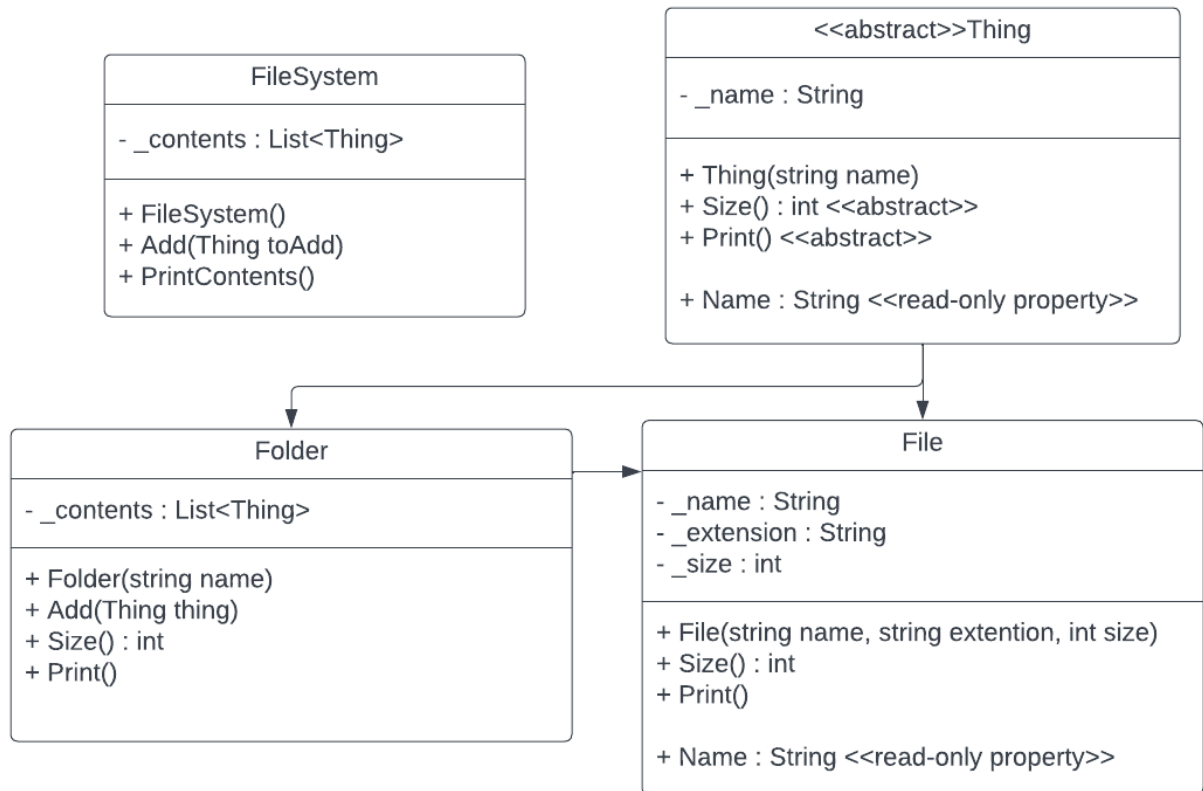


SWINBURNE UNIVERSITY OF TECHNOLOGY

COS20007 OBJECT ORIENTED PROGRAMMING

Semester test

PDF generated at 15:47 on Thursday 28th September, 2023



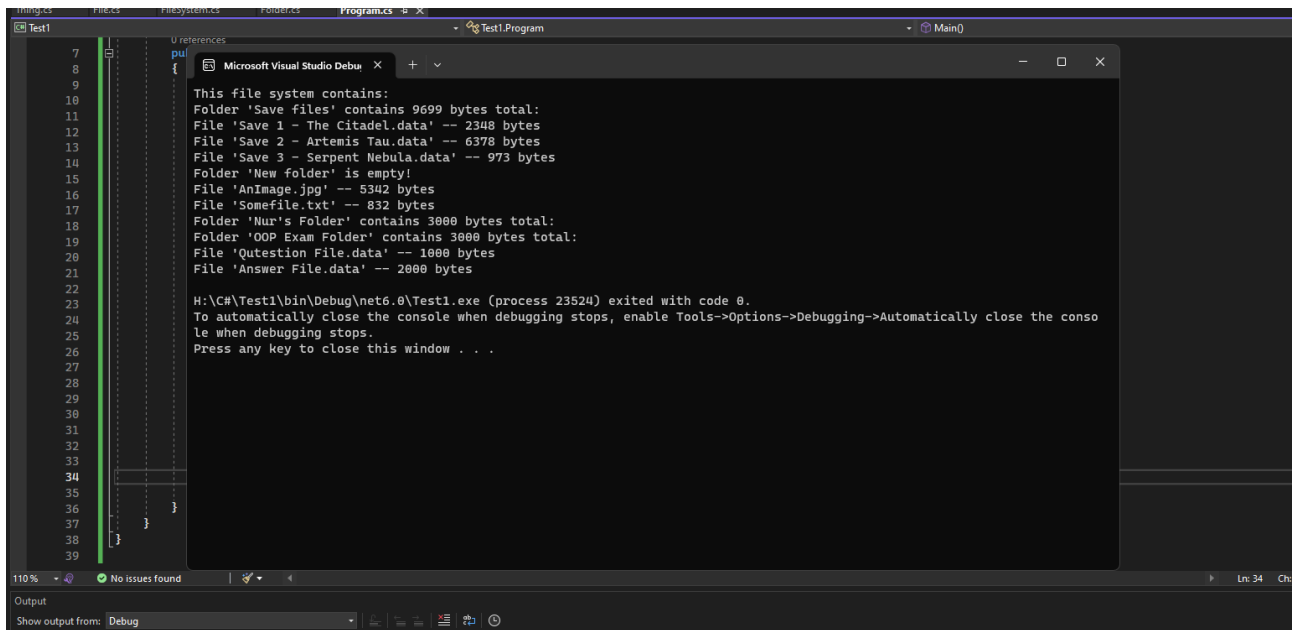
```
1  using System;
2
3  namespace Test1
4  {
5      public class Program
6      {
7          public static void Main()
8          {
9              FileSystem fs = new FileSystem();
10
11              File file1 = new File("AnImage", "jpg", 5342);
12              File file2 = new File("Somefile", "txt", 832);
13
14              Folder saveFiles = new Folder("Save files");
15              saveFiles.AddThing(new File("Save 1 - The Citadel", "data", 2348));
16              saveFiles.AddThing(new File("Save 2 - Artemis Tau", "data", 6378));
17              saveFiles.AddThing(new File("Save 3 - Serpent Nebula", "data", 973));
18
19              Folder newFolder = new Folder("New folder");
20
21              // Adding a folder that contains another folder which contains files.
22              Folder myFolder = new Folder("Nur's Folder");
23              Folder examFolder = new Folder("OOP Exam Folder");
24              examFolder.AddThing(new File("Qutestion File", "data", 1000));
25              examFolder.AddThing(new File("Answer File", "data", 2000));
26              myFolder.AddThing(examFolder);
27
28              fs.Add(saveFiles);
29              fs.Add(newFolder);
30              fs.Add(file1);
31              fs.Add(file2);
32              fs.Add(myFolder);
33
34              fs.PrintContents();
35
36          }
37      }
38  }
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Test1
8  {
9
10     public class FileSystem
11     {
12         private List<Thing> _contents = new List<Thing>();
13
14         public FileSystem()
15         {
16         }
17
18         public void Add(Thing toAdd)
19         {
20             _contents.Add(toAdd);
21         }
22
23         public void PrintContents()
24         {
25             Console.WriteLine("This file system contains:");
26
27             foreach (var item in _contents)
28             {
29                 item.Print();
30             }
31         }
32     }
33
34
35
36 }
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Test1
8  {
9      public abstract class Thing
10     {
11         private string _name;
12
13         public Thing(string name)
14         {
15             _name = name;
16         }
17
18         public abstract int Size();
19         public abstract void Print();
20
21         public string Name
22         {
23             get
24             {
25                 return _name;
26             }
27         }
28     }
29
30
31 }
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Test1
8  {
9
10     public class Folder : Thing
11     {
12         private List<Thing> _contents = new List<Thing>();
13
14         public Folder(string name) : base(name)
15         {
16         }
17
18         public void AddThing(Thing thing)
19         {
20             _contents.Add(thing);
21         }
22
23         public override int Size()
24         {
25             int totalSize = 0;
26             foreach (var file in _contents)
27             {
28                 totalSize += file.Size();
29             }
30             return totalSize;
31         }
32
33         public override void Print()
34         {
35             if (_contents.Count == 0)
36             {
37                 Console.WriteLine($"Folder '{Name}' is empty!");
38             }
39             else
40             {
41                 Console.WriteLine($"Folder '{Name}' contains {Size()} bytes total:");
42                 foreach (var file in _contents)
43                 {
44                     file.Print();
45                 }
46             }
47         }
48     }
49
50
51
52 }
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Test1
8  {
9      public class File : Thing
10     {
11         private string _name;
12         private string _extension;
13         private int _size;
14
15         public File(string name, string extension, int size) : base(name)
16         {
17             _name = name;
18             _extension = extension;
19             _size = size;
20         }
21
22         public override int Size()
23         {
24             return _size;
25         }
26
27         public override void Print()
28         {
29             Console.WriteLine($"File '{_name}.{_extension}' -- {_size} bytes");
30         }
31
32         public new string Name
33         {
34             get
35             {
36                 return _name;
37             }
38         }
39     }
40
41 }
42 }
```



The screenshot shows the Visual Studio Code interface with a C# program being debugged. The code in the editor defines a file system with folders and files, their sizes, and a list of files. The program then prints the contents of this file system and exits. The debug console shows the output of the program, which matches the code's logic. The console also displays the exit message and a prompt to close the window.

```
7  1  namespace Test1
8  2  {
9  3  4
10 4  5
11 5  6
12 6  7
13 7  8
14 8  9
15 9 10
16 10 11
17 11 12
18 12 13
19 13 14
20 14 15
21 15 16
22 16 17
23 17 18
24 18 19
25 19 20
26 20 21
27 21 22
28 22 23
29 23 24
30 24 25
31 25 26
32 26 27
33 27 28
34 28 29
35 29 30
36 30 31
37 31 32
38 32 33
39 33 34
40 34 35
41 35 36
42 36 37
43 37 38
44 38 39
45 39 40
46 40 41
47 41 42
48 42 43
49 43 44
50 44 45
51 45 46
52 46 47
53 47 48
54 48 49
55 49 50
56 50 51
57 51 52
58 52 53
59 53 54
60 54 55
61 55 56
62 56 57
63 57 58
64 58 59
65 59 60
66 60 61
67 61 62
68 62 63
69 63 64
70 64 65
71 65 66
72 66 67
73 67 68
74 68 69
75 69 70
76 70 71
77 71 72
78 72 73
79 73 74
80 74 75
81 75 76
82 76 77
83 77 78
84 78 79
85 79 80
86 80 81
87 81 82
88 82 83
89 83 84
90 84 85
91 85 86
92 86 87
93 87 88
94 88 89
95 89 90
96 90 91
97 91 92
98 92 93
99 93 94
100 94 95
101 95 96
102 96 97
103 97 98
104 98 99
105 99 100
106 100 101
107 101 102
108 102 103
109 103 104
110 104 105
111 105 106
112 106 107
113 107 108
114 108 109
115 109 110
116 110 111
117 111 112
118 112 113
119 113 114
120 114 115
121 115 116
122 116 117
123 117 118
124 118 119
125 119 120
126 120 121
127 121 122
128 122 123
129 123 124
130 124 125
131 125 126
132 126 127
133 127 128
134 128 129
135 129 130
136 130 131
137 131 132
138 132 133
139 133 134
140 134 135
141 135 136
142 136 137
143 137 138
144 138 139
145 139 140
146 140 141
147 141 142
148 142 143
149 143 144
150 144 145
151 145 146
152 146 147
153 147 148
154 148 149
155 149 150
156 150 151
157 151 152
158 152 153
159 153 154
160 154 155
161 155 156
162 156 157
163 157 158
164 158 159
165 159 160
166 160 161
167 161 162
168 162 163
169 163 164
170 164 165
171 165 166
172 166 167
173 167 168
174 168 169
175 169 170
176 170 171
177 171 172
178 172 173
179 173 174
180 174 175
181 175 176
182 176 177
183 177 178
184 178 179
185 179 180
186 180 181
187 181 182
188 182 183
189 183 184
190 184 185
191 185 186
192 186 187
193 187 188
194 188 189
195 189 190
196 190 191
197 191 192
198 192 193
199 193 194
200 194 195
201 195 196
202 196 197
203 197 198
204 198 199
205 199 200
206 200 201
207 201 202
208 202 203
209 203 204
210 204 205
211 205 206
212 206 207
213 207 208
214 208 209
215 209 210
216 210 211
217 211 212
218 212 213
219 213 214
220 214 215
221 215 216
222 216 217
223 217 218
224 218 219
225 219 220
226 220 221
227 221 222
228 222 223
229 223 224
230 224 225
231 225 226
232 226 227
233 227 228
234 228 229
235 229 230
236 230 231
237 231 232
238 232 233
239 233 234
240 234 235
241 235 236
242 236 237
243 237 238
244 238 239
245 239 240
246 240 241
247 241 242
248 242 243
249 243 244
250 244 245
251 245 246
252 246 247
253 247 248
254 248 249
255 249 250
256 250 251
257 251 252
258 252 253
259 253 254
260 254 255
261 255 256
262 256 257
263 257 258
264 258 259
265 259 260
266 260 261
267 261 262
268 262 263
269 263 264
270 264 265
271 265 266
272 266 267
273 267 268
274 268 269
275 269 270
276 270 271
277 271 272
278 272 273
279 273 274
280 274 275
281 275 276
282 276 277
283 277 278
284 278 279
285 279 280
286 280 281
287 281 282
288 282 283
289 283 284
290 284 285
291 285 286
292 286 287
293 287 288
294 288 289
295 289 290
296 290 291
297 291 292
298 292 293
299 293 294
300 294 295
301 295 296
302 296 297
303 297 298
304 298 299
305 299 300
306 300 301
307 301 302
308 302 303
309 303 304
310 304 305
311 305 306
312 306 307
313 307 308
314 308 309
315 309 310
316 310 311
317 311 312
318 312 313
319 313 314
320 314 315
321 315 316
322 316 317
323 317 318
324 318 319
325 319 320
326 320 321
327 321 322
328 322 323
329 323 324
330 324 325
331 325 326
332 326 327
333 327 328
334 328 329
335 329 330
336 330 331
337 331 332
338 332 333
339 333 334
340 334 335
341 335 336
342 336 337
343 337 338
344 338 339
345 339 340
346 340 341
347 341 342
348 342 343
349 343 344
350 344 345
351 345 346
352 346 347
353 347 348
354 348 349
355 349 350
356 350 351
357 351 352
358 352 353
359 353 354
360 354 355
361 355 356
362 356 357
363 357 358
364 358 359
365 359 360
366 360 361
367 361 362
368 362 363
369 363 364
370 364 365
371 365 366
372 366 367
373 367 368
374 368 369
375 369 370
376 370 371
377 371 372
378 372 373
379 373 374
380 374 375
381 375 376
382 376 377
383 377 378
384 378 379
385 379 380
386 380 381
387 381 382
388 382 383
389 383 384
390 384 385
391 385 386
392 386 387
393 387 388
394 388 389
395 389 390
396 390 391
397 391 392
398 392 393
399 393 394
400 394 395
401 395 396
402 396 397
403 397 398
404 398 399
405 399 400
406 400 401
407 401 402
408 402 403
409 403 404
410 404 405
411 405 406
412 406 407
413 407 408
414 408 409
415 409 410
416 410 411
417 411 412
418 412 413
419 413 414
420 414 415
421 415 416
422 416 417
423 417 418
424 418 419
425 419 420
426 420 421
427 421 422
428 422 423
429 423 424
430 424 425
431 425 426
432 426 427
433 427 428
434 428 429
435 429 430
436 430 431
437 431 432
438 432 433
439 433 434
440 434 435
441 435 436
442 436 437
443 437 438
444 438 439
445 439 440
446 440 441
447 441 442
448 442 443
449 443 444
450 444 445
451 445 446
452 446 447
453 447 448
454 448 449
455 449 450
456 450 451
457 451 452
458 452 453
459 453 454
460 454 455
461 455 456
462 456 457
463 457 458
464 458 459
465 459 460
466 460 461
467 461 462
468 462 463
469 463 464
470 464 465
471 465 466
472 466 467
473 467 468
474 468 469
475 469 470
476 470 471
477 471 472
478 472 473
479 473 474
480 474 475
481 475 476
482 476 477
483 477 478
484 478 479
485 479 480
486 480 481
487 481 482
488 482 483
489 483 484
490 484 485
491 485 486
492 486 487
493 487 488
494 488 489
495 489 490
496 490 491
497 491 492
498 492 493
499 493 494
500 494 495
501 495 496
502 496 497
503 497 498
504 498 499
505 499 500
506 500 501
507 501 502
508 502 503
509 503 504
510 504 505
511 505 506
512 506 507
513 507 508
514 508 509
515 509 510
516 510 511
517 511 512
518 512 513
519 513 514
520 514 515
521 515 516
522 516 517
523 517 518
524 518 519
525 519 520
526 520 521
527 521 522
528 522 523
529 523 524
530 524 525
531 525 526
532 526 527
533 527 528
534 528 529
535 529 530
536 530 531
537 531 532
538 532 533
539 533 534
540 534 535
541 535 536
542 536 537
543 537 538
544 538 539
545 539 540
546 540 541
547 541 542
548 542 543
549 543 544
550 544 545
551 545 546
552 546 547
553 547 548
554 548 549
555 549 550
556 550 551
557 551 552
558 552 553
559 553 554
560 554 555
561 555 556
562 556 557
563 557 558
564 558 559
565 559 560
566 560 561
567 561 562
568 562 563
569 563 564
570 564 565
571 565 566
572 566 567
573 567 568
574 568 569
575 569 570
576 570 571
577 571 572
578 572 573
579 573 574
580 574 575
581 575 576
582 576 577
583 577 578
584 578 579
585 579 580
586 580 581
587 581 582
588 582 583
589 583 584
590 584 585
591 585 586
592 586 587
593 587 588
594 588 589
595 589 590
596 590 591
597 591 592
598 592 593
599 593 594
600 594 595
601 595 596
602 596 597
603 597 598
604 598 599
605 599 600
606 600 601
607 601 602
608 602 603
609 603 604
610 604 605
611 605 606
612 606 607
613 607 608
614 608 609
615 609 610
616 610 611
617 611 612
618 612 613
619 613 614
620 614 615
621 615 616
622 616 617
623 617 618
624 618 619
625 619 620
626 620 621
627 621 622
628 622 623
629 623 624
630 624 625
631 625 626
632 626 627
633 627 628
634 628 629
635 629 630
636 630 631
637 631 632
638 632 633
639 633 634
640 634 635
641 635 636
642 636 637
643 637 638
644 638 639
645 639 640
646 640 641
647 641 642
648 642 643
649 643 644
650 644 645
651 645 646
652 646 647
653 647 648
654 648 649
655 649 650
656 650 651
657 651 652
658 652 653
659 653 654
660 654 655
661 655 656
662 656 657
663 657 658
664 658 659
665 659 660
666 660 661
667 661 662
668 662 663
669 663 664
670 664 665
671 665 666
672 666 667
673 667 668
674 668 669
675 669 670
676 670 671
677 671 672
678 672 673
679 673 674
680 674 675
681 675 676
682 676 677
683 677 678
684 678 679
685 679 680
686 680 681
687 681 682
688 682 683
689 683 684
690 684 685
691 685 686
692 686 687
693 687 688
694 688 689
695 689 690
696 690 691
697 691 692
698 692 693
699 693 694
700 694 695
701 695 696
702 696 697
703 697 698
704 698 699
705 699 700
706 700 701
707 701 702
708 702 703
709 703 704
710 704 705
711 705 706
712 706 707
713 707 708
714 708 709
715 709 710
716 710 711
717 711 712
718 712 713
719 713 714
720 714 715
721 715 716
722 716 717
723 717 718
724 718 719
725 719 720
726 720 721
727 721 722
728 722 723
729 723 724
730 724 725
731 725 726
732 726 727
733 727 728
734 728 729
735 729 730
736 730 731
737 731 732
738 732 733
739 733 734
740 734 735
741 735 736
742 736 737
743 737 738
744 738 739
745 739 740
746 740 741
747 741 742
748 742 743
749 743 744
750 744 745
751 745 746
752 746 747
753 747 748
754 748 749
755 749 750
756 750 751
757 751 752
758 752 753
759 753 754
760 754 755
761 755 756
762 756 757
763 757 758
764 758 759
765 759 760
766 760 761
767 761 762
768 762 763
769 763 764
770 764 765
771 765 766
772 766 767
773 767 768
774 768 769
775 769 770
776 770 771
777 771 772
778 772 773
779 773 774
780 774 775
781 775 776
782 776 777
783 777 778
784 778 779
785 779 780
786 780 781
787 781 782
788 782 783
789 783 784
790 784 785
791 785 786
792 786 787
793 787 788
794 788 789
795 789 790
796 790 791
797 791 792
798 792 793
799 793 794
800 794 795
801 795 796
802 796 797
803 797 798
804 798 799
805 799 800
806 800 801
807 801 802
808 802 803
809 803 804
810 804 805
811 805 806
812 806 807
813 807 808
814 808 809
815 809 810
816 810 811
817 811 812
818 812 813
819 813 814
820 814 815
821 815 816
822 816 817
823 817 818
824 818 819
825 819 820
826 820 821
827 821 822
828 822 823
829 823 824
830 824 825
831 825 826
832 826 827
833 827 828
834 828 829
835 829 830
836 830 831
837 831 832
838 832 833
839 833 834
840 834 835
841 835 836
842 836 837
843 837 838
844 838 839
845 839 840
846 840 841
847 841 842
848 842 843
849 843 844
850 844 845
851 845 846
852 846 847
853 847 848
854 848 849
855 849 850
856 850 851
857 851 852
858 852 853
859 853 854
860 854 855
861 855 856
862 856 857
863 857 858
864 858 859
865 859 860
866 860 861
867 861 862
868 862 863
869 863 864
870 864 865
871 865 866
872 866 867
873 867 868
874 868 869
875 869 870
876 870 871
877 871 872
878 872 873
879 873 874
880 874 875
881 875 876
882 876 877
883 877 878
884 878 879
885 879 880
886 880 881
887 881 882
888 882 883
889 883 884
890 884 885
891 885 886
892 886 887
893 887 888
894 888 889
895 889 890
896 890 891
897 891 892
898 892 893
899 893 894
900 894 895
901 895 896
902 896 897
903 897 898
904 898 899
905 899 900
906 900 901
907 901 902
908 902 903
909 903 904
910 904 905
911 905 906
912 906 907
913 907 908
914 908 909
915 909 910
916 910 911
917 911 912
918 912 913
919 913 914
920 914 915
921 915 916
922 916 917
923 917 918
924 918 919
925 919 920
926 920 921
927 921 922
928 922 923
929 923 924
930 924 925
931 925 926
932 926 927
933 927 928
934 928 929
935 929 930
936 930 931
937 931 932
938 932 933
939 933 934
940 934 935
941 935 936
942 936 937
943 937 938
944 938 939
945 939 940
946 940 941
947 941 942
948 942 943
949 943 944
950 944 945
951 945 946
952 946 947
953 947 948
954 948 949
955 949 950
956 950 951
957 951 952
958 952 953
959 953 954
960 954 955
961 955 956
962 956 957
963 957 958
964 958 959
965 959 960
966 960 961
967 961 962
968 962 963
969 963 964
970 964 965
971 965 966
972 966 967
973 967 968
974 968 969
975 969 970
976 970 971
977 971 972
978 972 973
979 973 974
980 974 975
981 975 976
982 976 977
983 977 978
984 978 979
985 979 980
986 980 981
987 981 982
988 982 983
989 983 984
990 984 985
991 985 986
992 986 987
993 987 988
994 988 989
995 989 990
996 990 991
997 991 992
998 992 993
999 993 994
1000 994 995
1001 995 996
1002 996 997
1003 997 998
1004 998 999
1005 999 1000
1006 1000 1001
1007 1001 1002
1008 1002 1003
1009 1003 1004
1010 1004 1005
1011 1005 1006
1012 1006 1007
1013 1007 1008
1014 1008 1009
1015 1009 1010
1016 1010 1011
1017 1011 1012
1018 1012 1013
1019 1013 1014
1020 1014 1015
1021 1015 1016
1022 1016 1017
1023 1017 1018
1024 1018 1019
1025 1019 1020
1026 1020 1021
1027 1021 1022
1028 1022 1023
1029 1023 1024
1030 1024 1025
1031 1025 1026
1032 1026 1027
1033 1027 1028
1034 1028 1029
1035 1029 1030
1036 1030 1031
103
```


1. Describe the principle of polymorphism and how it was used in Task 1

- **Polymorphism:** Polymorphism in OOP allows objects of various classes to be treated as if they were objects of the same superclass. This allows for the use of a single interface for several data types, increasing code reusability and flexibility.
- **Usage in Task 1:** Since the “Thing class” is abstract, both the File and Folder subclasses override its methods Size() and Print(). This demonstrates polymorphism by enabling the addition of both files and folders to the “_contents” list in the FileSystem and Folder classes.

2. Consider the FileSystem and Folder classes from the updated design in Task 1. Do we need both of these classes? Explain why or why not?

The FileSystem and Folder classes each play a unique role in Task 1, despite the fact that at first glance their functions may appear to overlap.

- **Purpose:** The FileSystem class, which stores numerous objects like files and directories and so represents a full file system, serves as an overarching framework. In contrast, the Folder class mimics real-world directory hierarchies by acting as a sub-structure that may include more files and sub-folders.
- **Scalability and Maintainability:** Separating them enables for more flexible and adaptable architecture. If one were to implement unique methods or characteristics relevant to either the file system or individual folders in future iterations (such as rights, ownership, or mounting capabilities for FileSystem), having discrete classes would make this process simple.

Therefore, despite the fact that both classes handle and store "Thing" objects, their responsibilities, scalability, and representation in a common file system point to the requirement of both types. Their separation follows the Single Responsibility Principle, which ensures that each class has only one cause to change.

3. What is wrong with the class name Thing? Suggest a better name for the class and explain the reasoning behind your answer.

The category name Particularly in the context of a file system, Thing is somewhat unclear and does not provide an intuitive idea of its function or the kinds of items it's designed to represent.

- **Specificity and Context:** One would anticipate class names that reflect real-world elements in a system that deals with files and folders. A class's purpose in representing entities in a file system cannot be understood by its general name, Thing.
- **Improved Name Suggestion:** FileSystemEntity or FileSystemItem could be more appropriate names for the Thing class. Both approaches give a more straightforward representation of the purpose of the class and provide clarity when developers or collaborators interact with the codebase.

By renaming the Thing class to something more descriptive, we ensure that the name of the class is more directly related to its purpose and function, enabling improved readability and maintainability.

4. Define the principle of abstraction and explain how you would use it to design a class to represent a Book.
 - **Abstraction:** Abstraction in OOP refers to the notion of concealing complicated implementation details and displaying only the basic properties of an object. It is concerned with simplifying complicated reality by modelling classes based on the traits and behaviors that an item should have.
 - **Application for a Book class:** If we were to use abstraction to create a class for a book, we would concentrate on the fundamental attributes and operations that characterise a book. The following are some examples of attributes and methods: ReadPage(), BookmarkPage(), ISBN, Title, Author, and PageCount. The colour of the book is an example of a non-essential element that we wouldn't add unless it was pertinent to the application at hand. By doing this, we are able to convey the core of what a book is and how it works without becoming mired down in pointless details.