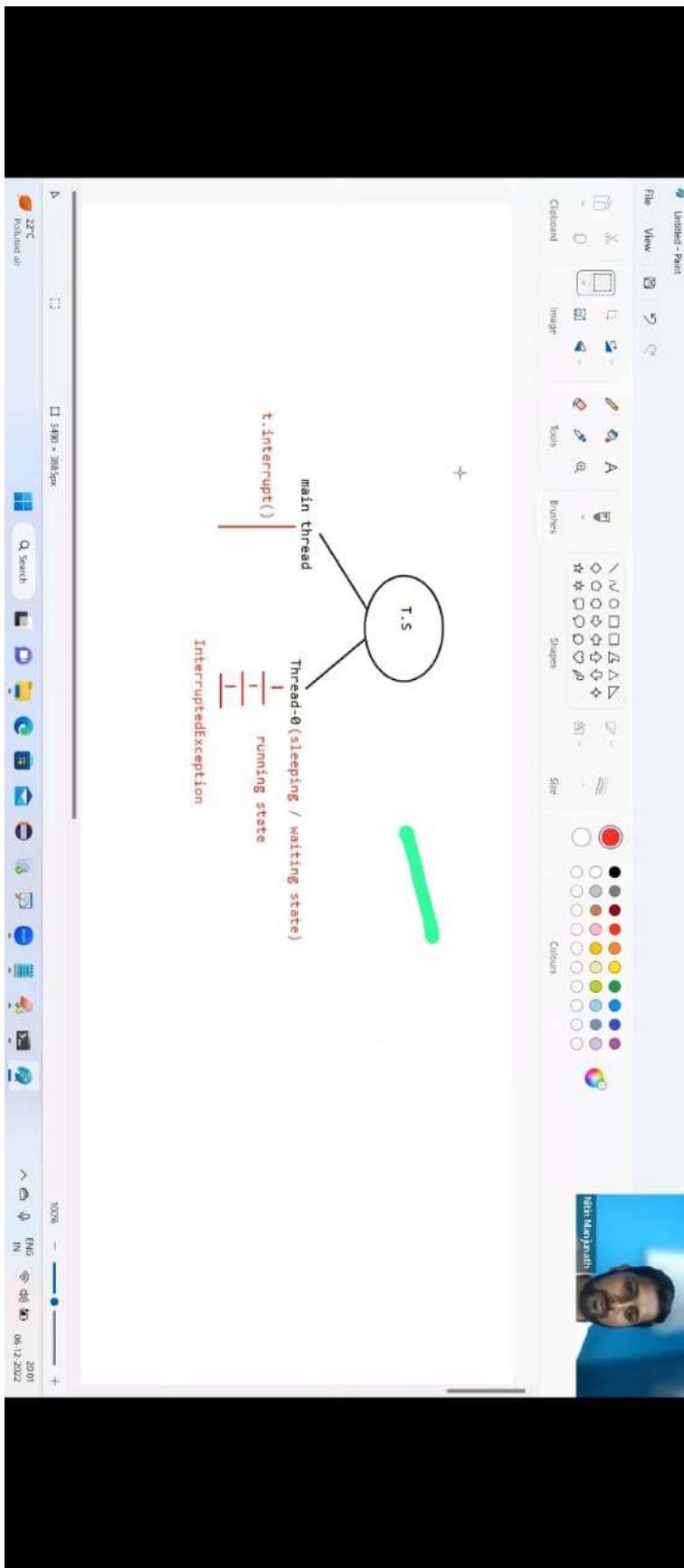


Java Multithreading Part4

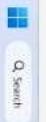


=> If thread is in sleeping state or in waiting state we can interrupt a thread.

```
eg#1.  
class MyThread extends Thread{  
    @Override  
    public void run(){  
        try{  
            for (int i=1;i<=10;i++){  
                System.out.println("I am lazy thread");  
                Thread.sleep(2000);  
            }  
        }  
        catch (InterruptedException e){  
            System.out.println("I got interrupted");  
        }  
    }  
}  
public class Test3 {
```



22°C
Haus

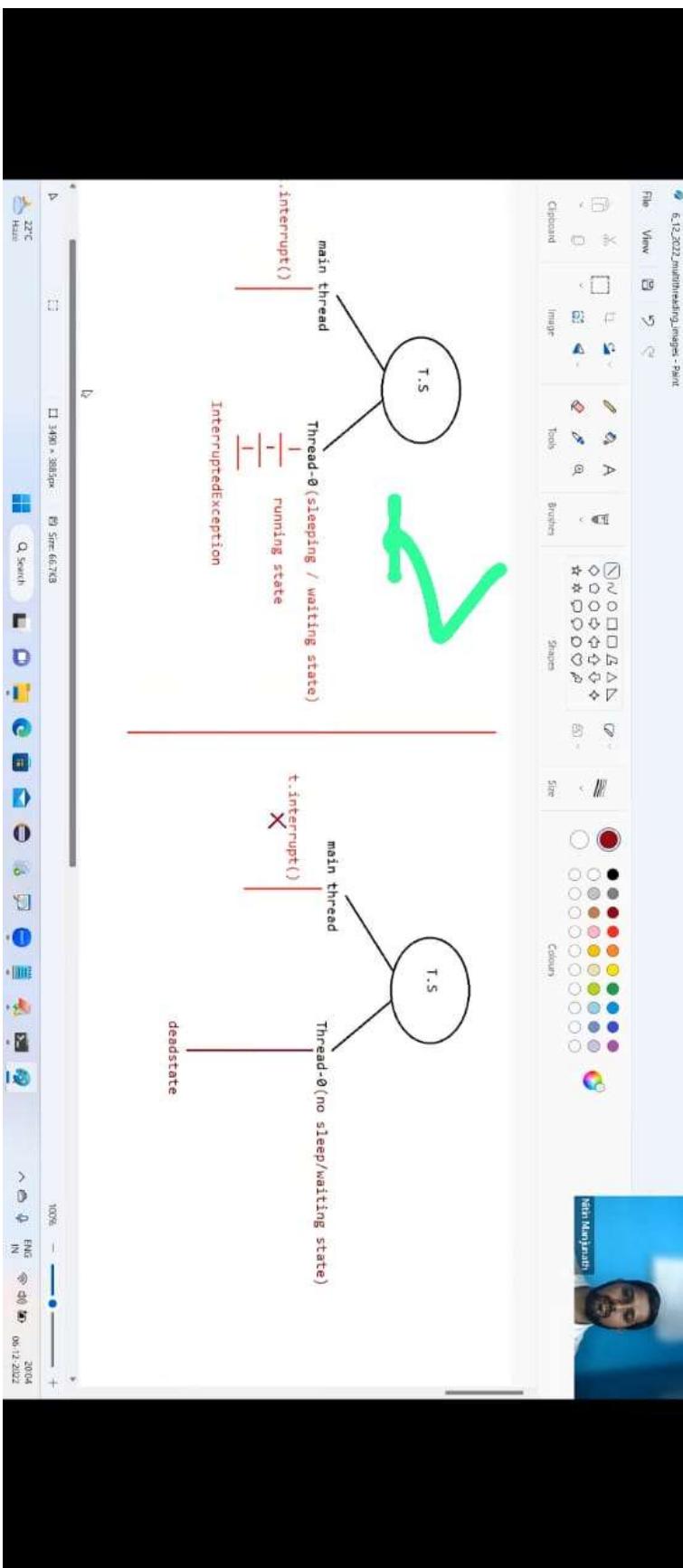


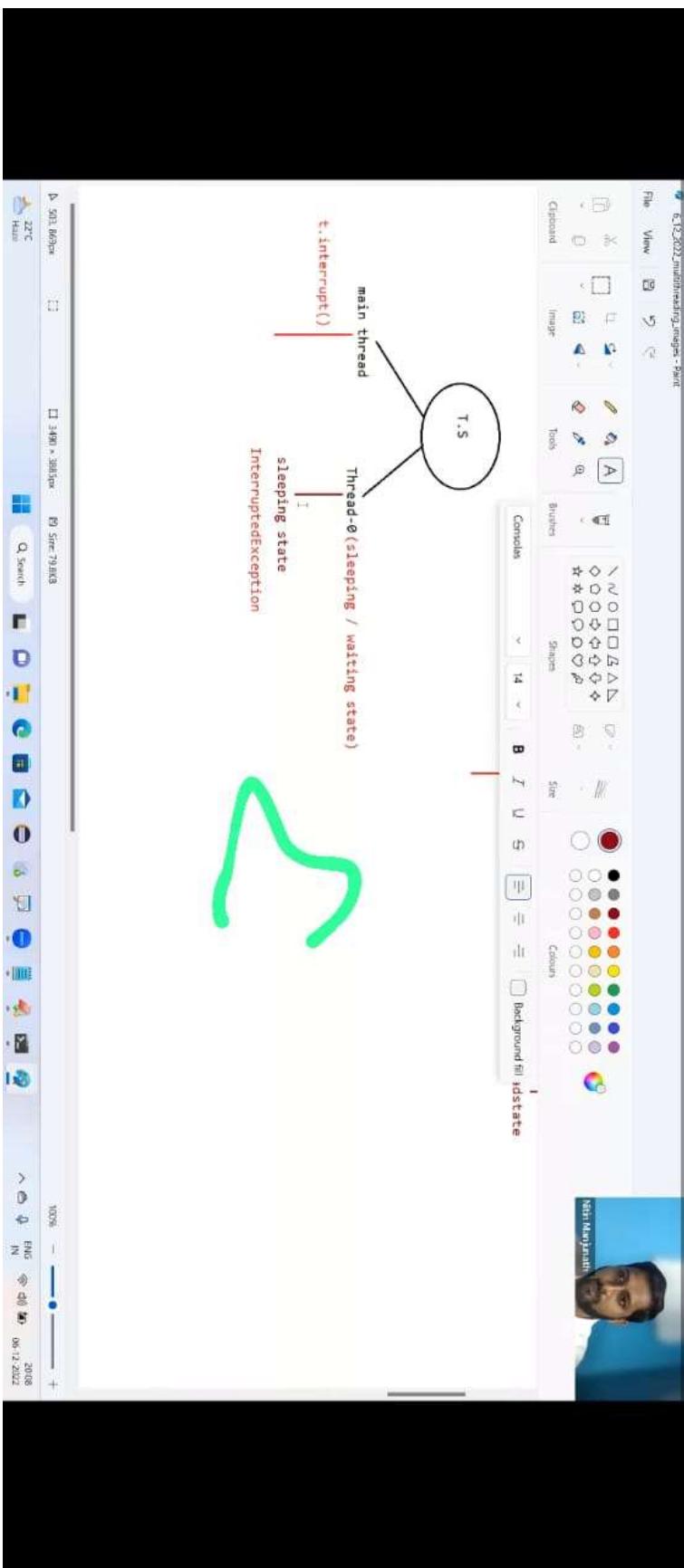
Ln: 90 Col: 174

TOP% Windows (CEP)

UTF-8

EN IN 06.12.2022







```
4.12.2022_multithreading_2dec_notes - Notepad
File Edit View
t.interrupt()://line-n1
System.out.println("main thread");
}
}

line-n1 is commented then no problem
line-n1 is not commented, then interrupt() will wait till the Thread enters into waiting state/sleeping state.

Note::
If thread is interrupting another thread, but target thread is not in waiting state/sleeping state then there would be no exception.
interrupt() call be waiting till the target thread enters into waiting state/sleeping state so this call wont be wasted.
once the target thread enters into waiting state/sleeping state then interrupt() will interrupt and it causes the exception.
interrupt() call will be wasted only if the Thread does not enters into waiting state/sleeping state.
```

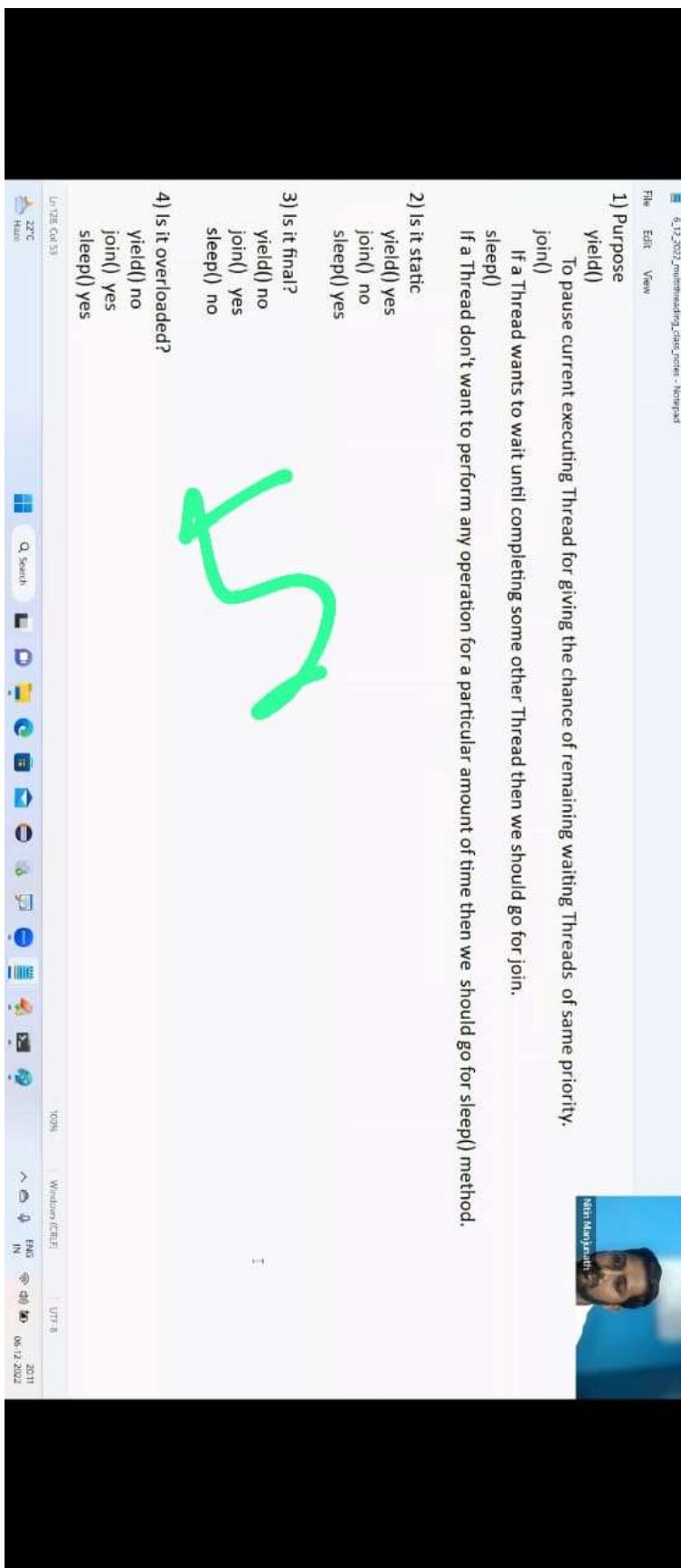
Un 97 Col 1

22°C Hau

Q. Search

TOPIK Windows (CEP) UTF-8

ENGLISH IN 06.12.2022



File Edit View
6.12.2022_multithreading_disc_notes - Notepad

If a Thread don't want to perform any operation for a particular amount of time then we should go for sleep() method

2) Is it static

yield() yes

join() no

sleep() yes

3) Is it final?

yield() no

join() yes

sleep() no

4) Is it overloaded?

yield() no

join() yes

sleep() yes

5) Is it throws I/E?

yield() no

join() yes

sleep() yes

6

22°C
Haze

Q. Search

T0% Windows (CEP) UTF-8

EN IN 05.12.2022

The screenshot shows a Java application running in a terminal window. The code is as follows:

```
public static void main(String[] args) {
    Runnable r = () -> {
        for (int i = 1; i <= 5; i++)
            System.out.println("child thread");
    };
    Thread t = new Thread(r);
    t.start();
}

for (int i = 1; i <= 5; i++)
    System.out.println("Main thread");
```

The output in the terminal window is:

```
child thread
child thread
child thread
child thread
child thread
Main thread
Main thread
Main thread
Main thread
Main thread
```

A large green checkmark is drawn over the terminal window, indicating the code is working correctly.

Erkhan - [D:\Test.java]

File Edit View Search Document Project Tools Browser Egret Window Help

Directory Object Functions

(D) New Volume 27

RECYLEBIN 28

CoronaEngine 29

Emulator 30

emulator 31

FieldPrograms 32

Front end vision 33

gitHub 34

gitment 35

gitrepo 36

IntelliJ 37

JAVA (Java) 38

Java 39

@ Test.java 40

@ Test.java 41

@ Test.java 42

@ Test.java 43

@ Test.java 44

@ Test.java 45

@ Test.java 46

@ Test.java 47

For Help press F1

22°C

Hair

1 2 3 4 5 6 7

```
public static void main(String[] args){  
    Runnable r = new Runnable(){  
        @Override  
        public void run(){  
            for (int i = 1;i<=5 ;i++ )  
            {  
                System.out.println("child thread");  
            }  
        }  
    };  
    Thread t = new Thread(r);  
    t.start();  
  
    for (int i = 1;i<=5 ;i++ )  
    {  
        System.out.println("Main thread");  
    }  
}
```

Nitin Manjrekar

In 35 col 54 32 20 PC ANSI EVALUATION VERSION

EN IN 05.12.2022

File Edit View Search Document Project Tools Browser Egret Window Help

Directory Clicked Functions

(D) New Volume SDCARD1IN C:\Windows

27 1 2 3 4 5 6 7

```
public static void main(String[] args){  
    Thread t = new Thread(new Runnable(){  
        @Override  
        public void run(){  
            for (int i = 1;i<=5 ;i++ )  
                System.out.println("child thread");  
        }  
    });  
    t.start();  
  
    for (int i = 1;i<=5 ;i++ )  
        System.out.println("Main thread");  
}
```

Java (Java) A0 In 39 col 16 53 03 PC ANSI Terminal Emulation Version: 0.6.12.2025
For Help press F1
22°C Hail

File Edit View Search Document Project Tools Browser Egret Window Help

Directory Clicked Functions

(D) New Volume SDCARD BIN

1 2 3 4 5 6 7

```
public static void main(String[] args)
{
    new Thread(new Runnable(){
        @Override
        public void run(){
            for (int i = 1;i<=5 ;i++ )
            {
                System.out.println("child thread");
            }
        }
    }).start();

}

for (int i = 1;i<=5 ;i++ )
{
    System.out.println("Main thread");
}
```

Java (Java) ADO JavaHelp preview 22°C Home

File Search Back Forward Stop Refresh Home

6.12.2022_maintaining_clean_code - Normal
Edit View

Note::using lambda expression
Runnable r = ()-> {

```
Thread t = new Thread(r);
t.start();
}
for (int i = 1; i<=5; i++)
{
    System.out.println("child thread");
}
```



```
6.12.2022_multithreading_2nd_exercise - Notepad  
File Edit View  
Thread t = new Thread(r);  
t.start();  
  
using anonymous inner class  
=====  
new Thread(new Runnable(){  
    @Override  
    public void run(){  
        for (int i = 1;i<=5 ;i++)  
        {  
            System.out.println("child thread");  
        }  
    }  
}).start();
```



22°C
Haus

Q. Search



Ln: 145 Col: 5
TOPK Windows (CEP7) UTF-8
EN IN 06.12.2022



The screenshot shows a Java code editor with the following code:

```
1  // The End View Search Document Project Tools browser Format Window Help
2  (b) New Volume
3  (b) [D] RECYCLE BIN
4  (b) CoreWAVEactive
5  (b) Enterprise
6  (b) Frontend
7  (b) Frontend
8  (b) Frontend
9  (b) Frontend
10 (b) Frontend
11 (b) Frontend
12 (b) Frontend
13 (b) Frontend
14 (b) Frontend
15 (b) Frontend
16 (b) Frontend
17 (b) Frontend
18 (b) Frontend
19 (b) Frontend
20 (b) Frontend
21 (b) Frontend
22 (b) Frontend
23 (b) Frontend
24 (b) Frontend
25 (b) Frontend
26 (b) Frontend
27 27 public static void main(String[] args)
28 {
29     new Thread(()-> {
30         for (int i = 1;i<=5 ;i++ )
31             System.out.println("Main thread");
32     }
33     ) .start();
34 }
35
36
37
38
39
40     for (int i = 1;i<=5 ;i++ )
41     {
42         System.out.println("Main thread");
43     }
44
45
46
47 }
```

A large green hand-drawn arrow points from the bottom left towards the `System.out.println("Main thread");` line at line 33.

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

```

File Edit View Search Document Project Tools Favorites Format Window Help

1 2 3 4 5 6 7

Java (*.java)

① Test.java ② Test.java ③ Test.java

For Help press F1

22°C Haze

1 2 3 4 5 6 7

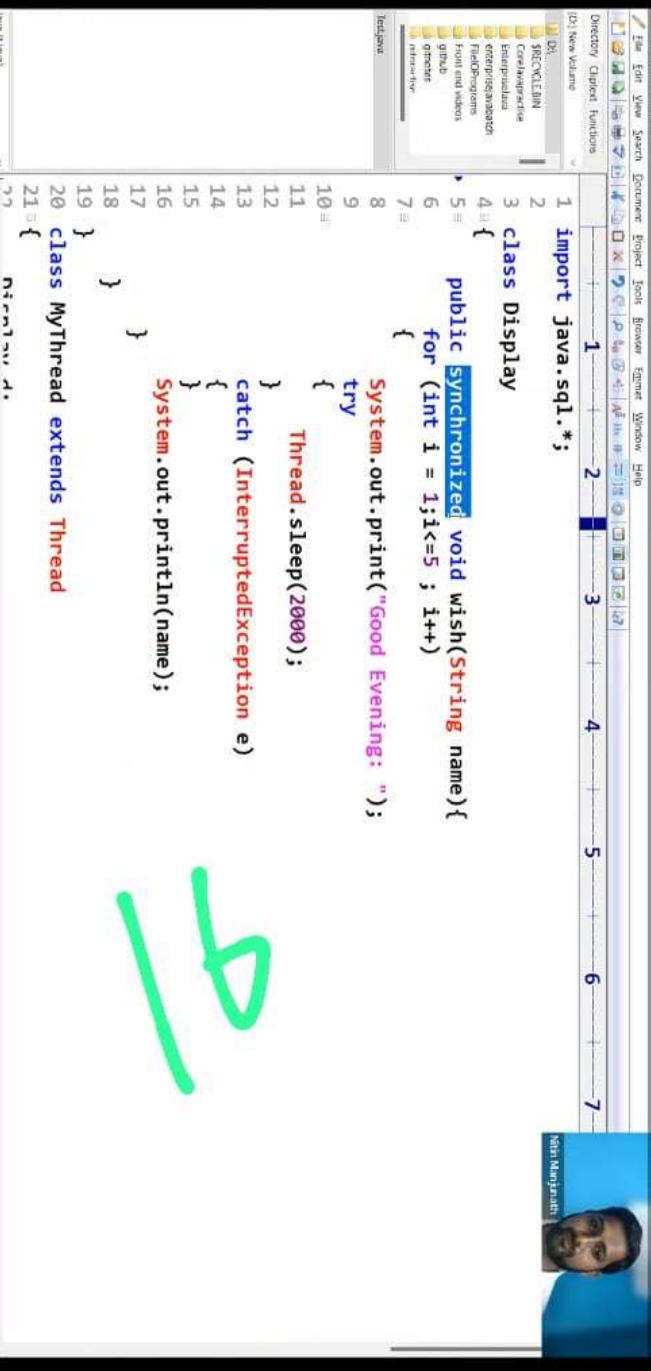
```
34 }
35
36 class Test
37 {
38     public static void main(String[] args)
39     {
40         Display d = new Display();
41         MyThread t1 =new MyThread(d, "sachin");
42         MyThread t2 =new MyThread(d, "dhoni");
43
44         t1.start();
45
46     }
47
48 }
```

Java Main Method

In 45 col 70 45 03 PC ANSI EVALUATION REPORT

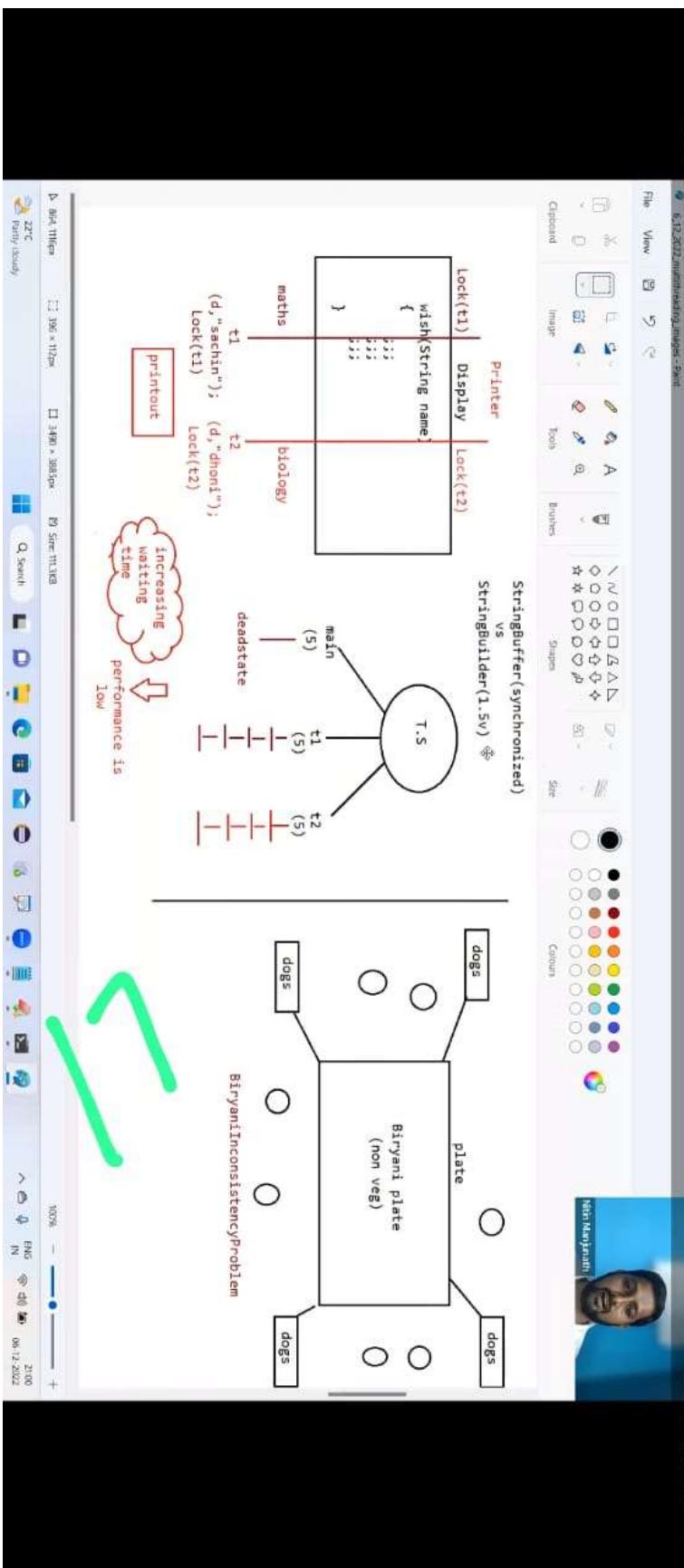
EN IN 05/12/2022 05:12:2022





```
File Edit View Search Document Project Tool Browser Format Window Help
New (Ctrl+N)
Open (Ctrl+O)
Save (Ctrl+S)
Print (Ctrl+P)
Exit (Alt+F4)
RECYCLE BIN
C:\Windows\Temp
Entire workspace
Front end workspace
gdb
grapes
rhino engine
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
import java.sql.*;
class Display
{
    public synchronized void wish(String name){
        System.out.println("Good Evening: ");
        try
        {
            Thread.sleep(2000);
        }
        catch (InterruptedException e)
        {
        }
    }
}
System.out.println(name);
}
class MyThread extends Thread
{
    public void run()
    {
        wish("Amit");
    }
}

```



```
6.12.2022 miniprojects - Java - NetBeans
```

```
File Edit View
```

```
System.out.println("child thread");
```

```
}
```

```
}).start();
```

```
synchronization
```

```
=====
```

```
1. synchronized is a keyword applicable only for methods and blocks
```

```
2. if we declare a method/block as synchronized then at a time only one thread can execute that method/block on that object.
```

```
3. The main advantage of synchronized keyword is we can resolve data inconsistency problems.
```

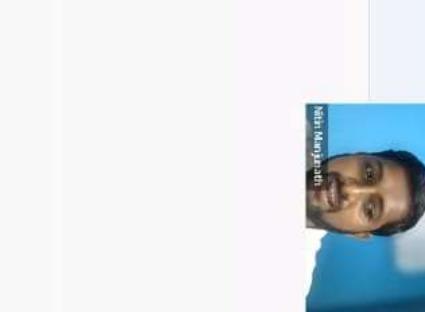
```
4. But the main disadvantage of synchronized keyword is it increases waiting time of the Thread and effects performance of the system.
```

```
5. Hence if there is no specific requirement then never recommended to use synchronized keyword.
```

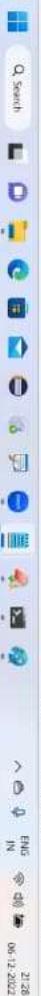
```
6. Internally synchronization concept is implemented by using lock concept.
```



```
4.12.2022_multithreading_discusses - Notepad  
File Edit View  
-----  
}  
  
Note:  
class X{  
    synchronized void m1(){  
        synchronized void m2(){  
            void m3(){  
        }  
    }  
}
```



Line 217 Col B



Top Windows (CEP) | UTF-8

21:28

06.12.2022

1. if t1 thread invokes m1() then on the Object X lock will applied.
 2. if t2 thread invokes m2() then m2() can't be called because lock of X object is with m1.
 3. if t3 thread invokes m3() then execution will happen becoz m3() is non-synchronized.
- Lock concept is applied at the Object level not at the method level.

Threads - Notepad

File Edit View

KeyPoints

=====

1. if t1 thread invokes m1() then on the Object X lock will applied.

2. if t2 thread invokes m2() then m2() can't be called because lock of X object is with m1.

3. if t3 thread invokes m3() then execution will happen becoz m3() is non-synchronized.

Lock concept is applied at the Object level not at the method level.

7. Every object in java has a unique lock. Whenever we are using synchronized keyword then only lock concept will come into the picture.

8. If a Thread wants to execute any synchronized method on the given object 1st it has to get the lock of that object. Once a Thread got the lock of that object then it's allow to execute any synchronized method on that object. If the synchronized method execution completes then automatically Thread releases lock.

9. While a Thread executing any synchronized method the remaining Threads are not allowed execute any synchronized method on that object simultaneously. But remaining Threads are allowed to execute any non-synchronized method simultaneously. [lock concept is implemented based on object but not based on method].

Note:- Every object will have 2 areas[Synchronized area and NonSynchronized area]

Synchronized Area => write the code only to perform update,insert,delete

NonSynchronized Area => write the code only to perform select operation

9

22°C
Partly cloudy

100% Windows (C:\EF7) UTR-8

EN IN 05.12.2022 21:29

File Edit View
4.12.2022_multithreading_zine notes - Notepad

simultaneously. [lock concept is implemented based on object but not based on method].

Note:: Every object will have 2 area[Syncronized area and NonSyncronized area]

Synchronized Area => write the code only to perform update,insert,delete

NonSyncronized Area => write the code only to perform select operation

```
class ReservationApp{  
    checkAvailability(){  
        }  
        //perform read operation  
    synchronized bookTicket(){  
        }  
        //perform update operation  
    }  
}
```

D
2

Ln 243 Col 1
21°C
Polluted air

Q Search

10%

Windows (CEP)

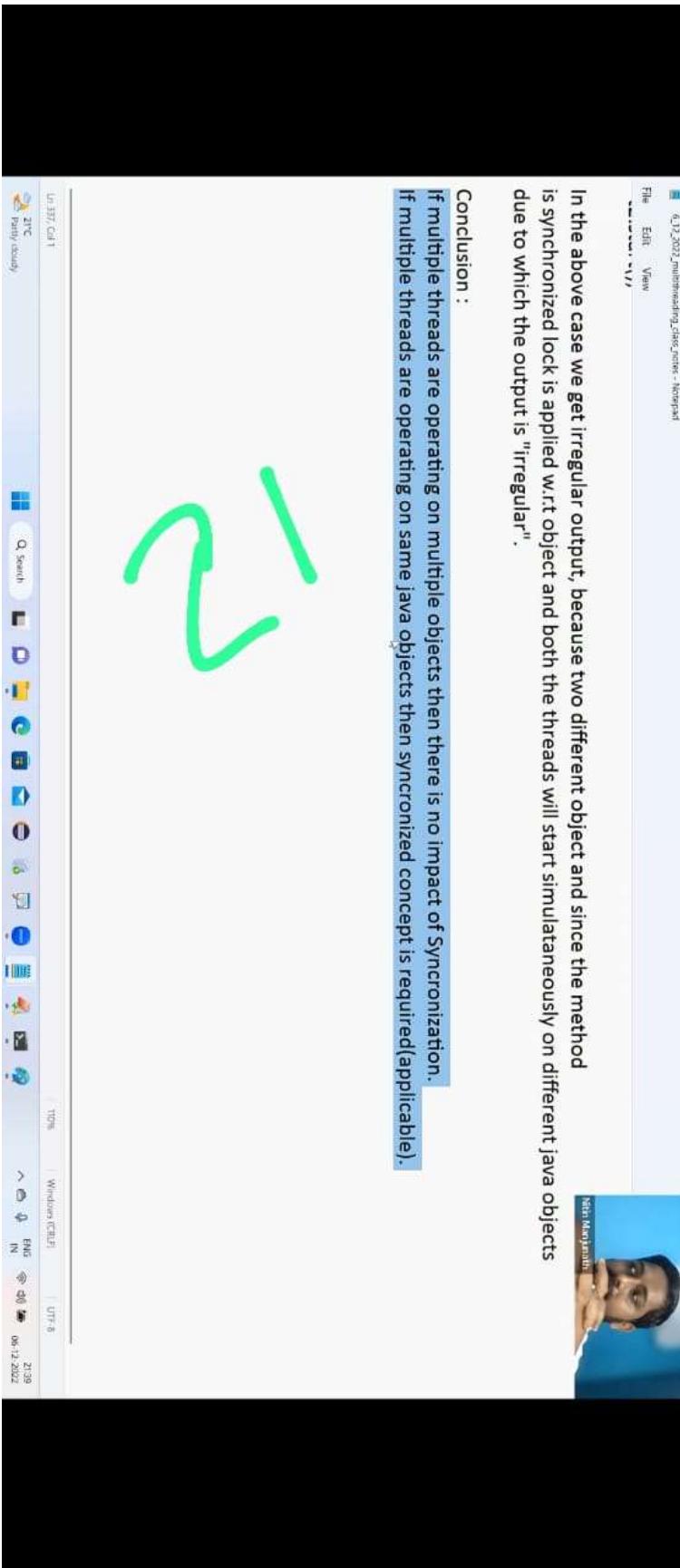
UTF-8

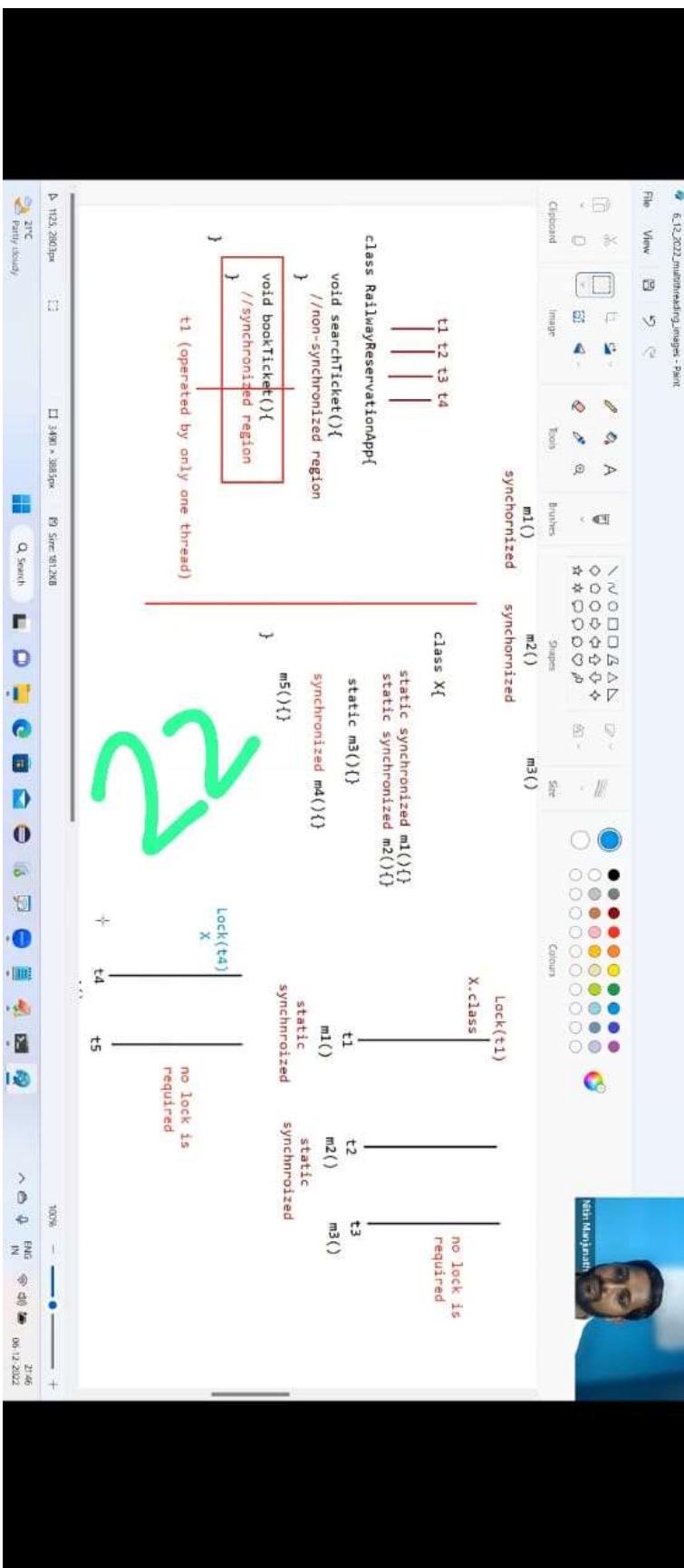
21:37

EN

IN

05.12.2022





File Edit View
6.17.2012 - multiplying class_size - Microsoft Word

1. Every class in java has a unique level lock.
2. If a thread wants to execute static synchronized method then the thread requires "class level lock".
3. While a Thread executing any static synchronized method the remaining Threads are not allowed to execute any static synchronized method of that class simultaneously.
4. But remaining Threads are allowed to execute normal synchronized methods, normal static methods, and normal instance methods simultaneously.
5. Class level lock and object lock both are different and there is no relationship between these two.

class X{
 eg..

```
static synchronized m1()//class level lock  
static synchronized m2(){  
    static m3()//no lock required
```



File Edit View
4.12.2022_multithreading_2dec_2022 - Notepad
methods simultaneously.
5. Class level lock and object lock both are different and there is no relationship between these two.

e.g.:

```
class X{  
    static synchronized m1(){//class level lock  
    static synchronized m2(){  
        static m3(){}//no lock required  
        synchronized m4(){}//object level lock  
    }  
    m5(){}//no lock required
```

- t1=> m1() => class level lock applied and chance is given
- t2=> m2() => enter into waiting state
- t3=> m3() => gets a chance for execution without any lock
- t4=> m4() => object level lock applied and chance is given
- t5=> m5() => gets a chance for execution without any lock

2X



Editor: - (D:\Test.java)

File Edit View Search Document Project Tools Browser Egret Window Help

Directory Object Functions

D:\ New Volume RECYCLE.BIN Computer Properties Enterprise Resource Front and windows GitHub GitKraken MinGW MinGW-W64

TextArea

Java (*.java) Test.java Test.java Test.java

For Help press F1

20°C Party (Study)

Q. Search

1 2 3 4 5 6 7

1 class Display{
2
3 public synchronized void displayNumbers(){
4 for (int i=1;i<=10 ;i++)
5 {
6 System.out.print(i);
7 try{
8 Thread.sleep(2000);
9 }
10 catch (InterruptedException e){
11 }
12 }
13 }
14 public synchronized void displayCharacters(){
15 for (int i=65;i<=75 ;i++)
16 {
17 System.out.print((char)i);
18 try{
19 Thread.sleep(2000);
20 }
21 catch (InterruptedException e){
22 }
23 }
24 }
25 }
26 }
27 }
28 }
29 }
30 }
31 }
32 }
33 }
34 }
35 }
36 }
37 }
38 }
39 }
40 }
41 }
42 }
43 }
44 }
45 }
46 }
47 }
48 }
49 }
50 }
51 }
52 }
53 }
54 }
55 }
56 }
57 }
58 }
59 }
60 }
61 }
62 }
63 }
64 }
65 }
66 }
67 }
68 }
69 }
70 }
71 }
72 }
73 }
74 }
75 }
76 }
77 }
78 }
79 }
80 }
81 }
82 }
83 }
84 }
85 }
86 }
87 }
88 }
89 }
90 }
91 }
92 }
93 }
94 }
95 }
96 }
97 }
98 }
99 }
100 }
101 }
102 }
103 }
104 }
105 }
106 }
107 }
108 }
109 }
110 }
111 }
112 }
113 }
114 }
115 }
116 }
117 }
118 }
119 }
120 }
121 }
122 }
123 }
124 }
125 }
126 }
127 }
128 }
129 }
130 }
131 }
132 }
133 }
134 }
135 }
136 }
137 }
138 }
139 }
140 }
141 }
142 }
143 }
144 }
145 }
146 }
147 }
148 }
149 }
150 }
151 }
152 }
153 }
154 }
155 }
156 }
157 }
158 }
159 }
160 }
161 }
162 }
163 }
164 }
165 }
166 }
167 }
168 }
169 }
170 }
171 }
172 }
173 }
174 }
175 }
176 }
177 }
178 }
179 }
180 }
181 }
182 }
183 }
184 }
185 }
186 }
187 }
188 }
189 }
190 }
191 }
192 }
193 }
194 }
195 }
196 }
197 }
198 }
199 }
200 }
201 }
202 }
203 }
204 }
205 }
206 }
207 }
208 }
209 }
210 }
211 }
212 }
213 }
214 }
215 }
216 }
217 }
218 }
219 }
220 }
221 }
222 }
223 }
224 }
225 }
226 }
227 }
228 }
229 }
230 }
231 }
232 }
233 }
234 }
235 }
236 }
237 }
238 }
239 }
240 }
241 }
242 }
243 }
244 }
245 }
246 }
247 }
248 }
249 }
250 }
251 }
252 }
253 }
254 }
255 }
256 }
257 }
258 }
259 }
260 }
261 }
262 }
263 }
264 }
265 }
266 }
267 }
268 }
269 }
270 }
271 }
272 }
273 }
274 }
275 }
276 }
277 }
278 }
279 }
280 }
281 }
282 }
283 }
284 }
285 }
286 }
287 }
288 }
289 }
290 }
291 }
292 }
293 }
294 }
295 }
296 }
297 }
298 }
299 }
300 }
301 }
302 }
303 }
304 }
305 }
306 }
307 }
308 }
309 }
310 }
311 }
312 }
313 }
314 }
315 }
316 }
317 }
318 }
319 }
320 }
321 }
322 }
323 }
324 }
325 }
326 }
327 }
328 }
329 }
330 }
331 }
332 }
333 }
334 }
335 }
336 }
337 }
338 }
339 }
340 }
341 }
342 }
343 }
344 }
345 }
346 }
347 }
348 }
349 }
350 }
351 }
352 }
353 }
354 }
355 }
356 }
357 }
358 }
359 }
360 }
361 }
362 }
363 }
364 }
365 }
366 }
367 }
368 }
369 }
370 }
371 }
372 }
373 }
374 }
375 }
376 }
377 }
378 }
379 }
380 }
381 }
382 }
383 }
384 }
385 }
386 }
387 }
388 }
389 }
390 }
391 }
392 }
393 }
394 }
395 }
396 }
397 }
398 }
399 }
400 }
401 }
402 }
403 }
404 }
405 }
406 }
407 }
408 }
409 }
410 }
411 }
412 }
413 }
414 }
415 }
416 }
417 }
418 }
419 }
420 }
421 }
422 }
423 }
424 }
425 }
426 }
427 }
428 }
429 }
430 }
431 }
432 }
433 }
434 }
435 }
436 }
437 }
438 }
439 }
440 }
441 }
442 }
443 }
444 }
445 }
446 }
447 }
448 }
449 }
450 }
451 }
452 }
453 }
454 }
455 }
456 }
457 }
458 }
459 }
460 }
461 }
462 }
463 }
464 }
465 }
466 }
467 }
468 }
469 }
470 }
471 }
472 }
473 }
474 }
475 }
476 }
477 }
478 }
479 }
480 }
481 }
482 }
483 }
484 }
485 }
486 }
487 }
488 }
489 }
490 }
491 }
492 }
493 }
494 }
495 }
496 }
497 }
498 }
499 }
500 }
501 }
502 }
503 }
504 }
505 }
506 }
507 }
508 }
509 }
510 }
511 }
512 }
513 }
514 }
515 }
516 }
517 }
518 }
519 }
520 }
521 }
522 }
523 }
524 }
525 }
526 }
527 }
528 }
529 }
530 }
531 }
532 }
533 }
534 }
535 }
536 }
537 }
538 }
539 }
540 }
541 }
542 }
543 }
544 }
545 }
546 }
547 }
548 }
549 }
550 }
551 }
552 }
553 }
554 }
555 }
556 }
557 }
558 }
559 }
560 }
561 }
562 }
563 }
564 }
565 }
566 }
567 }
568 }
569 }
570 }
571 }
572 }
573 }
574 }
575 }
576 }
577 }
578 }
579 }
580 }
581 }
582 }
583 }
584 }
585 }
586 }
587 }
588 }
589 }
590 }
591 }
592 }
593 }
594 }
595 }
596 }
597 }
598 }
599 }
600 }
601 }
602 }
603 }
604 }
605 }
606 }
607 }
608 }
609 }
610 }
611 }
612 }
613 }
614 }
615 }
616 }
617 }
618 }
619 }
620 }
621 }
622 }
623 }
624 }
625 }
626 }
627 }
628 }
629 }
630 }
631 }
632 }
633 }
634 }
635 }
636 }
637 }
638 }
639 }
640 }
641 }
642 }
643 }
644 }
645 }
646 }
647 }
648 }
649 }
650 }
651 }
652 }
653 }
654 }
655 }
656 }
657 }
658 }
659 }
660 }
661 }
662 }
663 }
664 }
665 }
666 }
667 }
668 }
669 }
670 }
671 }
672 }
673 }
674 }
675 }
676 }
677 }
678 }
679 }
680 }
681 }
682 }
683 }
684 }
685 }
686 }
687 }
688 }
689 }
690 }
691 }
692 }
693 }
694 }
695 }
696 }
697 }
698 }
699 }
700 }
701 }
702 }
703 }
704 }
705 }
706 }
707 }
708 }
709 }
710 }
711 }
712 }
713 }
714 }
715 }
716 }
717 }
718 }
719 }
720 }
721 }
722 }
723 }
724 }
725 }
726 }
727 }
728 }
729 }
730 }
731 }
732 }
733 }
734 }
735 }
736 }
737 }
738 }
739 }
740 }
741 }
742 }
743 }
744 }
745 }
746 }
747 }
748 }
749 }
750 }
751 }
752 }
753 }
754 }
755 }
756 }
757 }
758 }
759 }
760 }
761 }
762 }
763 }
764 }
765 }
766 }
767 }
768 }
769 }
770 }
771 }
772 }
773 }
774 }
775 }
776 }
777 }
778 }
779 }
770 }
771 }
772 }
773 }
774 }
775 }
776 }
777 }
778 }
779 }
780 }
781 }
782 }
783 }
784 }
785 }
786 }
787 }
788 }
789 }
790 }
791 }
792 }
793 }
794 }
795 }
796 }
797 }
798 }
799 }
800 }
801 }
802 }
803 }
804 }
805 }
806 }
807 }
808 }
809 }
8010 }
8011 }
8012 }
8013 }
8014 }
8015 }
8016 }
8017 }
8018 }
8019 }
8020 }
8021 }
8022 }
8023 }
8024 }
8025 }
8026 }
8027 }
8028 }
8029 }
8030 }
8031 }
8032 }
8033 }
8034 }
8035 }
8036 }
8037 }
8038 }
8039 }
8040 }
8041 }
8042 }
8043 }
8044 }
8045 }
8046 }
8047 }
8048 }
8049 }
8050 }
8051 }
8052 }
8053 }
8054 }
8055 }
8056 }
8057 }
8058 }
8059 }
8060 }
8061 }
8062 }
8063 }
8064 }
8065 }
8066 }
8067 }
8068 }
8069 }
8070 }
8071 }
8072 }
8073 }
8074 }
8075 }
8076 }
8077 }
8078 }
8079 }
8080 }
8081 }
8082 }
8083 }
8084 }
8085 }
8086 }
8087 }
8088 }
8089 }
8080 }
8081 }
8082 }
8083 }
8084 }
8085 }
8086 }
8087 }
8088 }
8089 }
8090 }
8091 }
8092 }
8093 }
8094 }
8095 }
8096 }
8097 }
8098 }
8099 }
80100 }
80101 }
80102 }
80103 }
80104 }
80105 }
80106 }
80107 }
80108 }
80109 }
80110 }
80111 }
80112 }
80113 }
80114 }
80115 }
80116 }
80117 }
80118 }
80119 }
80120 }
80121 }
80122 }
80123 }
80124 }
80125 }
80126 }
80127 }
80128 }
80129 }
80130 }
80131 }
80132 }
80133 }
80134 }
80135 }
80136 }
80137 }
80138 }
80139 }
80140 }
80141 }
80142 }
80143 }
80144 }
80145 }
80146 }
80147 }
80148 }
80149 }
80150 }
80151 }
80152 }
80153 }
80154 }
80155 }
80156 }
80157 }
80158 }
80159 }
80160 }
80161 }
80162 }
80163 }
80164 }
80165 }
80166 }
80167 }
80168 }
80169 }
80170 }
80171 }
80172 }
80173 }
80174 }
80175 }
80176 }
80177 }
80178 }
80179 }
80180 }
80181 }
80182 }
80183 }
80184 }
80185 }
80186 }
80187 }
80188 }
80189 }
80190 }
80191 }
80192 }
80193 }
80194 }
80195 }
80196 }
80197 }
80198 }
80199 }
80100 }
80101 }
80102 }
80103 }
80104 }
80105 }
80106 }
80107 }
80108 }
80109 }
80110 }
80111 }
80112 }
80113 }
80114 }
80115 }
80116 }
80117 }
80118 }
80119 }
80120 }
80121 }
80122 }
80123 }
80124 }
80125 }
80126 }
80127 }
80128 }
80129 }
80130 }
80131 }
80132 }
80133 }
80134 }
80135 }
80136 }
80137 }
80138 }
80139 }
80140 }
80141 }
80142 }
80143 }
80144 }
80145 }
80146 }
80147 }
80148 }
80149 }
80150 }
80151 }
80152 }
80153 }
80154 }
80155 }
80156 }
80157 }
80158 }
80159 }
80160 }
80161 }
80162 }
80163 }
80164 }
80165 }
80166 }
80167 }
80168 }
80169 }
80170 }
80171 }
80172 }
80173 }
80174 }
80175 }
80176 }
80177 }
80178 }
80179 }
80180 }
80181 }
80182 }
80183 }
80184 }
80185 }
80186 }
80187 }
80188 }
80189 }
80190 }
80191 }
80192 }
80193 }
80194 }
80195 }
80196 }
80197 }
80198 }
80199 }
80100 }
80101 }
80102 }
80103 }
80104 }
80105 }
80106 }
80107 }
80108 }
80109 }
80110 }
80111 }
80112 }
80113 }
80114 }
80115 }
80116 }
80117 }
80118 }
80119 }
80120 }
80121 }
80122 }
80123 }
80124 }
80125 }
80126 }
80127 }
80128 }
80129 }
80130 }
80131 }
80132 }
80133 }
80134 }
80135 }
80136 }
80137 }
80138 }
80139 }
80140 }
80141 }
80142 }
80143 }
80144 }
80145 }
80146 }
80147 }
80148 }
80149 }
80150 }
80151 }
80152 }
80153 }
80154 }
80155 }
80156 }
80157 }
80158 }
80159 }
80160 }
80161 }
80162 }
80163 }
80164 }
80165 }
80166 }
80167 }
80168 }
80169 }
80170 }
80171 }
80172 }
80173 }
80174 }
80175 }
80176 }
80177 }
80178 }
80179 }
80180 }
80181 }
80182 }
80183 }
80184 }
80185 }
80186 }
80187 }
80188 }
80189 }
80190 }
80191 }
80192 }
80193 }
80194 }
80195 }
80196 }
80197 }
80198 }
80199 }
80100 }
80101 }
80102 }
80103 }
80104 }
80105 }
80106 }
80107 }
80108 }
80109 }
80110 }
80111 }
80112 }
80113 }
80114 }
80115 }
80116 }
80117 }
80118 }
80119 }
80120 }
80121 }
80122 }
80123 }
80124 }
80125 }
80126 }
80127 }
80128 }
80129 }
80130 }
80131 }
80132 }
80133 }
80134 }
80135 }
80136 }
80137 }
80138 }
80139 }
80140 }
80141 }
80142 }
80143 }
80144 }
80145 }
80146 }
80147 }
80148 }
80149 }
80150 }
80151 }
80152 }
80153 }
80154 }
80155 }
80156 }
80157 }
80158 }
80159 }
80160 }
80161 }
80162 }
80163 }
80164 }
80165 }
80166 }
80167 }
80168 }
80169 }
80170 }
80171 }
80172 }
80173 }
80174 }
80175 }
80176 }
80177 }
80178 }
80179 }
80180 }
80181 }
80182 }
80183 }
80184 }
80185 }
80186 }
80187 }
80188 }
80189 }
80190 }
80191 }
80192 }
80193 }
80194 }
80195 }
80196 }
80197 }
80198 }
80199 }
80100 }
80101 }
80102 }
80103 }
80104 }
80105 }
80106 }
80107 }
80108 }
80109 }
80110 }
80111 }
80112 }
80113 }
80114 }
80115 }
80116 }
80117 }
80118 }
80119 }
80120 }
80121 }
80122 }
80123 }
80124 }
80125 }
80126 }
80127 }
80128 }
80129 }
80130 }
80131 }
80132 }
80133 }
80134 }
80135 }
80136 }
80137 }
80138 }
80139 }
80140 }
80141 }
80142 }
80143 }
80144 }
80145 }
80146 }
80147 }
80148 }
80149 }
80150 }
80151 }
80152 }
80153 }
80154 }
80155 }
80156 }
80157 }
80158 }
80159 }
80160 }
80161 }
80162 }
80163 }
80164 }
80165 }
80166 }
80167 }
80168 }
80169 }
80170 }
80171 }
80172 }
80173 }
80174 }
80175 }
80176 }
80177 }
80178 }
80179 }
80180 }
80181 }
80182 }
80183 }
80184 }
80185 }
80186 }
80187 }
80188 }
80189 }
80190 }
80191 }
80192 }
80193 }
80194 }
80195 }
80196 }
80197 }
80198 }
80199 }
80100 }
80101 }
80102 }
80103 }
80104 }
80105 }
80106 }
80107 }
80108 }
80109 }
80110 }
80111 }
80112 }
80113 }
80114 }
80115 }
80116 }
80117 }
80118 }
80119 }
80120 }
80121 }
80122 }
80123 }
80124 }

Editor - (D:\Test.java)

```
File Edit View Search Document Project Tools Browser Egret Window Help  
Directory Object Functions  
(D) New Volume 25 } 1 2 3 4 5 6 7  
D:\ RECYCLE.BIN  
CoronaEngine  
Enterprise  
EnterpriseBundles  
FieldPrograms  
Font and icons  
gitHub  
gitment  
gitnitive  
IntelliJ  
Java (Java)  
@ Test.java @ Test.java @ Test.java  
For Help press F1  
20C  
Fully Study
```

Nitin Manjrekar

Java (Java)

int i = 1; col 14 67 78 PC ANSI EVALUATION VERSION: 21.52 EN IN 05.12.2022

```
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25 }  
26 }  
27  
28 class MyThread1 extends Thread{  
29     Display d;  
30     MyThread1(Display d){  
31         this.d=d;  
32     }  
33     @Override  
34     public void run(){  
35         d.displayNumbers();  
36     }  
37 }  
38 }  
39  
40 class MyThread2 extends Thread{  
41     Display d;  
42     MyThread2(Display d){  
43         this.d=d;  
44     }  
45     @Override  
46     public void run(){  
47         System.out.println("MyThread2");  
48     }  
49 }
```

b
~

A screenshot of a Java IDE showing two classes: MyThread1 and MyThread2. The MyThread1 class extends Thread and overrides the run() method to call d.displayNumbers(). The MyThread2 class also extends Thread and overrides the run() method to call d.displayCharacters(). Both classes have a field named d of type Display. The IDE interface includes a toolbar, menu bar, and status bar at the bottom. A large, stylized green hand-drawn mark is overlaid on the bottom left of the code editor area.

Erkhan - [D:\Net.java]

File Edit View Search Document Project Icons Browser Engine Window Help

Directory Context Functions

(D:\) New Volume

D:\ RECYCLE.BIN

CoronaPractice

EmpJava

EnterpriseSearch

FieldPrograms

Front end vision

gitHub

gitment

gitment

IntelliJ

1 2 3 4 5 6 7

1. 49 =
2. 50
3. 51
4. 52 }
5. 53
6. 54 class Test
7. 55 = {
8. 56 public static void main(String[] args)
9. 57 {
10. 58 Display d1=new Display();
11. 59
12. 60 MyThread1 t1= new MyThread1(d1);
13. 61 MyThread2 t2= new MyThread2(d1);
14. 62
15. 63
16. 64 t1.start();
17. 65 t2.start();
18. 66
19. 67
20. 68
21. 69 }
22. 70

Java (*.java)

For Help press F1.

20°C Fully cloudy

Q. Search

In 54 col 1 70 63 PC ANSI EVALUATION VERSION

ENGLISH IN 05.12.2022 21:50



```
16.12.2022_multithreading_class_pptes - Notepad  
File Edit View  
displayNumbers()  
  
Synchronized block  
=====  
void m1(){  
    ....  
    ....  
    ....  
    ....  
    ....  
}  
  
synchronized(lock){  
    ....  
    ....  
    ....  
    ....  
    ....  
}  
  
29
```

6.12.2022_multithreading_2nd_Questions - Notepad

File Edit View

...

...

...

...

...

...

...

...

}

if few lines of code is required to get synchronized then it is not recommended to make method only as synchronized.
If we do this then for threads performance will be low, to resolve this problem we use "synchronized block",
due to synchronized block performance will be improved.

↑

}

if few lines of code is required to get synchronized then it is not recommended to make method only as synchronized.

If we do this then for threads performance will be low, to resolve this problem we use "synchronized block", due to synchronized block performance will be improved.

```
synchronized(this){  
}  
synchronized(Display.class){  
}  
synchronized(d){  
}
```



File
Edit
View

=====

If a thread got a lock of current object, then it is allowed to execute that block

a.

```
synchronized(this){  
    ....  
    ....  
    ....  
}
```

To get a lock of particular object:: B

b.

```
synchronized(B){  
    ....  
    ....  
    ....  
}
```

If a thread got a lock of particular object B, then it is allowed to execute that block.

c. To get class level lock we have to declare synchronized block as follow

```
    ....  
    ....  
    ....  
}
```

10% Windows (CEP)
UTF-8

EN IN 06.12.2022

20°C, 60%
Fully cloudy

Q. Search

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

<

Editor - Test.java

File Edit View Search Document Project Tools Browser Editor Window Help

Directory Object Functions

(D) New Volume

D:\ RECYCLE.BIN

CodeInsight

Enterprise

FilePrograms

Frontend

gitHub

gitment

gitment

IntelliJ

Java (*.java)

Test.java

For Help press F1

20°C
Fully cloudy

Q. Search

1 2 3 4 5 6 7

1 d.displayCharacters();

2 }

3 }

4 }

5 class Test

6 {

7 public static void main(String[] args)

8 {

9 Integer x= 48;

10 synchronized(x){

11 System.out.println(x);

12 }

13 }

14 }

15 }

16 }

17 }

18 }

19 }

20 }

21 }

22 }

23 }

24 }

25 }

26 }

27 }

28 }

29 }

30 }

31 }

32 }

33 }

34 }

35 }

36 }

37 }

38 }

39 }

40 }

41 }

42 }

43 }

44 }

45 }

46 }

47 }

48 }

49 }

50 }

51 }

52 {

53 }

54 }

55 }

56 }

57 }

58 }

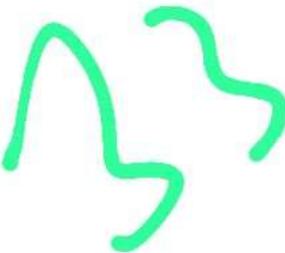
59 }

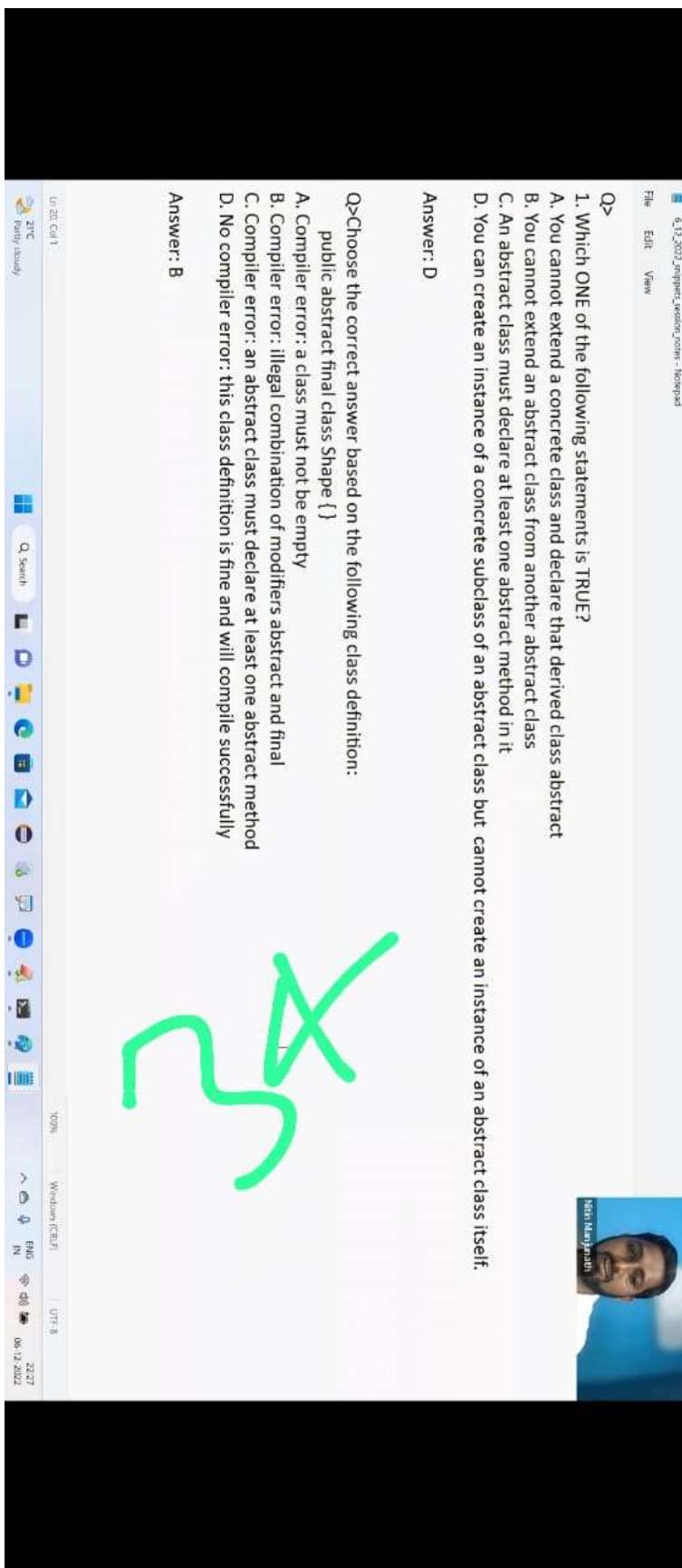
60 }

61 }

62 }

In 55 col 16 62 20 PC ANSI EVALUATION VERSION: 22.06 EN IN 05.12.2022





File Edit View
46.12.2022 - snippets-session-classes - Notepad

```
Q>Choose the best option based on this program:  
class Shape{  
    public Shape(){  
        System.out.println("Shape constructor");  
    }  
    public class Color{  
        public Color(){  
            System.out.println("Color constructor");  
        }  
    }  
}  
class TestColor{  
    public static void main(String []args){  
        Shape.Color black = new Shape().Color(); // #1  
    }  
}
```



- A. Compiler error: the method Color() is undefined for the type Shape
B. Compiler error: invalid inner class
C. Works fine: Shape constructor, Color constructor
D. Works fine: Color constructor, Shape constructor

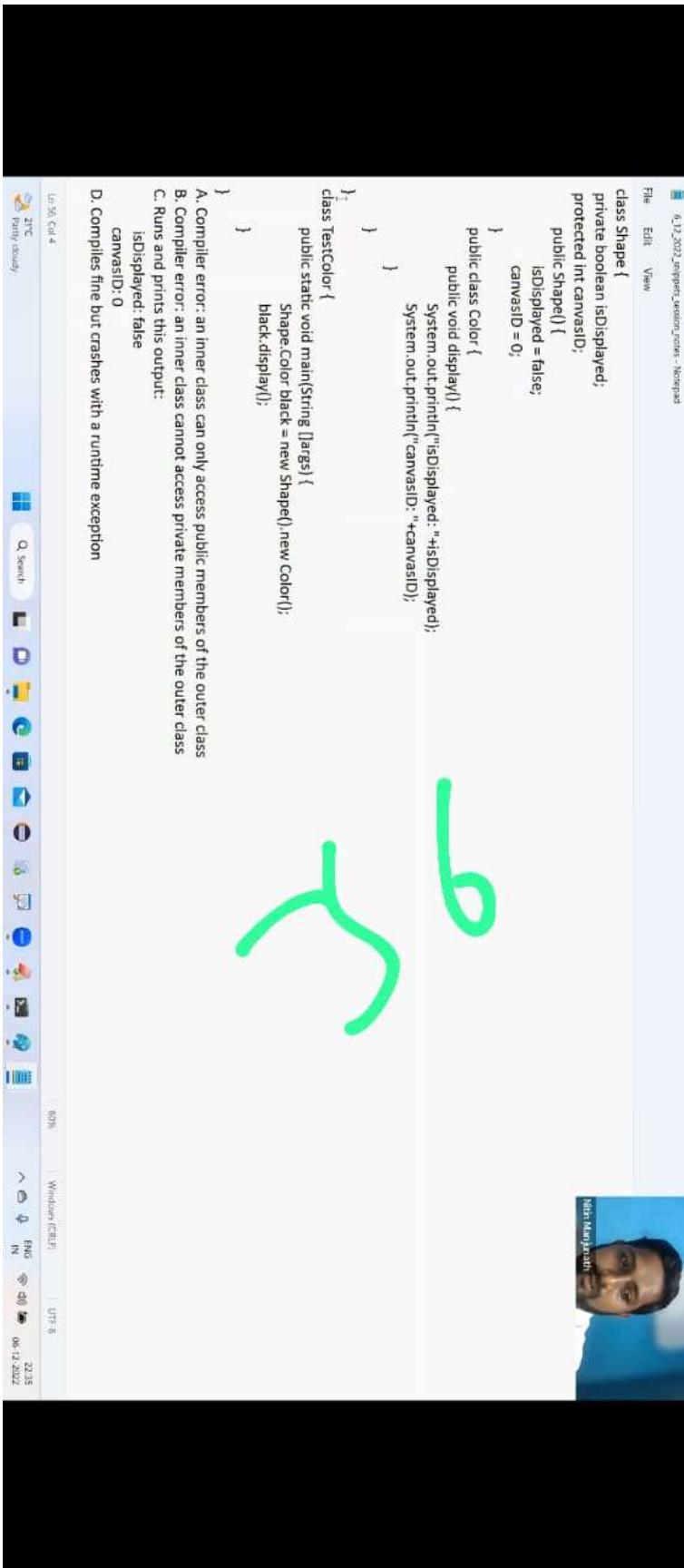
Answer: A

Un 41 Col 10
20°C Fully cloudy



T0M Windows (C:\E\T) UTF-8

ENGLISH IN 06.12.2022



A screenshot of a Java code editor showing a class hierarchy. The code is as follows:

```
6.12.2022.javafiles_exercise_main - Notepad
File Edit View
class Shape{
    private boolean isDisplayed;
    protected int canvasID;
    public Shape() {
        isDisplayed = false;
        canvasID = 0;
    }
    public class Color{
        public void display(){
            System.out.println("isDisplayed: "+isDisplayed);
            System.out.println("canvasID: "+canvasID);
        }
    }
}
class TestColor{
    public static void main(String [args){
        Shape.Color black = new Shape().new Color();
        black.display();
    }
}
```

Below the code, there is a list of four options:

- A. Compiler error: an inner class can only access public members of the outer class
- B. Compiler error: an inner class cannot access private members of the outer class
- C. Runs and prints this output:
isDisplayed: false
canvasID: 0
- D. Compiles fine but crashes with a runtime exception

Handwritten notes in green ink are present in the image: a large question mark is written near the bottom left, and a small checkmark is written near the top right.



```
6.12.2022 Anupama_awalne - Notepad  
File Edit View  
}  
public class Color {  
    public void display() {  
        System.out.println("isDisplayed: "+isDisplayed);  
        System.out.println("canvasID: "+canvasID);  
    }  
}  
  
class TestColor {  
    public static void main(String []args) {  
        Shape.color black = new Shape().new Color();  
        black.display();  
    }  
}  
A. Compiler error: an inner class can only access public members of the outer class  
B. Compiler error: an inner class cannot access private members of the outer class  
C. Runs and prints this output:  
    isDisplayed: false  
    canvasID: 0  
D. Compiles fine but crashes with a runtime exception  
Answer: C
```

20°C
Partly cloudy



11:53, 01/12/2022 60% Windows (CEP) UTC-8
EN IN 22:37 05.12.2022

4.12.2022 Anupam_Agarwal - Notepad

File Edit View

Determine the behavior of this program:

```
interface DoNothing {
```

```
}
```

```
default void doNothing() { System.out.println("doNothing"); } //from jdk1.8 inside an interface we can have concrete methods also.
```

```
@FunctionalInterface
```

```
interface DontDoAnything extends DoNothing {
```

```
@Override
```

```
abstract void doNothing();
```

```
}
```

```
class LambdaTest {
```

```
    public static void main(String []args) {
```

```
        DontDoAnything beIdle = () -> System.out.println("be idle");
```

```
        beIdle.doNothing();
```

```
}
```

A. This program results in a compiler error for DontDoAnything interface: cannot override default method to be an abstract method

B. This program results in a compiler error: DontDoAnything is not a functional interface

C. This program prints: doNothing

D. This program prints: be idle

Answer: D

88



11:56 AM 1

21°C

Fatty (skimmy)

Q Search

U

Y

E

W

S

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

W

6.12.2022.Answers_exercise_name - Notepad

File Edit View

Answer: D

Q> Determine the behavior of this program:

```
interface BaseInterface {
    default void foo() { System.out.println("BaseInterface's foo");}
}
interface DerivedInterface extends BaseInterface {
    default void foo() { System.out.println("DerivedInterface's foo");}
}
interface AnotherInterface {
    public static void foo() { System.out.println("AnotherInterface's foo");}
}
public class MultipleInheritance implements DerivedInterface, AnotherInterface {
    public static void main(String []args) {
        new MultipleInheritance().foo();
    }
}
```

A. This program will result in a compiler error: Redundant method definition for function foo
B. This program will result in a compiler error in MultipleInheritance class: Ambiguous call to function foo
C. The program prints: DerivedInterface's foo
D. The program prints: AnotherInterface's foo

Answer: D

12:17:51 6/12/2022

20°C Fairly cloudy

Windows (CEP) 80% 6/12/2022 22:49

ENGLISH (US) IN 06:12:2022

File Edit View

File Edit View

Determine the behavior of this program:

```
class LambdaFunctionTest
    @FunctionalInterface
    interface LambdaFunction<T>
        int apply(int i);
```

```
    }
    public static void main(String []args) {
        boolean equals(java.lang.Object arg0);
        int appWin();
    }
}
```

```
    }  
  
    LambdaFunction lambdaFunction = i -> i * i; // #  
    System.out.println(lambdaFunction.apply(10));
```

- A. This program results in a compiler error: interfaces cannot be defined inside classes
- B. This program results in a compiler error: @FunctionalInterface used for LambdaFunction that defines two abstract methods
- C. This program results in a compiler error in code marked with #1: syntax error
- D. This program compiles without errors, and when run, it prints 100 in console

Answer: D

