by inserting a statement at line 02.

Which statement correctly completes the implementation?

Choose the correct answer

- var assembly = Assembly.LoadFrom(assemblyPath);
- var assembly = new Assembly {Location = assemblyPath};
- var assembly = Assembly.ReflectionOnlyLoadFrom(assemblyPath);
- var assembly = Assembly.Load(assemblyPath);

1.1.122.1 Correction



```
var assembly = Assembly.LoadFrom(assemblyPath);
```

^ Explanation

You should use this statement at line 02 to complete the implementation:

```
var assembly = Assembly.LoadFrom(assemblyPath);
```

The static `Assembly.LoadFrom` method accepts a file path argument and loads the identified assembly into the current `AppDomain`. You may then use .NET Reflection to discover and instantiate types, and execute methods.

You should not use this statement at line 02 to complete the implementation:

```
var assembly = Assembly.ReflectionOnlyLoadFrom(assemblyPath);
```

The static `Assembly.ReflectionOnlyLoadFrom` method accepts a file path argument and loads the identified assembly into a reflection-only context. This context allows you to use .NET Reflection to discover metadata including type names, member names, and annotations. The reflection-only context is unsuitable for instantiating discovered types because it prevents the execution of any code from the assembly.

You should not use this statement at line 02 to complete the implementation:

```
var assembly = new Assembly {Location = assemblyPath};
```

The `Assembly` class constructor has internal access and cannot be called from your code. You should use one of the static methods that can discover and load assemblies.

1.1.123 Question 123 Objective : Create and Use Types Subobjective : Enforce encapsulation.

You are developing a library by using C# and Microsoft Visual Studio 2015. Your library handles customer data.

Your library requires a Customer class. This class must include a property called Name. The value of this property should not be stored when the value is either null or an empty string, except when it is initialized. Applications that use the Customer class should not be able to take a reference to the variable, and they should not be able to bypass the validation logic.

Complete the Name property declaration by dragging the code fragments from the list on the left to the appropriate location on the right. You may use items more than once, and you do not have to use each item.

Drag and drop the answers

```
public class Customer
{
    private string Name = "";
    public string name
    {
        get
        {
            return this.name;
        }
        set
        {
            if (string.IsNullOrEmpty(name))
                throw new ArgumentException();
            this.name = value;
        }
    }
}
```

Name name value

1.1.123.1 Correction

```
public class Customer
{
    private string name = "";
    public string Name
    {
        get
        {
            return this.name;
        }
        set
        {
            if (string.IsNullOrEmpty(value))
                throw new ArgumentException();
            this._name = value;
        }
    }
    #region implementation omitted
}
```

Name _name value

^ Explanation

Your code should look like this:

```
public class Customer
{
    private string _name = "";
    public string Name
    {
        get
        {
            return this._name;
        }
        set
        {
            if (string.IsNullOrEmpty(value))
                throw new ArgumentException();
            this._name = value;
        }
    }
    #region implementation omitted
    #endregion
}
```

A property is a class member that encapsulates data by declaring two special methods that are invoked by using variable-assignment and variable-reference syntax. This allows you to centralize validation and dynamic calculation logic. This also prevents other code from taking a reference to your data, because it only obtains a reference to a method instead of a variable.

A basic property declaration requires a data type, a property name, a get block if the property is readable, and a set block if the property is writable. For example, the most simple property declaration - an auto-property - is:

```
public class Customer
{
    public string Name { get; set; }
}
```

When your property requires custom validation or calculation logic, you should implement the individual get and set blocks. These blocks require a storage location to retrieve and store the data, respectively. You generally use a private variable for storage. The get block is written like a function that accepts no parameters and has a return value of the same type as the property. The set block is written like a function that uses a keyword instead of a named parameter, and does not have a return value. For example, a basic property pattern is:

```
public class Customer
{
    private string _name;
    public string Name
    {
        get
        {
            return this._name;
        }
        set
        {
            this._name = value;
        }
    }
}
```

It is common for a set block to error-check the value argument before storing it in the variable. The only way to indicate a validation error is to throw an exception because a set block may not return a value. You could use the `ArgumentException` class to indicate an invalid value. In the example, the value must not be stored if it is null or empty. If you implement this error check and add code to initialize the variable, you create the complete declaration listed above.

1.1.124 Question 124 Objective : Create and Use Types Subobjective : Enforce encapsulation.

You are developing a media transcoding library by using C# and Microsoft Visual Studio 2015. You want to declare a class to encapsulate a single work item.

Your library requires that the WorkItem class contains properties for MediaType and JobID. These properties should be initialized in the constructor. After initialization, code outside of the library should be unable to change the values of these properties. This version of the library does not require custom validation.

You need to begin the WorkItem class declaration and keep the code as simple as possible.

Which code block satisfies these requirements?

Choose the correct answer

public class WorkItem
{
 public int JobID { get; internal set; }
 public string MediaType { get; internal set; }
}

public class WorkItem
{
 internal int JobID { public get; set; }
 internal string MediaType { public get; set; }
}

internal class WorkItem
{
 public int JobID { get; set; }
 public string MediaType { get; set; }
}

public class WorkItem
{
 public int JobID { internal set; }
 public string MediaType { internal set; }
}

1.1.124.1 Correction

```
public class WorkItem
{
    public int JobID { get; internal set; }
    public string MediaType { get; internal set; }
}
```

You should use this code:

```
public class WorkItem
{
    public int JobID { get; internal set; }
    public string MediaType { get; internal set; }
}
```

This code uses auto-implemented properties declared with the public access modifier. This specifies that any code that can access the WorkItem class may also directly reference the properties. However, the set keywords are declared with the internal access modifier. This indicates that only code in the same assembly as the WorkItem class may directly reference the set block.

You should not use this code:

```
public class WorkItem
{
    internal int JobID { public get; set; }
    internal string MediaType { public get; set; }
}
```

This code declares the auto-implemented properties with the internal access modifier. The get keywords are declared with the less-restrictive public access modifier. You may only use an alternative access modifier on the get or set keyword if it is more restrictive than the access modifier on the whole property.

You should not use this code:

```
internal class WorkItem
{
    public int JobID { get; set; }
    public string MediaType { get; set; }
}
```

This code declares the entire WorkItem class with the internal keyword. This hides the WorkItem class from code outside of its containing assembly. The requirements indicate that only the set blocks of the properties should be this restrictive.

You should not use this code:

```
public class WorkItem
{
    public int JobID { internal set; }
    public string MediaType { internal set; }
}
```

This code declares the set keyword with the internal access modifier and omits the get keyword. You may only declare a more-restrictive access modifier on the get or set block of an auto-implemented property if you declare both the get and set keywords.

1.1.124.2 Question 125 Objective : Debug Applications and Implement Security Subobjective : Perform symmetric and asymmetric encryption.

You use Microsoft .NET Framework 4.5 to create an application for Company A. Company B sends data to Company A through a Web service.

You need to verify that the data received by the Web service was not modified in transit, and that it did indeed come from Company B. Both companies use public key cryptography.

Which key should you use to decrypt the hash of the data?

Choose the correct answer

- The public key from Company A
- The private key from Company A
- The public key from Company B
- The private key from Company B

1.1.124.3 Correction



The public key from Company B

^ Explanation

You should use the public key from Company B. To ensure data integrity, you must verify a digital signature. Company B creates a digital signature by hashing the data, and then encrypting the hash with its private key. It sends the data and the encrypted hash to Company A. Only the corresponding public key can decrypt the hash, which is why you need to use the public key from Company B. When you decrypt the hash, you can rehash the data and compare the two hash values. If they match, you have proof that the data was not modified in transit.

You should not use the public key from Company A. Only the corresponding public key can be used to decrypt the encrypted hash. Because the hash is encrypted with Company B's private key, you need Company B's public key to decrypt the hash.

You should not use the private key from Company B. This key is used to encrypt the hash. You need the corresponding public key to decrypt the hash.

You should not use the private key from Company A. Only the corresponding public key can be used to decrypt the encrypted hash. Because the hash is encrypted with Company B's private key, you need Company B's public key to decrypt the hash.

1.1.125 Question 126 Objective : Debug Applications and Implement Security Subobjective : Perform symmetric and asymmetric encryption.

You are creating a C# Windows Communication Foundation (WCF) client by using Visual Studio 2015. You need to create and install a self-signed SSL certificate and install it on development computer. Your client will use the certificate to encrypt messages.

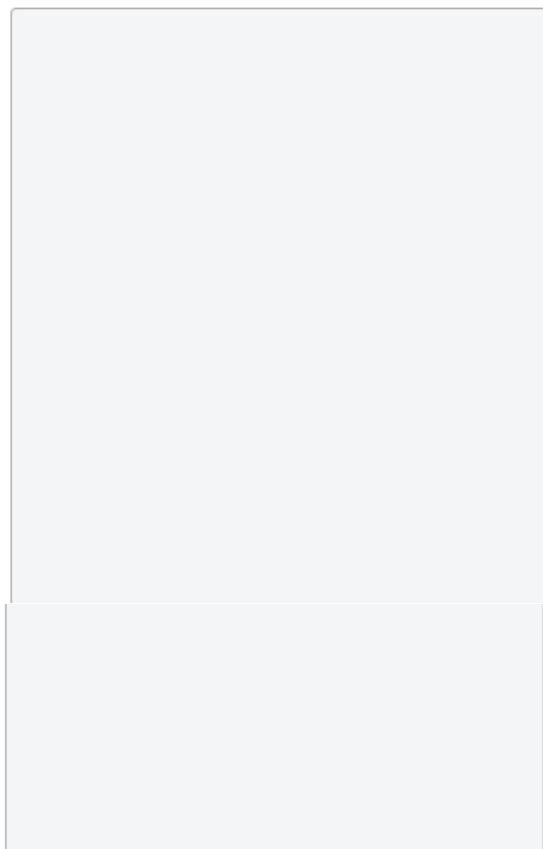
What should you do? Select the four steps that you should perform. Place your selections in the order in which actions must be completed.

Create a list in the correct order

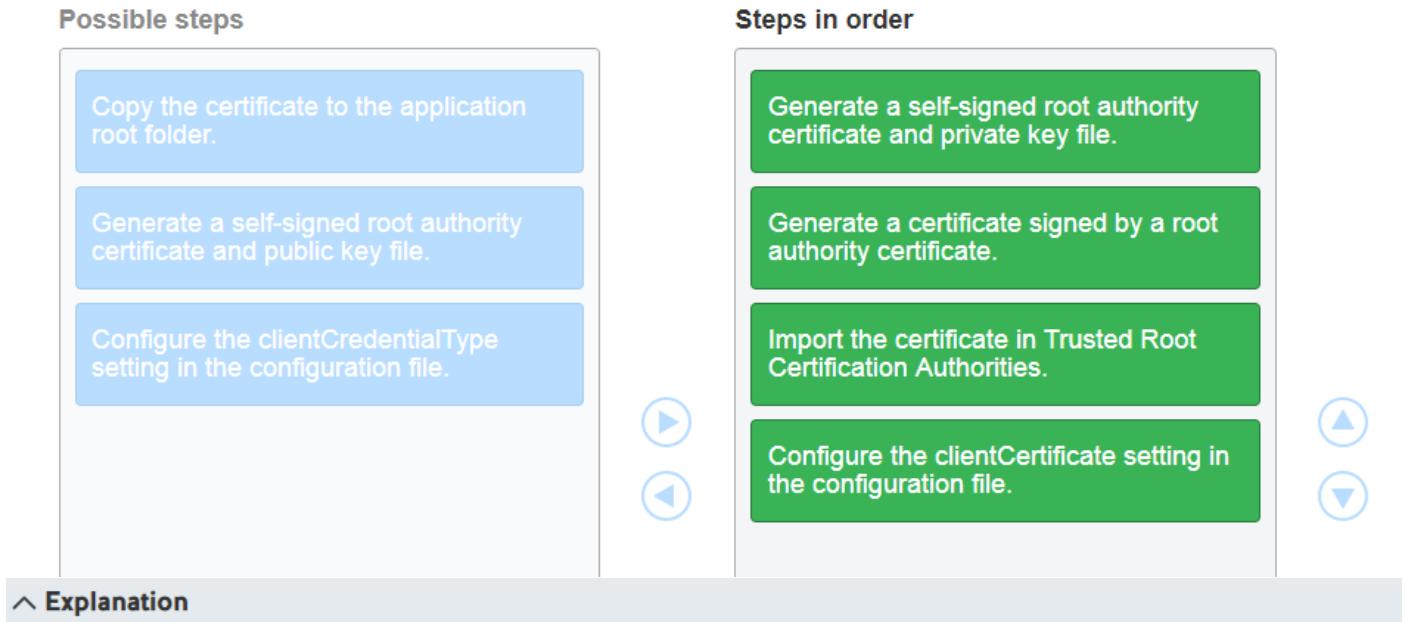
Possible steps

- Copy the certificate to the application root folder.
- Configure the clientCertificate setting in the configuration file.
- Generate a self-signed root authority certificate and private key file.
- Import the certificate in Trusted Root Certification Authorities.
- Generate a self-signed root authority certificate and public key file.
- Configure the clientCredentialType setting in the configuration file.
- Generate a certificate signed by a root authority certificate.

Steps in order



1.1.125.1 Correction



Explanation

You should choose the following steps in order:

- * Generate a self-signed root authority certificate and private key file.
- * Generate a certificate signed by a root authority certificate.
- * Import the certificate in Trusted Root Certification Authorities.
- * Configure the clientCertificate setting in the configuration file.

First, you must generate a self-signed root authority certificate and use the flags to export a private key file. The private key file is then used as input for the second step, which creates another certificate signed by the root authority certificate you created in the first step. The third step is to import the certificate in Trusted Root Certification Authorities. Finally, to use this certificate in your client application, you must configure the clientCertificate setting in the application's configuration file.

The following steps should not be used:

- * Configure the clientCredentialType setting in the configuration file - The clientCredentialsType setting is used to configure a WCF service to use the certificate. Our requirement is to configure a client app, which uses the clientCertificate setting.
- * Copy the certificate to the application root folder - There is no need to copy the certificate into the application root folder.
- * Generate a self-signed root authority certificate and public key file - You always generate a self-signed root authority and a private key. The public key is actually added in step 2.

1.1.126 Question 127 Objective : Debug Applications and Implement Security Subobjective : Manage assemblies

You use Microsoft .NET Framework 4.5 to create a shared assembly. The assembly is not exposed to COM clients.

You need to install the assembly on client computers so that multiple applications can share the assembly.

What should you do?

Choose the correct answer

- Run the Regsvr32.exe command to register the assembly in the Windows Registry.
- Run the Gacutil.exe command to register the assembly in the Global Assembly Cache.
- Run the Regtlib.exe command to register the type library from the assembly.
- Run the Regasm.exe command to register the assembly in the Windows Registry.

1.1.126.1 Correction



Run the Gacutil.exe command to register the assembly in the Global Assembly Cache.

^ Explanation

You should run the Gacutil.exe command to place the assembly in the Global Assembly Cache (GAC). This allows multiple applications to share the assembly.

You should not run the Regsvr32.exe command to register the assembly in the Windows Registry. You would run this command to register Component Object Model (COM) libraries and ActiveX controls. This command does not register an assembly as a shared assembly.

You should not run the Regtlb.exe command to register a type library. Type libraries are used with COM objects. However, in this scenario, the assembly is not exposed to COM clients.

You should not run the Regasm.exe command to register the assembly in the Windows Registry. This registers the assembly as a COM object. However, in this scenario, the assembly is not exposed to COM clients.

1.1.127 Question 128 Objective : Debug Applications and Implement Security Subobjective : Manage assemblies

You use Microsoft .NET Framework 4.5 to create an application. You have three resource files that you want to merge into a single assembly.

You need to merge the resource files.

Which command should you run?

Choose the correct answer

- Regasm.exe
- Ildasm.exe
- Gacutil.exe
- Al.exe

1.1.127.1 Correction



Al.exe

Explanation

You should run the Al.exe command. This is the Assembly Linker tool. It allows you to merge manifest or resource files into a single assembly.

You should not run the Gacutil.exe command. This is the Global Assembly Cache (GAC) tool. It allows you to register an assembly as a shared assembly. It does not allow you to merge resource files into a single assembly.

You should not run the Regasm.exe command. This is the Assembly Registration tool. It allows you to register an assembly as a COM object. It does not allow you to merge resource files into a single assembly.

You should not run the Ildasm.exe command. This is the (Microsoft Intermediate Language) MSIL Disassembly tool. It allows you to access the intermediate language of an assembly. It does not allow you to merge resource files into a single assembly.

1.1.128 Question 129 Objective : Debug Applications and Implement Security Subobjective : Manage assemblies

You use Microsoft .NET Framework 2.0 to create an application. During deployment, you discover that the host server contains .NET Framework 1.1 and .NET Framework 4.0.

You need to configure the application to use .NET Framework 4.0.

Which configuration should you use?

Choose the correct answer

```
<configuration>
  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
        <assemblyIdentity name=".NET Framework" culture="neutral"/>
        <bindingRedirect oldVersion="1.1" newVersion="4.0"/>
      </dependentAssembly>
    </assemblyBinding>
  </runtime>
</configuration>
```



```
<configuration>
  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
        <assemblyIdentity name=".NET Framework" culture="neutral"/>
        <bindingRedirect oldVersion="2.0.50727" newVersion="4.0"/>
      </dependentAssembly>
    </assemblyBinding>
  </runtime>
</configuration>
```



```
<configuration>
  <startup >
    <supportedRuntime version="v1.1" />
    <supportedRuntime version="v2.0" />
  </startup>
</configuration>
```



```
<configuration>
  <startup >
    <supportedRuntime version="v2.0.50727" />
    <supportedRuntime version="v4.0" />
  </startup>
</configuration>
```



1.1.128.1 Correction



```
<configuration>
  <startup >
    <supportedRuntime version="v2.0.50727" />
    <supportedRuntime version="v4.0" />
  </startup>
</configuration>
```

Explanation

You should use this configuration:

```
<configuration>
  <startup >
    <supportedRuntime version="v2.0.50727" />
    <supportedRuntime version="v4.0" />
  </startup>
</configuration>
```

This forces the application to use either .NET Framework 2.0 or .NET Framework 4.0, whichever is present. Because only .NET Framework 1.1 and .NET Framework 4.0 exist on the host server, this configuration forces the application to use .NET Framework 4.0.

```
<configuration>
  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
        <assemblyIdentity name=".NET Framework" culture="neutral"/>
        <bindingRedirect oldVersion="2.0.50727" newVersion="4.0"/>
      </dependentAssembly>
    </assemblyBinding>
  </runtime>
</configuration>
```

This configuration attempts to redirect a referenced assembly named .NET Framework from version 2.0.50727 to version 4.0. It does not force an application to use a specific version of the .NET Framework. You should only use this configuration to redirect references assemblies, not to have the application use a specific version of the .NET Framework.

```
<configuration>
  <startup >
    <supportedRuntime version="v1.1" />
    <supportedRuntime version="v2.0" />
  </startup>
</configuration>
```

This configuration forces the application to use .NET Framework 1.1 or .NET Framework 2.0. However, the application does not run on .NET Framework 1.1, and .NET Framework 2.0 does not exist on the host server.

```
<configuration>
  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
        <assemblyIdentity name=".NET Framework" culture="neutral"/>
        <bindingRedirect oldVersion="1.1" newVersion="4.0"/>
      </dependentAssembly>
    </assemblyBinding>
  </runtime>
</configuration>
```

This configuration attempts to redirect a referenced assembly named .NET Framework from version 1.1 to version 4.0. It does not force an application to use a specific version of the .NET Framework. You should only use this configuration to redirect references assemblies, not to have the application use a specific version of the .NET Framework.

1.1.129 Question 130 Objective : Debug Applications and Implement Security Subobjective : Manage assemblies

You use Microsoft .NET Framework 4.5 to create a shared assembly. You plan to install the assembly in the global assembly cache (GAC).

You need to prepare the assembly to be placed in the GAC.

What should you do?

Choose the correct answer

- Run the Component Registration tool (Regsvr32).
- Run the (Microsoft Intermediate Language) MSIL Disassembly tool (Ildasm.exe).
- Run the Strong Name tool (Sn.exe).
- Run the Assembly Registration tool (Regasm.exe).

1.1.129.1 Correction



Run the Strong Name tool (Sn.exe).

^ Explanation

You should run Sn.exe. The Strong Name tool creates a strong name for an assembly, which is required for the assembly to be placed in the GAC. A strong name is a unique name for an assembly. It consists of the assembly name, version, culture, public key, and digital signature.

You should not run Regasm.exe. The Assembly Registration tool registers the assembly as a COM object. This does not prepare an assembly for being placed in the GAC.

You should not run Regsvr32.exe. The Component Registration tool allows you to register an assembly to be used by COM clients. This does not prepare an assembly for being placed in the GAC.

You should not run Ildasm.exe. The MSIL Disassembly tool allows you to access the intermediate language of an assembly. This does not prepare an assembly for being placed in the GAC.

1.1.130 Question 131 Objective : Debug Applications and Implement Security Subobjective : Manage assemblies

You are creating a C# Windows 8 desktop app by using Visual Studio 2012. You create a project. You want to enable access to the globalization features in the WinRT API from inside Visual Studio.

What should you do? Select the four steps that you should perform. Place your selections in the order in which the action should be performed.

Create a list in the correct order

Possible steps

Set the TargetPlatformVersion setting in the .config file.

Add a reference to System.Messaging.dll.

Set the TargetPlatformVersion setting in the .csproj file.

Add a reference to Windows.winmd.

Add a reference to Windows.Globalization.winmd.

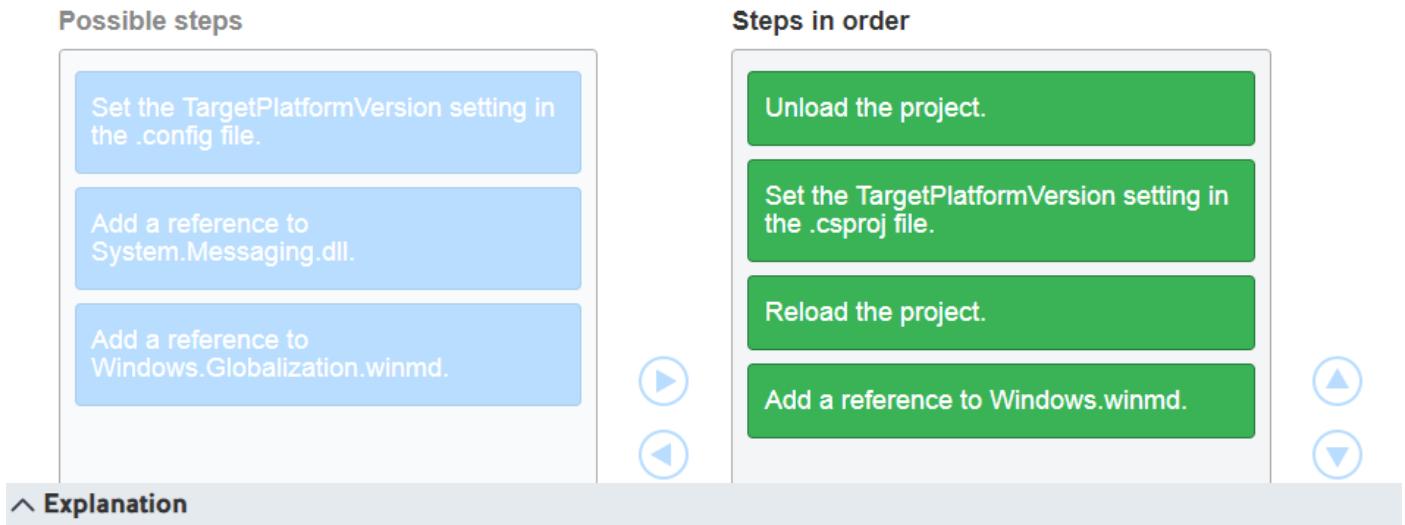
Reload the project.

Unload the project.

Steps in order



1.1.130.1 Correction



Explanation

You should choose the following steps in order:

- * Unload the project.
- * Set the TargetPlatformVersion in the .csproj file.
- * Reload the project.
- * Add a reference to the Windows.winmd.

Because you have already opened the project in Visual Studio, you must unload the project before you will be allowed to edit the csproj file. Second, you must set the TargetPlatformVersion in the csproj file. The third step is to reload the project. The fourth step is to add a reference to Windows.winmd. This file will not show up as a choice until you have performed the first three steps in order.

You should not choose any of the following steps:

- * Add a reference to the Windows.Globalization.winmd file - The Windows.Globalization namespace is available after you add a reference to Windows.winmd.
- * Add a reference to System.Messaging.dll - The dll is used for managing Windows messaging, which is not required by this app.
- * Set the TargetPlatformVersion setting in the .config file - The .config file is not the correct location to set the TargetPlatformVersion setting. This setting is in the csproj file.

1.1.131 Question 132 Objective : Debug Applications and Implement Security Subobjective : Implement diagnostics in an application.

You use Microsoft .NET Framework 4.5 to create an application. This configuration exists:

```
<configuration>
  <system.diagnostics>
    <switches>
      <add name="mySwitch" value="1"/>
    </switches>
    <sources>
      <source name="MySource">
        <listeners>
          <add name="messages" type="System.Diagnostics.XmlWriterTraceListener"
            initializeData="C:\ServiceLog.xml"/>
        </listeners>
      </source>
    </sources>
  </system.diagnostics>
</configuration>
```

This code exists:

```
TraceSwitch switch = new TraceSwitch("mySwitch", null);
if (switch.TraceError)
  Trace.TraceError("Error");
if (switch.TraceWarning)
  Trace.TraceWarning("Warning");
if (switch.TraceInfo)
  Trace.TraceInfo("Info");
  Trace.TraceInfo("Info");
if (switch.TraceVerbose)
  Trace.TraceVerbose("Verbose");
```

You need to determine the output of the previous code.

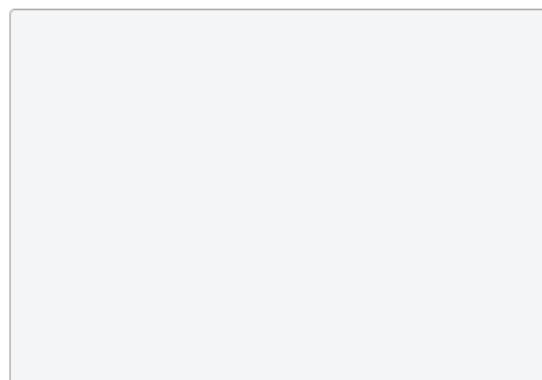
In the list on the right, select the messages that will be written to the trace output. Place your selections in the list on the left in the order in which the messages are written. Place your selections in the list on the left by clicking the items in the list on the right and clicking the arrow button. You can also use the up and down buttons to rearrange items in the list on the left. You may not need to use all of the items from the list on the right.

Create a list in the correct order

Available Messages

- Verbose
- Info
- Warning
- Error

Messages Written



1.1.131.1 Correction

Create a list in the correct order

Available Messages

Verbose

Info

Warning

Messages Written

Error

Explanation

You should conclude that only the message Error is written to the trace output. The code uses a trace switch named mySwitch to determine the trace level. When you create a TraceSwitch instance, it checks the configuration for the value of the switch. In this scenario, the switch is set to value 1. This means that only errors are written to the trace output. The code checks the Trace.TraceError property to determine whether to call the Trace.TraceError method. This property returns true because the switch is set to value 1. If you set the switch to 2, then errors and warnings are written. If you set the switch to 3, errors, warnings, and informational messages are written. If you set the switch to 4, errors, warnings, informational messages, and verbose messages are written.

1.1.132 Question 133 Objective : Debug Applications and Implement Security Subobjective : Implement diagnostics in an application.

You use Microsoft .NET Framework 4.5 to create an application. This configuration exists:

```
<configuration>
  <system.diagnostics>
    <switches>
      <add name="mySwitch" value="Data Access"/>
    </switches>
  </system.diagnostics>
</configuration>
```

You need to programmatically obtain the value of the mySwitch switch in code.

What should you do?

Choose the correct answer

- Create an instance of SourceSwitch and examine its Attributes property.
- Create an instance of TraceSwitch and examine its Attributes property.
- Create a custom class that inherits from Switch and examine its Level property.
- Create a custom class that inherits from Switch and examine its Value property.

1.1.132.1 Correction



Create a custom class that inherits from Switch and examine its Value property.

Explanation

You should create a custom class that inherits from Switch and examine its Value property. The Switch class is a base class for diagnostic switches. Because the value attribute in this scenario is set to Data Access, you cannot use the existing Switch-derived classes to obtain its value. The existing classes are SourceSwitch, which only understands these values (or their numerical equivalents): Critical, Error, Warning, Information, Verbose, Start, Stop, Suspend, Resume, and Transfer; TraceSwitch, which only understands these values (or their numerical equivalents): Off, Error, Warning, Info, Verbose; and BooleanSwitch, which only understands these values (or their numerical equivalents): False, True.

You should not create a custom class that inherits from Switch and examine its Level property. The Level property returns an integer that represents the numeric value of the switch. In this scenario, the value is set to a string, not an integer. Therefore, the Level property would not return the value that is set.

You should not create an instance of TraceSwitch and examine its Attributes property. This property allows you to retrieve custom attribute values from a custom switch that is configured. The TraceSwitch class has no custom attributes, so this property would not return the value that you are trying to obtain.

You should not create an instance of SourceSwitch and examine its Attributes property. This property allows you to retrieve custom attribute values from a custom switch that is configured. The SourceSwitch class has no custom attributes, so this property would not return the value that you are trying to obtain.

1.1.133 Question 134 Objective : Debug Applications and Implement Security Subobjective : Implement diagnostics in an application.

You use Microsoft .NET Framework 4.5 to create an application. This code exists:

```
PerformanceCounter pc = new PerformanceCounter("Page Hits", "Order", false);
```

This code creates a performance counter that monitors the number of hits to an Order page.

You need to increase the number of hits by one.

Which code segment should you use?

Choose the correct answer

- pc.RawValue = pc.NextSample().RawValue;
- pc.NextSample();
- pc.Increment();
- pc.NextValue();

1.1.133.1 Correction



pc.Increment();

^ Explanation

You should call the Increment method. This method increments the value of the performance counter by one.

You should not call the NextValue method of the PerformanceCounter class. This method returns the next value from the performance counter. It does not increase the value of the counter by one.

You should not call the NextSample method of the PerformanceCounter class. It returns an instance of CounterSample instance that represents the next sample of data for the counter. This does not increase the value of the counter by one.

You should not set the RawValue property of the PerformanceCounter class to the RawValue property of a CounterSample instance. A CounterSample instance represents the next sample of data for the counter. This does not increase the value of the counter by one.

1.1.134 Question 135 Objective : Debug Applications and Implement Security Subobjective : Implement diagnostics in an application.

You use Microsoft .NET Framework 4.5 to create an application. You create a performance session and generate the report shown in the graphic exhibit.

What can you conclude about the performance of the application?

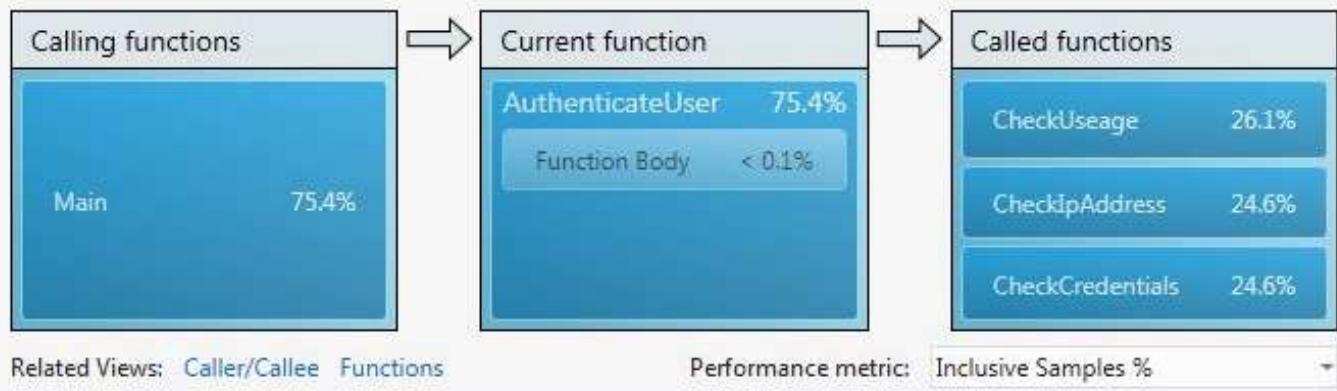
Choose the correct answer

- The Main method has a performance bottleneck, but the AuthenticateUser method does not.
- The CheckUsage, CheckIpAddress, and CheckCredentials methods have performance bottlenecks.
- The AuthenticateUser method has a performance bottleneck, but the CheckUsage, CheckIpAddress, and CheckCredentials methods do not.
- Both the Main method and the AuthenticateUser methods have performance bottlenecks, but the CheckUsage, CheckIpAddress, and CheckCredentials methods do not.

Exhibit

AppCenter.Program.AuthenticateUser()

AppCenter.exe



1.1.134.1 Correction



The CheckUsage, CheckIpAddress, and CheckCredentials methods have performance bottlenecks.

Explanation

You should conclude that the CheckUsage, CheckIpAddress, and CheckCredentials methods have performance bottlenecks. Both the Main method and the AuthenticateUser method have the same inclusive sample percentage, which is 75.4. This is the same as the total inclusive sample percentages for CheckUsage, CheckIpAddress, and CheckCredentials. This means that the Main method only calls AuthenticateUser. The AuthenticateUser method only calls CheckUsage, CheckIpAddress, and CheckCredentials. Therefore, the bottleneck must be in these three methods, or in methods called by these three methods.

You should not conclude that the AuthenticateUser method has a performance bottleneck, but the CheckUsage, CheckIpAddress, and CheckCredentials methods do not. The AutenticateUser method has an inclusive sample percentage that is equal to the total inclusive sample percentages for CheckUsage, CheckIpAddress, and CheckCredentials, which is 75.4. This means that the AuthenticateUser method only calls CheckUsage, CheckIpAddress, and CheckCredentials. Therefore, the bottleneck must be in these three methods, or in methods called by these three methods.

You should not conclude that the Main method has a performance bottleneck, but the AuthenticateUser method does not. Both methods have the same inclusive sample percentage, which is 75.4. This means that the Main method only calls AuthenticateUser.

You should not conclude that both the Main method and the AuthenticateUser methods have performance bottlenecks, but the CheckUsage, CheckIpAddress, and CheckCredentials methods do not. Both methods have the same inclusive sample percentage, which is 75.4. This is the same as the total inclusive sample percentages for CheckUsage, CheckIpAddress, and CheckCredentials. This means that the Main method only calls AuthenticateUser. The AuthenticateUser method only calls CheckUsage, CheckIpAddress, and CheckCredentials. Therefore, the bottleneck must be in these three methods, or in methods called by these three methods.

1.1.135 Question 136 Objective : Debug Applications and Implement Security Subobjective : Implement diagnostics in an application.

You are creating a C# application by using Visual Studio 2015. You are implementing performance counters in the code to monitor performance.

You need to create a timer that allows you to track the amount of time that has passed since the process started.

Which block of code should you write?

Choose the correct answer

CounterCreationDataCollection counter = new CounterCreationDataCollection();
CounterCreationData data = new CounterCreationData();
data.CounterType = PerformanceCounterType.ElapsedTime;
data.CounterName = "Time";
counter.Add(data);

CounterCreationData data = new CounterCreationData();
counter.CounterType = PerformanceCounterType.ElapsedTime;
counter.CounterName = "Time";
CounterCreationDataCollection counter = new CounterCreationDataCollection(data);

CounterCreationDataCollection counter = new CounterCreationDataCollection();
CounterCreationData data = new CounterCreationData();
data.CounterType = PerformanceCounterType.ElapsedTime;
data.CounterName = "Time";
data.Add(counter);

CounterCreationDataCollection counter = new CounterCreationDataCollection();
data = new CounterCreationData();
counter.CounterType = PerformanceCounterType.ElapsedTime;
counter.CounterName = "Time";
counter.Add(data);

1.1.135.1 Correction

^ Explanation

You should choose the block of code below:

```
CounterCreationDataCollection counter = new CounterCreationDataCollection();
CounterCreationData data = new CounterCreationData();
data.CounterType = PerformanceCounterType.ElapsedTime;
data.CounterName = "Time";
counter.Add(data);
```

This code creates a new CounterCreationDataCollection, which is a container for CounterCreationData objects that define each counter. It then creates a CounterCreationData object, sets the parameters CounterType and CounterName, and then adds it to the container. CounterType is correctly set to the type PerformanceCounterType.ElapsedTime.

You should not choose the block of code below:

```
CounterCreationDataCollection counter = new CounterCreationDataCollection();
data = new CounterCreationData();
counter.CounterType = PerformanceCounterType.ElapsedTime;
counter.CounterName = "Time";
counter.Add(data);
```

This block of code incorrectly prefixes CounterType and CounterName with counter, which is a CounterCreationDataCollection object. Instead, each of these objects should be prefixed with data, which is a CounterCreationData object.

You should not choose the block of code below:

```
CounterCreationDataCollection counter = new CounterCreationDataCollection();
CounterCreationData data = new CounterCreationData();
data.CounterType = PerformanceCounterType.ElapsedTime;
data.CounterName = "Time";
data.Add(counter);
```

This block of code incorrectly prefixes Add with data, which is a CounterCreationData object, and attempts to add counter, which is a CounterCreationDataCollection object. The actual use is exactly the opposite. The data object should be added to the counter object.

You should not choose the block of code below:

```
CounterCreationData data = new CounterCreationData();
counter.CounterType = PerformanceCounterType.ElapsedTime;
counter.CounterName = "Time";
CounterCreationDataCollection counter = new CounterCreationDataCollection(data);
```

This code creates a CounterCreationData object, sets its parameters, and then attempts to create a CounterCreationDataCollection object, passing data as a parameter. However, there is no constructor for a CounterCreationDataCollection object that accepts a single CounterCreationData object.

1.1.136 Question 137 Objective : Implement Data Access Subobjective Query and manipulate data and object by using LINQ

You use Microsoft .NET Framework 4.5 to create an application. This class exists:

```
public class Vehicle
{
    public string Make {get; set;}
    public string Model {get; set;}
    public int Year {get; set;}
    public string Vin {get; set;}

    public static List<Vehicle> GetVehicles()
    {
        // Implementation Omitted
    }
}
```

You need to create a query that returns a collection of Vehicle instances where the Make property is set to "Honda".

Which code segment should you use?

Choose the correct answer

- var query = Vehicle.GetVehicles().Where(v => v.Make == "Honda").Select (v => v.Make);
- var query = from v in Vehicle.GetVehicles() where v.Make == "Honda" select v;
- var query = from v in Vehicle.GetVehicles().Where(v => v.Make == "Honda")
select new {Make = v.Make};
- var query = from v in Vehicle.GetVehicles() where v.Make == "Honda" select v.Make;

1.1.136.1 Correction

^ Explanation

You should use this code segment:

```
var query = from v in Vehicle.GetVehicles() where v.Make == "Honda" select v;
```

This code uses a LINQ query expression to select `Vehicle` instances where each instance's `Make` property is set to "Honda".

You should not use this code segment:

```
var query = from v in Vehicle.GetVehicles() where v.Make == "Honda" select v.Make;
```

This code returns a collection of strings with each string set to "Honda".

You should not use this code segment:

```
var query = Vehicle.GetVehicles().Where(v => v.Make == "Honda").  
Select (v => v.Make);
```

This code uses LINQ's `Where` and `Select` methods to return a collection of strings with each string set to "Honda".

You should not use this code segment:

```
var query = from v in Vehicle.GetVehicles().Where(v => v.Make == "Honda")  
select new {Make = v.Make};
```

This code returns a collection of anonymous type instances with each type having a property named `Make` that is set to "Honda".

1.1.137 Question 138 Objective : Implement Data Access Subobjective Query and manipulate data and object by using LINQ

You use Microsoft .NET Framework 4.5 to create an application. This class exists:

```
public class Publisher
{
    public string Name;
}

public class Book
{
    public string ISBN {get; set;}
    public string Title {get; set;}
    public Publisher Publisher {get; set;}

    public static List<Book> GetBooks()
    {
        // Implementation Omitted
    }
}
```

You need to create a query that returns only the ISBN, title, and publisher's name of all Book instances. You must return the publisher's name as a property named Publisher.

Which code segment should you use?

Choose the correct answer

- var query = Book.GetBooks().Select(b =>
 new {b.ISBN, b.Title, b.Publisher});
- var query = Book.GetBooks().Select(b => b);
- var query = from b in Book.GetBooks()
 select new {b.ISBN, b.Title, b.Publisher};
- var query = from b in Book.GetBooks()
 select new {b.ISBN, b.Title, Publisher=b.Publisher.Name};

1.1.137.1 Correction



```
var query = from b in Book.GetBooks()  
select new {b.ISBN, b.Title, Publisher=b.Publisher.Name};
```

^ Explanation

You should use this code segment:

```
var query = from b in Book.GetBooks()  
select new {b.ISBN, b.Title, Publisher=b.Publisher.Name};
```

This code uses a LINQ query expression to return a collection of anonymous type instances that have ISBN, Title, and Publisher properties. The ISBN and Title properties are set to the corresponding properties of the Book instances. The Publisher property is set to the Name property of the associated Publisher instance.

You should not use this code segment:

```
var query = from b in Book.GetBooks()  
select new {b.ISBN, b.Title, b.Publisher};
```

This code returns a collection of anonymous type instances that have properties named ISBN, Title, and Publisher. However, the Publisher property is set to a Publisher instance, not a string that represents the name of a publisher.

You should not use this code segment:

```
var query = Book.GetBooks().Select(b =>  
new {b.ISBN, b.Title, b.Publisher});
```

This code uses the LINQ Select method to return a collection of anonymous type instances that have properties named ISBN, Title, and Publisher. However, the Publisher property is set to a Publisher instance, not a string that represents the name of a publisher.

1.1.138 Question 139 Objective : Implement Data Access Subobjective Query and manipulate data and object by using LINQ

You use Microsoft .NET Framework 4.5 to create an application. This class exists:

```
public class Person
{
    public int Age {get; set;}
    public string Name {get; set;}
    public static List<Person> GetPeople()
    {
        // Implementation Omitted
    }
}
```

You need to call the GetPeople method and sort the list of Person instances by name and then by age.

Which code segment should you use?

Choose the correct answer

- var sortedPeople = from p in Person.GetPeople() orderby p.Name orderby p.Age
select p;
- var sortedPeople = Person.GetPeople().OrderBy(p => p.Name).OrderBy(p => p.Age);
- var sortedPeople = Person.GetPeople().OrderBy(n => n).
ThenBy(p => p.Age).OrderBy(p => p.Name);
- var sortedPeople = Person.GetPeople().OrderBy(p => p.Name).ThenBy(p => p.Age);

1.1.138.1 Correction



```
var sortedPeople = Person.GetPeople().OrderBy(p => p.Name).ThenBy(p => p.Age);
```

Explanation

You should use this code segment:

```
var sortedPeople = Person.GetPeople().OrderBy(p => p.Name).ThenBy(p => p.Age);
```

This code calls the `OrderBy` method to first order the list by the `Name` property of each `Person` instance. It then calls the `ThenBy` method, which allows you to order results by a second key.

You should not use this code segment:

```
var sortedPeople = Person.GetPeople().OrderBy(p => p.Name).OrderBy(p => p.Age);
```

You should use only one `OrderBy` method. If you use multiple `OrderBy` methods, only the last method is used to determine the order of the returned data. To order results by multiple keys, you must call the `ThenBy` method.

You should not use this code segment:

```
var sortedPeople = Person.GetPeople().OrderBy(n => n).
    ThenBy(p => p.Age).OrderBy(p => p.Name);
```

This code calls the `OrderBy` method, followed by the `ThenBy` method, followed by the `OrderBy` method. You must call the `OrderBy` method first. You must use the `ThenBy` method for all subsequent ordering.

You should not use this code segment:

```
var sortedPeople = from p in Person.GetPeople() orderby p.Name orderby p.Age
select p;
```

This code uses a LINQ query expression and specifies multiple `orderby` clauses. The `orderby` clause works the same as the `OrderBy` method. If you use the `orderby` clause, you can specify multiple keys separated by commas. The result set is ordered according to the order of the placement of the keys. You should only specify the `orderby` clause once, such as follows:

```
orderby p.Name, p.Age
```

If you specify multiple `orderby` clauses, only the last clause is evaluated.

1.1.139 Question 140 Objective : Implement Data Access Subobjective Query and manipulate data and object by using LINQ

You use Microsoft .NET Framework 4.5 to create an application.

```
public class Result
{
    public double Value;
    public DateTime Date;

    public static List<Result> GetResults()
    {
        // Implementation Omitted
    }
}
```

You need to create a query to determine the number of Result instances for each distinct Date value.

Which code segment should you use?

Choose the correct answer

- var query = from r in Result.GetResults() select new {r, Count=r.Date};
- var query = Result.GetResults().Select(r => new {Count=r.Date});
- var query = from r in Result.GetResults() select new {Count=r.Date};
- var query = from r in Result.GetResults() group r by r.Date into g
select new {Date=g.Key, Count=g.Count());

1.1.139.1 Correction



```
var query = from r in Result.GetResults() group r by r.Date into g  
select new {Date=g.Key, Count=g.Count());
```

Explanation

You should use this code segment:

```
var query = from r in Result.GetResults() group r by r.Date into g  
select new {Date=g.Key, Count=g.Count());
```

This code uses a LINQ query expression to group data returned by `GetResults()` by the `Date` property of the `Result` class. The `group` clause allows you to group items in a collection by a key that you provide. The `by` keyword specifies the key. In this scenario, the key is the `Date` property. This code returns an anonymous type instance that contains two properties named `Date` and `Count`. The `Date` property represents the value of the distinct key, and the `Count` property represents the count of the number of items grouped by that key.

You should not use this code segment:

```
var query = from r in Result.GetResults() select new {r, Count=r.Date);
```

This query returns a collection of anonymous type instances that contain two properties named `r` and `Count`. The `r` property returns an anonymous type instance, and the `Count` property returns the value of the `Date` property. This query does not group the results as required in this scenario.

You should not use this code segment:

```
var query = from r in Result.GetResults() select new {Count=r.Date);
```

This query returns a collection of anonymous type instances that contain one property named `Count`, which returns the value of the `Date` property. This query does not group the results as required in this scenario.

You should not use this code segment:

```
var query = Result.GetResults().Select(r => new {Count=r.Date});
```

This query uses the `Select` statement to return a collection of anonymous type instances that contain one property named `Count`, which returns the value of the `Date` property. This query does not group the results as required in this scenario.

1.1.140 Question 141 Objective : Implement Data Access Subobjective Query and manipulate data and object by using LINQ

You use Microsoft .NET Framework 4.5 to create an application. This XML data exists:

```
<Theater>
<Movie Title="The Hobbit: An Unexpected Journey" Rating="PG-13" ReleaseDate="1987"/>
<Movie Title="The Croods" Rating="PG" ReleaseDate="" />
<Movie Title="Escape from Planet Earth" Rating="PG" ReleaseDate="" />
</Theater>
```

You create an XDocument instance named xDoc that contains the previous XML data.

You need to use LINQ to XML to set the ReleaseDate attribute of all Movie elements to 2013 if the ReleaseDate attribute is not set.

Which code segment should you use?

Choose the correct answer

var movies = from m in xDoc.Descendants("Movie")
where m.Attribute("ReleaseDate").Value == "" select m;
movies.ToList().ForEach(m => m.Attribute("ReleaseDate").Value = "2013");

var attributes = (from m in xDoc.Elements("Movie") select m.Attributes());
SelectMany(a => a);
foreach (var a in attributes)
{
 if (a.Value == "") a.Value = "2013";
}

var releaseDates = from m in xDoc.Descendants("Movie")
select m.Attribute("ReleaseDate").Value;
releaseDates.ToList().ForEach(r =>
{
 if (r == "") r = "2013";
});

var movies = xDoc.Descendants("Movie").TakeWhile
(x => x.Attribute("ReleaseDate").Value == "");
movies.ToList().ForEach(m => m.Attribute("ReleaseDate").Value = "2013");

1.1.140.1 Correction

^ Explanation

You should use this code segment:

```
var movies = from m in xDoc.Descendants("Movie")
where m.Attribute("ReleaseDate").Value == "" select m;
movies.ToList().ForEach(m => m.Attribute("ReleaseDate").Value = "2013");
```

This code uses a LINQ query expression to return all descendant elements named Movie that have an attribute named ReleaseDate set to an empty string. It then calls the `ToList` method to create a list of `XElement` instances that represent the found elements. It calls the `ForEach` method of the returned list to iterate through each element and set the value of the `ReleaseDate` attribute to 2013.

You should not use this code segment:

```
var releaseDates = from m in xDoc.Descendants("Movie")
select m.Attribute("ReleaseDate").Value;
releaseDates.ToList().ForEach(r =>
{
    if (r == "") r = "2013";
});
```

This code returns a list of the `ReleaseDate` attribute values. It then iterates through each string in the list and attempts to reset the value of that string if it is empty. However, because strings are passed by value, this code only sets the value of a local variable in each iteration. It does not update the underlying XML tree.

You should not use this code segment:

```
var movies = xDoc.Descendants("Movie").TakeWhile
(x => x.Attribute("ReleaseDate").Value == "");
movies.ToList().ForEach(m => m.Attribute("ReleaseDate").Value = "2013");
```

The `TakeWhile` method queries data until a certain condition is no longer met. This code queries all descendant `Movie` elements as long as the `ReleaseDate` attribute is not set. It begins at the first descendant `Movie` element. Since the value of the `ReleaseDate` attribute of that element is set, then the query would immediately stop processing. Therefore, it would return an empty result set.

You should not use this code segment:

```
var attributes = (from m in xDoc.Elements("Movie") select m.Attributes()).SelectMany(a => a);
foreach (var a in attributes)
{
    if (a.Value == "") a.Value = "2013";
}
```

There are no `Movie` elements that are directly beneath the root document. Therefore, execution would never enter the `foreach` loop.

1.1.141 Question 142 Objective : Implement Data Access Subobjective Store data and retrieve data from collections.

You use Microsoft .NET Framework 4.5 to create a component. This method exists:

```
public void AddProduct(Product product)
{
}
```

The Product class contains an ID property that uniquely identifies a product. Another application calls your component to add products to the company's database. Sometimes the application might attempt to add duplicate products. You need to use a collection class that allows you to add a product only once so you cannot add a product to the collection if another product with the same identifier has already been added. If a product is retrieved from the collection, it must not be removed from the collection automatically.

You need to choose a collection class.

Which class should you use?

Choose the correct answer

- Dictionary
- Queue
- Stack
- ArrayList

1.1.141.1 Correction

^ Explanation

You should use the Dictionary class. This class allows you to organize data in a collection by keys. Keys must be unique. In this scenario, the product identifier is unique. Therefore, a Product instance can be stored in the collection only once.

You should not use the ArrayList class. This class allows you to order items in a sizable collection. You cannot ensure the uniqueness of items in the list.

You should not use the Stack class. This class implements a last in first out (LIFO) algorithm. You cannot ensure the uniqueness of items in the list. Also, retrieved items are automatically removed from the collection.

You should not use the Queue class. This class implements a first in first out (FIFO) algorithm. You cannot ensure the uniqueness of items in the list. Also, retrieved items are automatically removed from the collection.

1.1.142 Question 143 Objective : Implement Data Access Subobjective Store data and retrieve data from collections.

You use Microsoft .NET Framework 4.5 to create an application. The application must sort instances of an Employee class in a collection. You must be able to access an Employee instance by its index in the collection. Retrieving an Employee instance must not remove the instance from the collection.

You need to choose the collection class.

Which class should you use?

Choose the correct answer

- Dictionary
- List
- Stack
- Queue

1.1.142.1 Correction

^ Explanation

You should use the List class. This class allows you to order items in a sizable collection. You can access each item in the collection by its index.

You should not use the Dictionary class. This class allows you to organize data in a collection by keys. You cannot access items in this collection by index.

You should not use the Stack class. This class implements a last in first out (LIFO) algorithm. You cannot access items in this collection by index. Also, retrieved items are automatically removed from the collection.

You should not use the Queue class. This class implements a first in first out (FIFO) algorithm. You cannot access items in this collection by index. Also, retrieved items are automatically removed from the collection.

1.1.143 Question 144 Objective : Implement Data Access Subobjective Store data and retrieve data from collections.

You use Microsoft .NET Framework 4.5 to create an application. Your application must store a collection of people in a particular order. If one person is removed from the collection, the collection automatically resizes itself to not leave any gaps. Also, accessing a person from the collection must not remove the person from the collection.

You need to choose a collection class.

Which class should you use?

Choose the correct answer

- Queue
- Stack
- LinkedList
- ArrayList

1.1.143.1 Correction

^ **Explanation**

You should use the `LinkedList` class. This collection class implements a doubly linked list. It allows you to quickly determine the immediate sibling items for a specified item in the collection. Removing an item from the collection automatically resizes it so that it does not leave any gaps.

You should not use the `ArrayList` class. This class implements an ordered array that allows you to access items by index. Removing an item does not automatically resize the collection, so it leaves gaps.

You should not use the `Queue` class. This class implements a first in first out (FIFO) algorithm. Retrieved items are automatically removed from the collection.

You should not use the `Stack` class. This class implements a last in first out (LIFO) algorithm. Retrieved items are automatically removed from the collection.

1.1.144 Question 145 Objective : Implement Data Access Subobjective Store data and retrieve data from collections.

You use Microsoft .NET Framework 4.5 to create an application. This class exists (line numbers are included for reference only):

```
public enum Planet
{
    Mercury,
    Venus,
    Earth,
    Mars,
    Jupiter,
    Saturn,
    Uranus,
    Neptune,
    PlutoDwarf
}

01 public class SolarSystem
02 {
03     public static IEnumerable<Planet> PlanetsByDistanceFromSun
04     {
05         get
06         {
07         }
08     }
09 }
```

You must implement the PlanetsByDistanceFromSun property to iterate through the Planet enumeration when it is used in a foreach loop as follows:

```
foreach (Planet p in Universe.PlanetsByDistanceFromSun)
{
}
```

You need to write code to accomplish your goal. You must ensure that reorganization of the Planet enumeration does not affect the order of the items returned in the foreach loop.

What should you do?

Choose the correct answer

Add this code between lines 06 and 07:



return (IEnumerable)typeof(Planet).GetEnumValues();

Add this code between lines 02 and 03:

```
Stack planets = new Stack();
planets.Add(Planet.Mercury);
planets.Add(Planet.Venus);
planets.Add(Planet.Earth);
planets.Add(Planet.Mars);
planets.Add(Planet.Jupiter);
planets.Add(Planet.Saturn);
planets.Add(Planet.Uranus);
planets.Add(Planet.Neptune);
planets.Add(Planet.PlutoDwarf);
```



Add this code between lines 06 and 07:

```
return planets.Pop();
```

Add this code between lines 06 and 07:

```
yield return Planet.Mercury;
yield return Planet.Venus;
yield return Planet.Earth;
yield return Planet.Mars;
yield return Planet.Jupiter;
yield return Planet.Saturn;
yield return Planet.Uranus;
yield return Planet.Neptune;
yield return Planet.PlutoDwarf;
```



Add this code between lines 02 and 03:

```
Queue planets = new Queue();
planets.Enqueue(Planet.Mercury);
planets.Enqueue(Planet.Venus);
planets.Enqueue(Planet.Earth);
planets.Enqueue(Planet.Mars);
planets.Enqueue(Planet.Jupiter);
planets.Enqueue(Planet.Saturn);
planets.Enqueue(Planet.Uranus);
planets.Enqueue(Planet.Neptune);
planets.Enqueue(Planet.PlutoDwarf);
```



Add this code between lines 06 and 07:

```
return planets.Dequeue();
```

1.1.144.1 Correction

Explanation

You should use this code segment:

```
yield return Planet.Mercury;
yield return Planet.Venus;
yield return Planet.Earth;
yield return Planet.Mars;
yield return Planet.Jupiter;
yield return Planet.Saturn;
yield return Planet.Uranus;
yield return Planet.Neptune;
yield return Planet.PlutoDwarf;
```

The `yield` keyword specifies that the current property or method is an iterator. The `yield return` statement allows you to return one element during each iteration. In this scenario, each `yield return` statement returns a different `Planet` enumeration value during each iteration in the `foreach` loop.

You should not use the `Stack` class to store a collection of `Planet` enumeration values. Calling the `Stack` instance's `Pop` method would return a `Planet` enumeration value. In this scenario, the property must return an `IEnumerable` instance.

You should not use the `Queue` class to store a collection of `Planet` enumeration values. Calling the `Queue` instance's `Dequeue` method would return a `Planet` enumeration value. In this scenario, the property must return an `IEnumerable` instance.

You should not use this code segment:

```
return (IEnumerable<Planet>)typeof(Planet).GetEnumValues();
```

This code may not necessarily return the enumeration values in the order that you want them to be returned. Also, reorganizing the `Planet` enumeration might change the order in which the items are returned in the `foreach` loop.

1.1.145 Question 146 Objective : Implement Data Access Subobjective Consume data.

You use Microsoft .NET Framework 4.5 to create an application. You are writing code that must call these methods:

- *Withdraw
- *Deposit
- *LogTransaction

The methods must meet these transactional requirements:

- *If the Deposit method throws an exception, changes made by the Withdraw method must be rolled back.
- *If an exception is thrown by the LogTransaction method, changes made by the Withdraw and Deposit methods must NOT be rolled back.

You need to write the necessary transaction management code.

Which code segment should you use?

Choose the correct answer

using (TransactionScope scope = new TransactionScope())
{
 Withdraw();
 Deposit();
 using (TransactionScope scope2 =
 new TransactionScope(TransactionScopeOption.Suppress))
 {
 LogTransaction();
 }
 scope.Complete();
}

using (TransactionScope scope = new TransactionScope())
{
 Withdraw();
 Deposit();
 scope.Complete();
 LogTransaction();
 scope.Complete();
}

using (Transaction scope = Transaction.Current)
{
 Withdraw();
 Deposit();
 using (TransactionScope scope2 = new TransactionScope(TransactionScopeOption.Suppress))
 {
 LogTransaction();
 }
}

using (TransactionScope scope = new TransactionScope())
{
 Withdraw();
 Deposit();
 LogTransaction();
 scope.Complete();
}

1.1.145.1 Correction

^ Explanation

You should use this code segment:

```
using (TransactionScope scope = new TransactionScope())
{
    Withdraw();
    Deposit();
    using (TransactionScope scope2 =
        new TransactionScope(TransactionScopeOption.Suppress))
    {
        LogTransaction();
    }
    scope.Complete();
}
```

The TransactionScope class allows you to manage implicit transactions, which are those that automatically commit when you call the Complete method, unless an exception is thrown. The transaction rolls back automatically if an exception is thrown. The Suppress TransactionScopeOption allows you to prevent an operation from enlisting in a current transaction. This allows transactional operations to succeed even if the operations within the nested transaction scope fail.

You should not use this code:

```
using (Transaction scope = Transaction.Current)
{
    Withdraw();
    Deposit();
    using (TransactionScope scope2 = new TransactionScope(TransactionScopeOption.Suppress))
    {
        LogTransaction();
    }
}
```

The Transaction class is used to manage explicit transactions explicitly. This class has a Rollback method that allows you to rollback a transaction. This code attempts to obtain the current Transaction instance. However, no current transaction has been created.

You should not use this code segment:

```
using (TransactionScope scope = new TransactionScope())
{
    Withdraw();
    Deposit();
    LogTransaction();
    scope.Complete();
}
```

This code calls all three methods within the same transaction scope. This means that if the LogTransaction method throws an exception, the operations made by the Withdraw and Deposit methods would be rolled back.

You should not use this code segment:

```
using (TransactionScope scope = new TransactionScope())
{
    Withdraw();
    Deposit();
    scope.Complete();
    LogTransaction();
    scope.Complete();
}
```

This code calls the Complete method twice. Once you call the Complete method, the transaction commits. If you call it more than once, an exception of type InvalidOperationException is thrown.

1.1.146 Question 147 Objective : Implement Data Access Subobjective Consume data.

You use Microsoft .NET Framework 4.5 to create an application that accesses data in a Microsoft SQL Server 2014 database. You write this code. (Line numbers are for reference only.)

```
01 using(SqlConnection connection = new SqlConnection(connectionString))
02 {
03     String query = "Update Group Set Name='Admin' Where ID= 2";
04     int rowsChanged;
05
06 }
```

You need to execute the query and store the number of rows affected in the rowsChanged variable.

What code should you insert at line 05?

Choose the correct answer

connection.Open();
SqlDataAdapter da = new SqlDataAdapter("", connection);
da.UpdateCommand = new SqlCommand(query);
rowsChanged = da.Update(new DataSet());

OleDbCommand command = new OleDbCommand(query,
(OleDbConnection)(IDbConnection)(connection));
rowsChanged = command.ExecuteNonQuery();

connection.Open();
SqlCommand command = new SqlCommand(query, connection);
rowsChanged = command.ExecuteNonQuery();

connection.Open();
SqlCommand command = new SqlCommand(query, connection);
rowsChanged = (int) command.ExecuteScalar();

1.1.146.1 Correction

^ Explanation

You should use this code segment:

```
connection.Open();
SqlCommand command = new SqlCommand(query, connection);
rowsChanged = command.ExecuteNonQuery();
```

The ExecuteNonQuery method of the SqlCommand class executes a query and returns the number of rows affected by the query.

You should not use this code segment:

```
connection.Open();
SqlCommand command = new SqlCommand(query, connection);
rowsChanged = (int) command.ExecuteScalar();
```

The ExecuteScalar method of the SqlCommand class executes a query that returns a single result. It does not store the number of rows affected in the rowsChanged variable.

You should not use this code segment:

```
OleDbCommand command = new OleDbCommand(query,
(OleDbConnection)(IDbConnection)(connection));
rowsChanged = command.ExecuteNonQuery();
```

You must open a connection before you execute the query. Also, you must not use an OleDbCommand instance with a SqlConnection instance. If you use a SqlConnection instance to create a connection, you must use a SqlCommand instance to construct and execute the query. It does not store the number of rows affected in the rowsChanged variable.

You should not use this code segment:

```
connection.Open();
SqlDataAdapter da = new SqlDataAdapter("", connection);
da.UpdateCommand = new SqlCommand(query);
rowsChanged = da.Update(new DataSet());
```

The SqlDataAdapter class allows you to fill a DataSet instance or update a database with changes made to a DataSet instance. It does not store the number of rows affected in the rowsChanged variable.

1.1.147 Question 148 Objective : Implement Data Access Subobjective Consume data.

You use Microsoft .NET Framework 4.5 to create an application. This stored procedure is declared in a Microsoft SQL Server 2012 database:

```
CREATE PROCEDURE AddUser
    @UserName nvarchar(50),
    @ID int out
```

You create a SqlCommand instance named cmd to represent the stored procedure. You must add parameters to the SqlCommand instance to add a user named JDavis and obtain the ID of that user.

You need to write code to accomplish your goal.

Which code segment should you use?

Choose the correct answer

cmd.Parameters.AddWithValue("@UserName", "JDavis");
 SqlParameter param = new SqlParameter("@ID", SqlDbType.Int);
 param.Direction = ParameterDirection.ReturnValue;
 cmd.Parameters.Add(param);

cmd.Parameters.AddWithValue("UserName", "JDavis");
 SqlParameter param = new SqlParameter("ID", SqlDbType.Int);
 param.Direction = ParameterDirection.Input;
 cmd.Parameters.Add(param);

cmd.Parameters.AddWithValue("@UserName", "JDavis");
 SqlParameter param = new SqlParameter("@ID", SqlDbType.Int);
 param.Direction = ParameterDirection.Output;
 cmd.Parameters.Add(param);

cmd.Parameters.AddWithValue("UserName", "JDavis");
 SqlParameter param = new SqlParameter("ID", SqlDbType.Int);
 param.Direction = ParameterDirection.InputOutput;
 cmd.Parameters.Add(param);

1.1.147.1 Correction

^ Explanation

You should use this code segment:

```
cmd.Parameters.AddWithValue("@UserName", "JDavis");
SqlParameter param = new SqlParameter("@ID", SqlDbType.Int);
param.Direction = ParameterDirection.Output;
cmd.Parameters.Add(param);
```

The SqlCommand class contains a Parameters property that returns a collection of SqlParameter instances, which represents parameters to a stored procedure or query. The AddWithValue method of the SqlParameter class allows you to add a SQL parameter to a stored procedure or query. The first parameter to the method represents the name of the SQL parameter. You must specify the name exactly as it appears in the stored procedure or query. The second parameter to the method represents the SQL parameter's value. By creating a new SqlParameter instance, you can set the Direction property to ParameterDirection.Output. This is necessary because the AddUser method accepts an OUT parameter. The value of this parameter is set after the stored procedure executes.

```
cmd.Parameters.AddWithValue("@UserName", "JDavis");
SqlParameter param = new SqlParameter("@ID", SqlDbType.Int);
param.Direction = ParameterDirection.ReturnValue;
cmd.Parameters.Add(param);
```

This code sets the Direction property of the SqlParameter class to ParameterDirection.ReturnValue. You should use this value for stored procedures or queries that return values using the RETURN statement.

```
cmd.Parameters.AddWithValue("UserName", "JDavis");
SqlParameter param = new SqlParameter("ID", SqlDbType.Int);
param.Direction = ParameterDirection.InputOutput;
cmd.Parameters.Add(param);
```

This code sets the Direction property of the SqlParameter class to ParameterDirection.InputOutput. Also, this code passes the wrong parameter name to the AddWithValue method. You must use @UserName as the parameter name.

```
cmd.Parameters.AddWithValue("UserName", "JDavis");
SqlParameter param = new SqlParameter("ID", SqlDbType.Int);
param.Direction = ParameterDirection.Input;
cmd.Parameters.Add(param);
```

This code sets the Direction property of the SqlParameter class to ParameterDirection.Input. You should use this value for setting input parameters. For example, @UserName is an input parameter, and @ID is an output parameter. Also, this code passes the wrong parameter name to the AddWithValue method. You must use @UserName as the parameter name.

1.1.148 Question 149 Objective : Implement Data Access Subobjective Consume data.

You use Microsoft .NET Framework 4.5 to create an application. A Microsoft SQL Server 2012 stored procedure is implemented with this statement:

```
SELECT * from dbo.Claim WHERE ClaimType=2 FOR XML AUTO
```

You create a SqlCommand instance to call the stored procedure.

You need to call a method of the SqlCommand instance to process the results returned by the stored procedure.

Which method should you call?

Choose the correct answer

- ExecuteXmlReader
- ExecuteReader
- ExecuteNonQuery
- ExecuteScalar

1.1.148.1 Correction

^ Explanation

You should call the ExecuteXmlReader method. This method allows you to process XML results that are returned from a query. In this scenario, the SQL statement uses the clause FOR XML AUTO. This clause returns a result set as XML data, automatically generating nested elements to represent field names and values.

You should not call the ExecuteReader method. This method allows you to process a result set that is not XML data. In this scenario, you need to process XML data that is returned from the stored procedure.

You should not call the ExecuteNonQuery method. This method allows you to execute a query that does not return a result set. This method returns the number of records affected by the query. In this scenario, you need to process results that are returned as XML data.

You should not call the ExecuteScalar method. This method allows you to execute a query that returns a single value. In this scenario, a result set is returned as XML data, not a single value.

1.1.149 Question 150 Objective : Implement Data Access Subobjective Consume data.

You use Microsoft .NET Framework 4.5 to create an application. The application accesses a Web service that returns data in this format:

```
{"ID":123,"ClaimType":2,"ClaimStatus":1}
```

This code exists in your application:

```
[DataContract]  
public class Claim  
{  
    public int ID {get; set;}  
    public int ClaimType {get; set;}  
    public int ClaimStatus {get; set;}  
}
```

You write this code to return data from the Web service:

```
01 string url = "http://app.measureup.com/GetLastClaim";  
02 HttpWebRequest request = (HttpWebRequest)WebRequest.Create(url);  
03 using (HttpWebResponse response = (HttpWebResponse)request.GetResponse())  
04 {  
05     Stream stream = response.GetResponseStream();  
06  
07 }
```

You need to insert code at line 06 to convert the data returned from the Web service to a Claim instance.

Which code segment should you use?

Choose the correct answer

StreamReader reader = new StreamReader(stream);
Claim claim = (Claim)(object)reader.ReadToEnd();

Claim claim = (Claim)(object)stream;

Claim claim = (Claim)(object)stream.CreateObjRef(typeof(Claim));

DataContractJsonSerializer serializer =
new DataContractJsonSerializer(typeof(Claim));
Claim claim = (Claim)serializer.ReadObject(stream);

1.1.149.1 Correction

```
DataContractJsonSerializer serializer =  
    new DataContractJsonSerializer(typeof(Claim));  
Claim claim = (Claim)serializer.ReadObject(stream);
```

^ Explanation

You should use this code segment:

```
DataContractJsonSerializer serializer =  
    new DataContractJsonSerializer(typeof(Claim));  
Claim claim = (Claim)serializer.ReadObject(stream);
```

The data returned by the Web service is JavaScript Object Notation (JSON) data. By using the DataContractJsonSerializer class, you can convert that data to a common language runtime (CLR) type. The ReadObject method of that class performs the conversion.

You should not use this code segment:

```
Claim claim = (Claim)(object)stream;
```

This code attempts to cast the Stream instance to a Claim instance. This will not work because it throws an exception of type InvalidCastException.

You should not use this code segment:

```
StreamReader reader = new StreamReader(stream);  
Claim claim = (Claim)(object)reader.ReadToEnd();
```

This code uses the StreamReader class and calls its ReadToEnd method. This method returns a string that represents a text-based stream. Casting the string to a Claim instance does not work. It would throw an exception of type InvalidCastException.

You should not use this code segment:

```
Claim claim = (Claim)(object)stream.CreateObjRef(typeof(Claim));
```

This code calls the CreateObjectRef method and casts the returned an ObjectRef instance. It then attempts to cast that instance to a Claim instance. This does not work because it throws an exception of type InvalidCastException. You should use the CreateObjectRef method to create a proxy to be used with remote procedure calls (RPCs).

1.1.150 Question 151 Objective : Implement Data Access Subobjective Consume data.

You are creating a C# application by using Visual Studio 2015. You need to consume JSON data. You have created the following code:

```
public class Vendor
{
    public string Company { get; set; }
    public string State { get; set; }
}
```

You need to create a Vendor object from JSON data.

Which code should you write?

Choose the correct answer

`Vendor CreateVendor(string json)
{
 DataContractJsonSerializer js = new DataContractJsonSerializer(Vendor);
 MemoryStream stream = new MemoryStream(Encoding.UTF8.GetBytes(json));
 Vendor vendor = (Vendor)js.ReadObject(stream);
 return vendor;
}`

`Vendor CreateVendor(string json)
{
 DataContractJsonSerializer js = new DataContractJsonSerializer(typeof(Vendor));
 MemoryStream stream = new MemoryStream(Encoding.UTF8.GetBytes(json));
 Vendor vendor = stream.ReadObject(js);
 return vendor;
}`

`Vendor CreateVendor(string json)
{
 DataContractJsonSerializer js = new DataContractJsonSerializer(typeof(Vendor));
 MemoryStream stream = new MemoryStream(Encoding.UTF8.GetBytes(json));
 Vendor vendor = js.ReadObject(stream);
 return vendor;
}`

`Vendor CreateVendor(string json)
{
 DataContractJsonSerializer js = new DataContractJsonSerializer(typeof(Vendor));
 MemoryStream stream = new MemoryStream(Encoding.UTF8.GetBytes(json));
 Vendor vendor = (Vendor)js.ReadObject(stream);
 return vendor;
}`

1.1.150.1 Correction

```
Vendor CreateVendor(string json)
{
    DataContractJsonSerializer js = new DataContractJsonSerializer(typeof(Vendor));
    MemoryStream stream = new MemoryStream(Encoding.UTF8.GetBytes(json));
    Vendor vendor = (Vendor)js.ReadObject(stream);
    return vendor;
}
```

Explanation

You should use the code below:

```
Vendor CreateVendor(string json)
{
    DataContractJsonSerializer js = new DataContractJsonSerializer(typeof(Vendor));
    MemoryStream stream = new MemoryStream(Encoding.UTF8.GetBytes(json));
    Vendor vendor = (Vendor)js.ReadObject(stream);
    return vendor;
}
```

This code correctly creates the `DataContractJsonSerializer` with the `Vendor` class. It then creates a `MemoryStream` to store the deserialized data and stores the resulting data into a `Vendor` object.

You should not use the code below:

```
Vendor CreateVendor(string json)
{
    DataContractJsonSerializer js = new DataContractJsonSerializer(Vendor);
    MemoryStream stream = new MemoryStream(Encoding.UTF8.GetBytes(json));
    Vendor vendor = (Vendor)js.ReadObject(stream);
    return vendor;
}
```

This code incorrectly omits the `typeOf` function that should wrap the `Vendor` object being passed to the `DataContractJsonSerializer`.

You should not use the code below:

```
Vendor CreateVendor(string json)
{
    DataContractJsonSerializer js = new DataContractJsonSerializer(typeof(Vendor));
    MemoryStream stream = new MemoryStream(Encoding.UTF8.GetBytes(json));
    Vendor vendor = stream.ReadObject(js);
    return vendor;
}
```

This code incorrectly prefixes the `ReadObject` method with the `MemoryStream` object, and passes the `DataContractJsonSerializer` object into the `ReadObject` method. Instead, the `MemoryStream` object should be passed into the `ReadObject` method, which is prefixed by the `DataContractJsonSerializer` object.

You should not use the code below:

```
Vendor CreateVendor(string json)
{
    DataContractJsonSerializer js = new DataContractJsonSerializer(typeof(Vendor));
    MemoryStream stream = new MemoryStream(Encoding.UTF8.GetBytes(json));
    Vendor vendor = js.ReadObject(stream);
    return vendor;
}
```

This code incorrectly omits the `Vendor` cast that should precede the `js.ReadObject` call. To work correctly, the generic object returned by `ReadObject` must be cast as the correct type.

1.1.151 Question 152 Objective : Implement Data Access Subobjective Serialize and deserialize data.

You use Microsoft .NET Framework 4.5 to create an application. This class definition exists:

```
public class Classroom
{
    public string RoomNumber {get; set;}
}
```

You write this code to create a Classroom instance:

```
Classroom room = new Classroom();
room.RoomNumber = "100A";
```

By not altering the Classroom class definition, you must serialize the instance to this XML:

```
<Classroom number="100A"/>
```

You need to write the code to accomplish your goal.

Which code segment should you use?

Choose the correct answer

`XmlAttributeOverrides overrides = new XmlAttributeOverrides();
XmlAttributes roomNumberAttributes = new XmlAttributes();
roomNumberAttributes.XmlRoot = new XmlRootAttribute("number");
overrides.Add(typeof(Classroom), "RoomNumber", roomNumberAttributes);
XmlSerializer serializer = new XmlSerializer(typeof(Classroom));
MemoryStream stream = new MemoryStream();
serializer.Serialize(stream, room);`

`XmlAttributeOverrides overrides = new XmlAttributeOverrides();
XmlAttributes roomNumberAttributes = new XmlAttributes();
roomNumberAttributes.XmlIgnore = true;
overrides.Add(typeof(Classroom), "number", roomNumberAttributes);
XmlSerializer serializer = new XmlSerializer(typeof(Classroom));
MemoryStream stream = new MemoryStream();
serializer.Serialize(stream, room);`

XmlAttributeOverrides overrides = new XmlAttributeOverrides();
XmlAttributes roomNumberAttributes = new XmlAttributes();
roomNumberAttributes.XmlAttribute = new XmlAttributeAttribute("number");
overrides.Add(typeof(Classroom), roomNumberAttributes);
XmlSerializer serializer = new XmlSerializer(typeof(Classroom));
MemoryStream stream = new MemoryStream();
serializer.Serialize(stream, room);

XmlAttributeOverrides overrides = new XmlAttributeOverrides();
XmlAttributes roomNumberAttributes = new XmlAttributes();
roomNumberAttributes.XmlAttribute = new XmlAttributeAttribute("number");
overrides.Add(typeof(Classroom), "RoomNumber", roomNumberAttributes);
XmlSerializer serializer = new XmlSerializer(typeof(Classroom), overrides);
MemoryStream stream = new MemoryStream();
serializer.Serialize(stream, room);

1.1.151.1 Correction

```
XmlAttributeOverrides overrides = new XmlAttributeOverrides();
XmlAttributes roomNumberAttributes = new XmlAttributes();
roomNumberAttributes.XmlAttribute = new XmlAttributeAttribute("number");
overrides.Add(typeof(Classroom), "RoomNumber", roomNumberAttributes);
XmlSerializer serializer = new XmlSerializer(typeof(Classroom), overrides);
MemoryStream stream = new MemoryStream();
serializer.Serialize(stream, room);
```



^ Explanation

You should use this code segment:

```
XmlAttributeOverrides overrides = new XmlAttributeOverrides();
XmlAttributes roomNumberAttributes = new XmlAttributes();
roomNumberAttributes.XmlAttribute = new XmlAttributeAttribute("number");
overrides.Add(typeof(Classroom), "RoomNumber", roomNumberAttributes);
XmlSerializer serializer = new XmlSerializer(typeof(Classroom), overrides);
MemoryStream stream = new MemoryStream();
serializer.Serialize(stream, room);
```

The `XmlAttributeOverrides` class allows you to override the default XML serialization. By default, a class and its members are serialized as XML elements. The `XmlSerializer` class accepts an `XmlAttributeOverrides` instance that specifies custom serialization attributes. The `XmlAttributeOverrides` class contains a collection of `XmlAttributes` instances. The `XmlAttributes` class represents XML serialization attributes that you can declaratively specify for classes and members. The `XmlAttribute` property of this class allows you to represent a class member as an XML attribute. In this scenario, this code represents the `RoomNumber` property as an XML attribute named `number`:

```
roomNumberAttributes.XmlAttribute = new XmlAttributeAttribute("number");
```

The code then adds the `XmlAttributes` instance to the `XmlAttributeOverrides` instance by calling its `Add` method.

You should not use this code segment:

```
XmlAttributeOverrides overrides = new XmlAttributeOverrides();
XmlAttributes roomNumberAttributes = new XmlAttributes();
roomNumberAttributes.XmlAttribute = new XmlAttributeAttribute("number");
overrides.Add(typeof(Classroom), roomNumberAttributes);
XmlSerializer serializer = new XmlSerializer(typeof(Classroom));
MemoryStream stream = new MemoryStream();
serializer.Serialize(stream, room);
```

This code calls the two-parameter overload of the Add method of the XmlAttributeOverrides class. This overload accepts a Type instance and an XmlAttributes instance. You should only call the two-parameter overload when applying XML attributes to a class itself. When you need to apply XML attributes to class members, you must call the three-parameter overload. This code generates this XML:

```
<Classroom>
  <RoomNumber>100A</RoomNumber>
</Classroom>
```

You should not use this code segment:

```
XmlAttributeOverrides overrides = new XmlAttributeOverrides();
XmlAttributes roomNumberAttributes = new XmlAttributes();
roomNumberAttributes.XmlIgnore = true;
overrides.Add(typeof(Classroom), "number", roomNumberAttributes);
XmlSerializer serializer = new XmlSerializer(typeof(Classroom));
MemoryStream stream = new MemoryStream();
serializer.Serialize(stream, room);
```

This code sets the XmlIgnore property of the XmlAttributes class. This property allows you to indicate that a class member should be ignored during XML serialization. This does not work in this scenario because you want the RoomNumber property to be serialized as an XML attribute, not ignored. However, this code does not specify the correct member as the second parameter to the Add method of the XmlAttributeOverrides class. To ignore the RoomNumber property, you should specify RoomNumber instead of number. Because an incorrect member is specified, this code generates this XML:

```
<Classroom>
  <RoomNumber>100A</RoomNumber>
</Classroom>
```

You should not use this code segment:

```
XmlAttributeOverrides overrides = new XmlAttributeOverrides();
XmlAttributes roomNumberAttributes = new XmlAttributes();
roomNumberAttributes.XmlRoot = new XmlRootAttribute("number");
overrides.Add(typeof(Classroom), "RoomNumber", roomNumberAttributes);
XmlSerializer serializer = new XmlSerializer(typeof(Classroom));
MemoryStream stream = new MemoryStream();
serializer.Serialize(stream, room);
```

This code sets the XmlRoot property of the XmlAttributes instance. You should only set the XmlRoot property when you apply XML attributes to a class that acts as the root of the XML document. This does not work in this scenario because you need to serialize the RoomNumber property as an XML attribute. Because the XmlRoot property has no effect on serialization of a class member, this code generates this XML:

```
<Classroom>
  <RoomNumber>100A</RoomNumber>
</Classroom>
```

<https://docs.microsoft.com/en-us/dotnet/standard/serialization/attributes-that-control-xml-serialization>

1.1.152 Question 153 Objective : Implement Data Access Subobjective Consume data.

You use Microsoft .NET Framework 4.5 to create an application. This class exists:

```
public class Customer
{
    private int customerId;
    public Customer(int customerId)
    {
        this.customerId = customerId;
    }
}
```

You write this code:

```
Customer customer = new Customer(12345);
```

You must serialize the customer instance to this XML:

```
<Customer>
    <customerId>12345</customerId>
</Customer>
```

You plan to apply serialization attributes to the class and its members, but you cannot modify the members.

You need to choose a serialization class.

Which class should you use?

Choose the correct answer

- DataContractJsonSerializer
- XmlSerializer
- BinaryFormatter
- DataContractSerializer

1.1.152.1 Correction



DataContractSerializer

Explanation

You should use the `DataContractSerializer` class. This class allows you to serialize a class and its members as XML elements. With the `DataContractSerializer` class, you can serialize both public and private properties and fields.

You should not use the `DataContractJsonSerializer` class. This class allows you to serialize a class and its members as JavaScript Object Notation (JSON) text. However, in this scenario, you must serialize the `Customer` class as XML.

You should not use the `XmlSerializer` class. With this class, you can only serialize a class and its public members. However, in this scenario, you need to serialize the private field `customerId`.

You should not use the `BinaryFormatter` class. This class allows you to serialize a class and its members into a binary format. However, in this scenario, you must serialize the `Customer` class as XML.

1.1.153 Question 154 Objective : Implement Data Access Subobjective Consume data.

You use Microsoft .NET Framework 4.5 to create an application. This class exists:

```
public class Claim
{
    public string Number {get; set;}
    public DateTime Date {get; set;}
}
```

This code exists to serialize an instance of Claim as XML:

```
Claim claim = new Claim{Number="12345", Date=DateTime.Now};
DataContractSerializer serializer = new DataContractSerializer(typeof(Claim));
MemoryStream stream = new MemoryStream();
serializer.WriteObject(stream, claim);
```

You must ensure that the claim instance is serialized as XML.

What should you do?

Choose the correct answer

- Apply the `XmlRoot` attribute to the `Claim` class. Apply the `XmlElement` attribute to the `Number` and `Date` properties.
- Apply the `SoapType` attribute to the `Claim` class. Apply the `SoapElement` attribute to the `Number` and `Date` properties.
- Apply the `DataContract` attribute to the `Claim` class. Apply the `DataMember` attribute to the `Number` and `Date` properties.
- Apply the `XmlRoot` attribute to the `Claim` class. Apply the `XmlAttribute` attribute to the `Number` and `Date` properties.

1.1.153.1 Correction

^ Explanation

You should apply the DataContract attribute to the Claim class and the DataMember attribute to the Number and Date properties. TheDataContractSerializer class uses the DataContract attribute or the Serializable attribute to determine whether or not to serialize a class. If you use the Serializable attribute, theDataContractSerializer class serializes all members that are not marked with the NonSerialized attribute. If you use the DataContract attribute, theDataContractSerializer class serializes all members that are explicitly marked with the DataMember attribute.

You should not apply the XmlRoot attribute to the Claim class and the XmlElement attribute to the Number and Date properties. TheDataContractSerializer class does not use the XmlRoot and XmlElement attributes. TheXmlSerializer class uses these attributes to control XML serialization.

You should not apply the XmlRoot attribute to the Claim class and theXmlAttribute attribute to the Number and Date properties. TheDataContractSerializer class does not use the XmlRoot and XmlAttribute attributes. TheXmlSerializer class uses these attributes to control XML serialization.

You should not apply the SoapType attribute to the Claim class and the SoapElement attribute to the Number and Date properties. TheDataContractSerializer class does not use the SoapType and SoapElement attributes. TheXmlSerializer class uses these attributes to control Simple Object Access Protocol (SOAP) serialization.

1.1.154 Question 155 Objective : Implement Data Access Subobjective Perform I/O operations.

You use Microsoft .NET Framework 4.5 to create an application. You deserialize text-based data into a `MemoryStream` instance named `memoryStream`.

You need to obtain the contents of the `memoryStream` instance.

Which code segment should you use?

Choose the correct answer

using (`StreamWriter writer = new StreamWriter(memoryStream)`)
{
 `string data = writer.ToString();`
}

using (`StreamReader reader = new StreamReader(memoryStream)`)
{
 `string data = reader.ReadToEnd();`
}

`StringReader reader = new StringReader(memoryStream.ToString());`
`String data = reader.ReadToEnd();`

`StringBuilder builder = new StringBuilder(memoryStream.ToString());`
`String data = builder.ToString();`

1.1.154.1 Correction

^ Explanation

You should use this code segment:

```
using (StreamReader reader = new StreamReader(memoryStream))
{
    string data = reader.ReadToEnd();
}
```

The StreamReader class allows you to read text-based input from a stream. The constructor accepts a Stream instance that represents the data to read. The ReadToEnd method returns a string that represents the text-based data read from the stream.

You should not use this code segment:

```
StringBuilder builder = new StringBuilder(memoryStream.ToString());
String data = builder.ToString();
```

The StringBuilder class allows you to dynamically create a string without having to allocate multiple string instances for each concatenation. It accepts an initial string as a parameter in its constructor. However, you should not call the ToString method of the MemoryStream class. This does not return the data contained by the stream. Instead, it returns the type name of the stream.

You should not use this code segment:

```
using (StreamWriter writer = new StreamWriter(memoryStream))
{
    string data = writer.ToString();
}
```

The StreamWriter class allows you to write data to a stream. This does not accomplish your goal because you need to read data from a stream.

You should not use this code segment:

```
StringReader reader = new StringReader(memoryStream.ToString());
String data = reader.ReadToEnd();
```

The StringReader class allows you to read data from a string. However, you should not call the ToString method of the MemoryStream class. This does not return the data contained by the stream. Instead, it returns the type name of the stream.

1.1.155 Question 156 Objective : Implement Data Access Subobjective Perform I/O operations.

You use Microsoft .NET Framework 4.5 to create an application. You create a variable named integers that represents a generic List instance of integers.

You need to write each integer on a separate line in a file named Integers.txt.

Which code segment should you use?

Choose the correct answer

using (StreamWriter writer = new StreamWriter("Integers.txt"))
{
 integers.ForEach(i => writer.WriteLine(i));
}

using (StreamWriter writer = new StreamWriter("Integers.txt"))
{
 integers.ForEach(i =>
 {
 writer.WriteLine(i);
 writer.Close();
 });
}

using (StreamWriter writer = new StreamWriter("Integers.txt"))
{
 integers.ForEach(i =>
 {
 writer.NewLine = "\r
";
 writer.Write(i);
 });
}

using (StreamWriter writer = new StreamWriter("Integers.txt"))
{
 integers.ForEach(i =>
 {
 writer.Write(i);
 writer.NewLine = "\r
";
 });
}

1.1.155.1 Correction

^ Explanation

You should use this code segment:

```
using (StreamWriter writer = new StreamWriter("Integers.txt"))
{
    integers.ForEach(i => writer.WriteLine(i));
}
```

This code calls the `WriteLine` method of the `StreamWriter` class during each iteration. The `WriteLine` method writes data to a new line in the underlying stream.

You should not use this code segment:

```
using (StreamWriter writer = new StreamWriter("Integers.txt"))
{
    integers.ForEach(i =>
    {
        writer.Write(i);
        writer.NewLine = "\r
";
    });
}
```

This code calls the `Write` method of the `StreamWriter` class. The `Write` method writes data to the current line, unless the data itself contains the newline character. Simply setting the `NewLine` property of the `StreamWriter` class does not create a newline. It only specifies the string that represents the newline character or characters.

You should not use this code segment:

```
using (StreamWriter writer = new StreamWriter("Integers.txt"))
{
    integers.ForEach(i =>
    {
        writer.NewLine = "\r
";
        writer.Write(i);
    });
}
```

This code calls the `Write` method of the `StreamWriter` class. The `Write` method writes data to the current line, unless the data itself contains the newline character. Simply setting the `NewLine` property of the `StreamWriter` class does not create a newline. It only specifies the string that represents the newline character or characters.

You should not use this code segment:

```
using (StreamWriter writer = new StreamWriter("Integers.txt"))
{
    integers.ForEach(i =>
    {
        writer.WriteLine(i);
        writer.Close();
    });
}
```

This code calls the `Close` method of the `StreamWriter` class during each iteration, which closes the underlying stream. This would cause an exception of type `ObjectDisposedException` to be thrown when the next `WriteLine` method is called. You cannot write to a stream once the stream is closed.

1.1.156 Question 157 Objective : Implement Data Access Subobjective Perform I/O operations.

You use Microsoft .NET Framework 4.5 to create an application. You use the BinaryWriter class to write five strings to a file named Strings.dat.

You need to retrieve each string written to the Strings.dat file.

Which code segment should you use?

Choose the correct answer

```
using (BinaryReader reader = new BinaryReader(File.OpenRead("Strings.dat")))
{
    for (int i = 0; i < 5; i++)
    {
        byte[] bytes = new byte[i];
        reader.Read(bytes, 0, 1);
        string data = Encoding.ASCII.GetString(bytes);
    }
}
```



```
using (BinaryReader reader = new BinaryReader(File.OpenRead("Strings.dat")))
{
    for (int i = 0; i < 5; i++)
    {
        byte[] bytes = reader.ReadBytes(i);
        string data = Encoding.ASCII.GetString(bytes);
    }
}
```



```
using (BinaryReader reader = new BinaryReader(File.OpenRead("Strings.dat")))
{
    for (int i = 0; i < 5; i++)
    {
        char[] characters = reader.ReadChars(i);
        string data = new String(characters);
    }
}
```



```
using (BinaryReader reader = new BinaryReader(File.OpenRead("Strings.dat")))
{
    for (int i = 0; i < 5; i++)
    {
        string data = reader.ReadString();
    }
}
```

1.1.156.1 Correction

```
using (BinaryReader reader = new BinaryReader(File.OpenRead("Strings.dat")))
{
    for (int i = 0; i < 5; i++)
    {
        string data = reader.ReadString();
    }
}
```



Explanation

You should use this code segment:

```
using (BinaryReader reader = new BinaryReader(File.OpenRead("Strings.dat")))
{
    for (int i = 0; i < 5; i++)
    {
        string data = reader.ReadString();
    }
}
```

This code uses the `BinaryReader` class to read data from a file. The `ReadString` method reads one string from the file. By calling the method within a loop, you can read all five strings from the file.

You should not use this code segment:

```
using (BinaryReader reader = new BinaryReader(File.OpenRead("Strings.dat")))
{
    for (int i = 0; i < 5; i++)
    {
        char[] characters = reader.ReadChars(i);
        string data = new String(characters);
    }
}
```

The `ReadChars` method reads a specified number of characters from the string. This code first reads zero characters from the string, followed by one, two, three, and four. It does not read all strings from the file.

You should not use this code segment:

```
using (BinaryReader reader = new BinaryReader(File.OpenRead("Strings.dat")))
{
    for (int i = 0; i < 5; i++)
    {
        byte[] bytes = new byte[i];
        reader.Read(bytes, 0, 1);
        string data = Encoding.ASCII.GetString(bytes);
    }
}
```

This code calls the `Read` method. This method reads data into a byte array. In this code, it first attempts to read one byte into a byte array of zero length. This throws an exception because you cannot read a byte into a zero-length byte array.

You should not use this code segment:

```
using (BinaryReader reader = new BinaryReader(File.OpenRead("Strings.dat")))
{
    for (int i = 0; i < 5; i++)
    {
        byte[] bytes = reader.ReadBytes(i);
        string data = Encoding.ASCII.GetString(bytes);
    }
}
```

This code calls the `ReadBytes` method. This method reads a specified number of bytes. This code attempts to read zero, followed by one, two, three, and four bytes from the file. It does not read all strings from the file.

1.1.157 Question 158 Objective : Implement Data Access Subobjective Perform I/O operations.

You use Microsoft .NET Framework 4.5 to create an application. This code exists to send data over the network (line numbers are included for reference only):

```
01 TcpClient client = new TcpClient("192.168.1.2", 500);
02 using (NetworkStream networkStream = client.GetStream())
03 {
04     StreamWriter writer = new StreamWriter(networkStream);
05     writer.WriteLine("Sending Network Data");
06 }
```

When the code runs, you discover that no data is sent to endpoint 192.168.1.2. You verify that the server is configured to accept incoming TCP connections over port 500.

You need to solve this problem.

What should you do?

Choose the correct answer

Add this statement after line 06:

client.Client.Accept();

Add this statement between lines 05 and 06:

networkStream.Close();

Add this statement between lines 05 and 06:

writer.Flush();

Add this statement after line 06:

client.Close();

1.1.157.1 Correction

^ Explanation

You should add this statement between lines 05 and 06:

```
writer.Flush();
```

The Flush method of the StreamWriter class causes all buffered data to be written to the underlying stream. In this scenario, the underlying stream is a NetworkStream instance. The problem is that data is being buffered, but not actually sent over the network. Calling the Flush method forces data to be sent over the network.

You should not add this statement after line 06:

```
client.Close();
```

The Close method of the TcpClient class closes an existing connection. Closing the connection does not force data to be sent over the network.

You should not add this statement between lines 05 and 06:

```
networkStream.Close();
```

The NetworkStream instance automatically closes after execution leaves the using block. The using block automatically disposes an object that implements IDisposable. When a NetworkStream instance is disposed, its underlying connection automatically closes. Closing the connection is not the problem in this scenario. The problem is that data is being buffered, but not actually sent over the network.

You should not add this statement after line 06:

```
client.Client.Accept();
```

The Client property of the TcpClient class returns a Socket instance that represents a network socket. The Accept method of the Socket class accepts a connection to the network socket. In this scenario, you should not accept connections because you are sending data over the network, not receiving it.

1.1.158 Question 159 Objective : Implement Data Access Subobjective Perform I/O operations.

You use Microsoft .NET Framework 4.5 to create an application. This code exists:

```
public async void WriteData(byte[] data, Stream stream)
{
}
```

You need to write the byte array represented by the data variable to the stream instance. You must write the data asynchronously and return control immediately to the caller of the WriteData method.

Which code segment should you use?

Choose the correct answer

- await stream.Write(data, 0, data.Length);
- await data.ToList().ForEach(b => stream.WriteByte(b));
- await stream.WriteAsync(data, 0, data.Length);
- data.ToList().ForEach(b => stream.WriteByte(b));

1.1.158.1 Correction

^ Explanation

You should use this code segment:

```
await stream.WriteAsync(data, 0, data.Length);
```

The `WriteAsync` method of the `Stream` class writes data to a stream asynchronously. The `await` operator causes execution to return immediately to the caller of the encapsulating method. Execution resumes within the method where it left after the asynchronous operation completes.

You should not use this code segment:

```
await stream.Write(data, 0, data.Length);
```

The `Write` method of the `Stream` class does not write data asynchronously. Also, you can only apply the `await` operator to a method that returns a `Task` instance.

You should not use this code segment:

```
data.ToList().ForEach(b => stream.WriteByte(b));
```

This code calls the `WriteByte` method for each byte in the array. The `WriteByte` method does not run asynchronously.

You should not use this code segment:

```
await data.ToList().ForEach(b => stream.WriteByte(b));
```

This code calls the `WriteByte` method for each byte in the array. The `WriteByte` method does not run asynchronously. Also, you can only apply the `await` operator to a method that returns a `Task` instance.

1.1.159 Question 160 Objective : Implement Data Access Subobjective Perform I/O operations.

You are developing a Microsoft Windows Presentation Foundation (WPF) application by using C# and Visual Studio 2015. Your application must be able to display the contents of large, encrypted files.

You have this code: (Line numbers are for reference purposes only.)

```
01 // The data variable is initialized elsewhere
02 private CryptoStream data;
03 private async void Button_Click(object sender, RoutedEventArgs args)
04 {
05
06 }
```

You need to complete the code to display the contents of the file in a user interface object named outputText. You should ensure that the application remains responsive. Note that the user interface objects in a WPF application must be accessed only by the one thread that owns all user interface objects.

Which code block completes the method when inserted at line 05?

Choose the correct answer

`outputText.Dispatcher.BeginInvoke(() => {
 var reader = new StreamReader(data);
 outputText.Text = reader.ReadToEnd();
})`

`var reader = new StreamReader(data);
outputText.Text = await reader.ReadToEndAsync();`

`new Thread(() => {
 var reader = new StreamReader(data);
 outputText.Text = reader.ReadToEnd();
}).Start();`

`outputText.Text = await data.ReadAsync(new byte[1024], 0, -1);`

1.1.159.1 Correction

^ Explanation

You should use this code block:

```
var reader = new StreamReader(data);
outputText.Text = await reader.ReadToEndAsync();
```

The StreamReader class inherits several asynchronous methods from the TextReader base class. These methods include the Async suffix in the method name. Each asynchronous method returns a Task instance that may be awaited in your code. When you use the await keyword on a Task instance, the compiler constructs an internal state machine that uses the SynchronizationContext.Current static property to run the remainder of the method on the appropriate thread. In a WPF application, the current SynchronizationContext ensures that the return value of the task is processed on the user-interface (UI) thread.

You should not use this code block:

```
outputText.Text = await data.ReadAsync(new byte[1024], 0, 1024);
```

The Stream class directly exposes several asynchronous methods. The ReadAsync and corresponding WriteAsync methods are buffered operations. The ReadAsync method returns the number of bytes read, not the data that was read. In this example, the first 1024 bytes of the file are buffered in an anonymous byte array, and then the number of bytes read is returned as an integer. The Text property expects a string of content to display. The compiler gives an error due to the type mismatch.

You should not use this code block:

```
new Thread(() => {
    var reader = new StreamReader(data);
    outputText.Text = reader.ReadToEnd();
}).Start();
```

The Thread class does not perform any form of cross-thread synchronization. In this example, a new thread is created. The thread then creates a StreamReader, reads the entire stream into a string, and assigns that string to the Text property of the outputText UI object. Then the thread ends. Your code throws an exception when the background thread attempts to make a change to the UI object, because the UI thread reserves exclusive access to all UI objects.

You should not use this code block:

```
outputText.Dispatcher.BeginInvoke(() => {
    var reader = new StreamReader(data);
    outputText.Text = reader.ReadToEnd();
})
```

This code uses the BeginInvoke method of the Dispatcher instance attached to the UI object. This is a feature of the WPF object model designed to make it easier to run code on the UI thread. The BeginInvoke method name resembles the common Asynchronous Programming Model (APM) pattern. However, the BeginInvoke method actually adds a method to a queue of methods that the UI thread executes after rendering each frame of the user interface. Each method queued with BeginInvoke must still have a short execution time (less than 33ms if the UI is to update at 30 frames per second) because they execute on the UI thread. A long-running operation, such as decryption and file input/output, should be executed on a background thread.