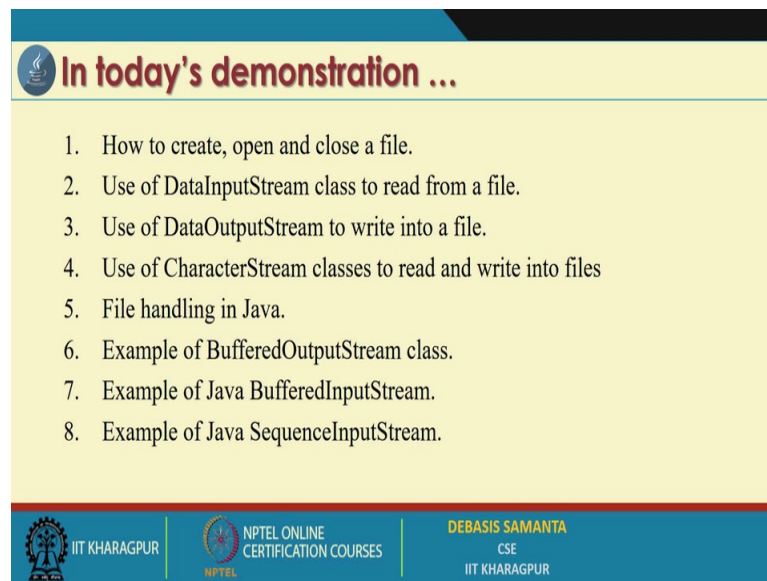


Programming in Java
Prof. Debasis Samanta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 33
Demonstration – XII

So, this demo is based on the topics that you have covered in the last three modules. In the last three modules we have cover on the topic Input-Output streams in java. And as you know the input and output is the very important activity in any program development. And in order to make this diversified input-output process that means input from the different sources, output to different sources, there are many mechanism rather many ways has been devised in java system.

(Refer Slide Time: 00:53)



In today's demonstration ...

1. How to create, open and close a file.
2. Use of DataInputStream class to read from a file.
3. Use of DataOutputStream to write into a file.
4. Use of CharacterStream classes to read and write into files
5. File handling in Java.
6. Example of BufferedOutputStream class.
7. Example of Java BufferedInputStream.
8. Example of Java SequenceInputStream.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | DEBASIS SAMANTA CSE IIT KHARAGPUR

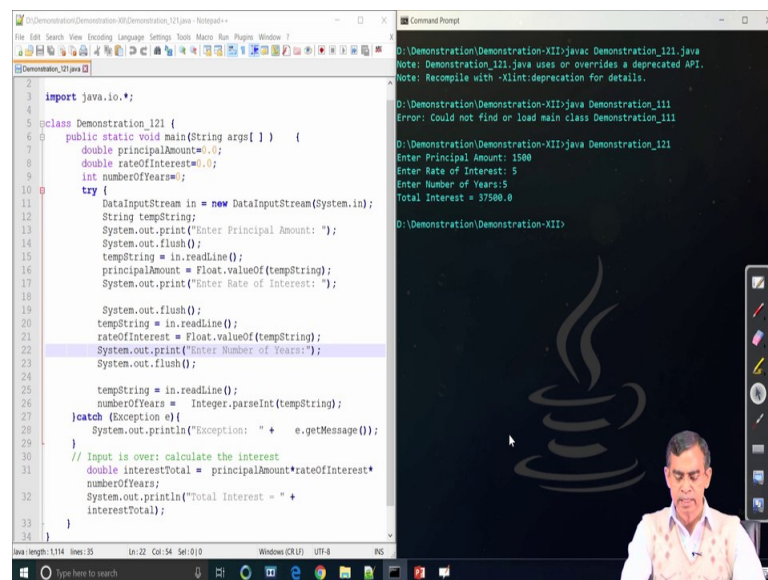
Now, today we will discussed about how the different in way of input-output is possible. There are some trivial input output mechanism, those are related to the standard input, and then standard output namely keyboard, and then display unit we have already covered, we usually used it in our previous programs. But, here we will discussed about other than the standard input and output, how we can store some data into memory or we can receive something which is stored in hard disk like.

So, our demonstration includes how to create, how to open, and close a file, how we can create our own file. And there are again two different classes rather we can say the ways

the byte stream classes, and then character stream classes to read and write into file from file by means of `DataInputStream` and the data output stream. So, we will discussed about the usage of these two classes to create files to access the data, all these things.

And then we will discussed again file handling in java. So, file is basically secondary storage right, we can store some data in a permanent non-volatile way into the secondary storage space. So, how we can create a file such a secondary storage, and also how we can open, how we can copy, how we can merge, so many other things are there. And few advanced methods, they are called `BufferedInputStream`, and `SequenceInput Stream` to make our programming easy we will discuss in this demo.

(Refer Slide Time: 02:51)



```
import java.io.*;

class Demonstration_121 {
    public static void main(String args[] ) {
        double principalAmount=0;
        double rateOfInterest=0;
        int numberOfYears=0;
        try {
            DataInputStream in = new DataInputStream(System.in);
            String tempString;
            System.out.print("Enter Principal Amount: ");
            System.out.flush();
            tempString = in.readLine();
            principalAmount = Float.valueOf(tempString);
            System.out.print("Enter Rate of Interest: ");
            System.out.flush();

            tempString = in.readLine();
            rateOfInterest = Float.valueOf(tempString);
            System.out.print("Enter Number of Years:");
            System.out.flush();

            tempString = in.readLine();
            numberOfYears = Integer.parseInt(tempString);
        } catch (Exception e) {
            System.out.println("Exception: " + e.getMessage());
        }
        // Input is over: calculate the interest
        double interestTotal = principalAmount*rateOfInterest*
        numberOfYears;
        System.out.println("Total Interest = " +
        interestTotal);
    }
}
```

```
D:\Demonstration\Demonstration-XII>javac Demonstration_121.java
Note: Demonstration_121.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.

D:\Demonstration\Demonstration-XII>java Demonstration_121
Enter Principal Amount: 1500
Enter Rate of Interest: 5
Enter Number of Years:5
Total Interest = 37500.0

D:\Demonstration\Demonstration-XII>
```

So, let us have the demo for first thing, and this demo we can start about from very simple idea about using the `DataInputStream` class. We have already familiar to this class in our last in our earlier discussion we have used it. So, `DataInputStream` class as in the name in implies that it will basically for the input purpose, then we using this class, we can read something from some sources.

Now, here we will see exactly `DataInputStream` class can read from memory, can be from network channel, it can read from standard input device. This example illustrate how the `DataInputStream` class can be configured, so that we can read some data from the standard input namely the keyboard. As we see the program here, so this is the main method as we see here. In this main method, we create two way fields namely the

principal amount, and the rate of interest for which we want to read the value from the keyboard, and the number of years also. So, the three input needs to be collected from the user through keyboard.

Now, here basically we create an object called in, and this is object that object is created. And then this object wants that it is created by means of a constructor, where the input is basically System dot in. Now, System dot in implies that it is a standard input the System dot in this is already defined in the java dot lang, so from there it will get the definitions. So, it will basically is a standard input.

Now, so basically what we do we here that we create and DataInputStream object, which basically connect your program to the keyboard. Now, let us see we given from to the user that entire principal amount, here basically we have discussed about that for seeing that means, keyboard has its own buffer. So, we have to clean the buffer, each time we are going to read from it.

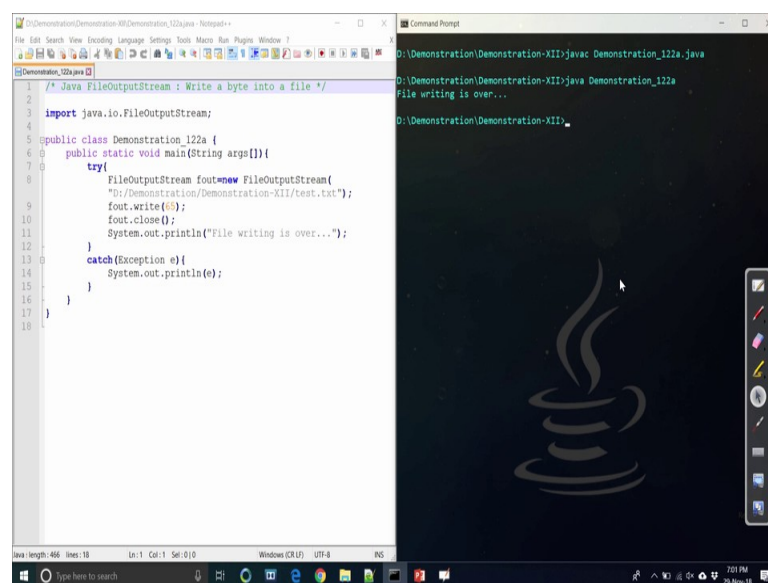
So, System dot out dot flush is basically clean it is now here then ok, so it will read from the object in here that means which connect the keyboard. And the in that read line for this method read line means, it will read the entire content that is there is a buffer. So, the entire whatever the value we will enter, it is read it here, but as you know the value that will be there is a buffer java program read it as a stream. So, we temporarily store this in the stream format here, so it will basically read as a stream. Although these are may have entered some value say 5, 6, 5.25, so it is basically plotting point or double value.

Now, here we convert this string value into our double value or float value. So, using the float method for this in the class float, which is defined in the java.net (Refer Time: 06:02) package, and for this class there is a method called value of this is a static class actually. So, we just convert this string into the float value.

Now, again we read the another form enter interest again we flush it, and then readLine, again read the buffer, and then again convert this string into the float value which stored in the rateofInterest. Say again then we need the integer number from the read, we convert it into integer format as initially it is a string, and then store the value is a number of years here. So, this is the way we can read the three inputs from the standard in standard input that is the keyboard here.

And then finally, we calculate these are trivial process. Now, main thing here we can see you can tell here is that how using this DataInputStream class, we can create a stream object which connect your program to your keyboard. So, the stream can be propagated from the input source to the program. Anyway, so this is the simple program we are already familiar to, and we have discussed it many times earlier. And other than this class also there are many way the input can be found the standard device like common line input, and then the using the Scanner class which is defined in java.util package, we can run all this things.

(Refer Slide Time: 07:43)



The screenshot shows a Java IDE window on the left and a Command Prompt window on the right. The IDE window displays the source code for a Java class named `Demonstration_122a`. The code imports `java.io.FileOutputStream` and defines a `main` method that creates a `FileOutputStream` object, writes the character 'a' to a file named `test.txt`, and prints "File writing is over...". The Command Prompt window shows the command `javac Demonstration_122a.java` being executed, followed by the output `File writing is over...`.

```
1  /* Java FileOutputStream : Write a byte into a file */
2
3  import java.io.FileOutputStream;
4
5  public class Demonstration_122a {
6      public static void main(String args[]){
7          try{
8              FileOutputStream fout=new FileOutputStream(
9                  "D:/Demonstration/Demonstration-XII/test.txt");
10             fout.write('a');
11             fout.close();
12             System.out.println("File writing is over...");
13         }
14         catch(Exception e){
15             System.out.println(e);
16         }
17     }
18 }
```

```
D:\Demonstration\Demonstration-XII>javac Demonstration_122a.java
D:\Demonstration\Demonstration-XII>java Demonstration_122a
File writing is over...
D:\Demonstration\Demonstration-XII>
```

Now, our next example that we are going to discuss about is basically as we have discussed about all the DataInputStream class is basically the two fold using the byte stream, and then character stream. Our next example this examples to illustrate how we can output to a file which is stored in a secondary storage, and we can propagate the output from the program to the file in the byte form. As you know byte is the smallest unit a chunk that the java program can handle it.

Now, here is a program as you see here ok. First you have to import `java.io.FileOutputStream`, this is because we want to propagate the output to a file. And this a `FileOutput Stream` class in defined in the `java.io` package, so the import is must. Now, here is the main method as we see we create an object name of the object is a `fout` is basically of type `FileOutputStream`. And here you see the constructor the `FileOutputStream` has its

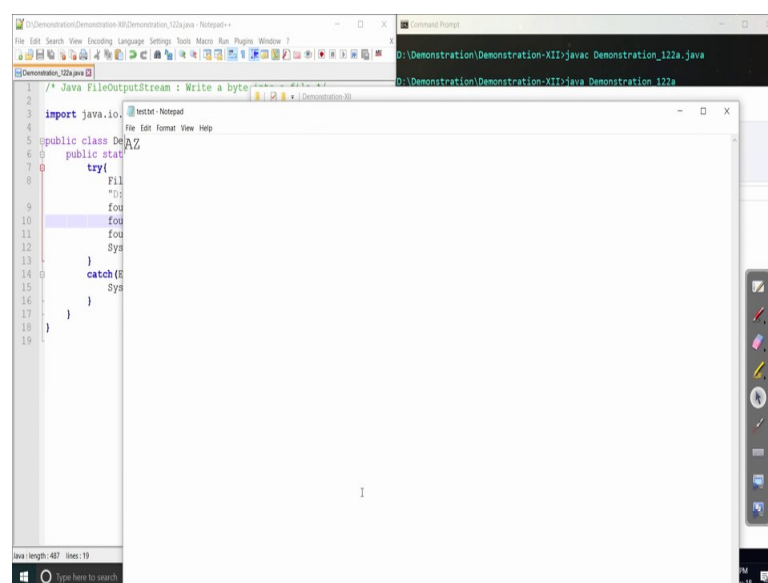
own constructor, where an argument is required is basically the target. So, here actually the target we have mentioned explicitly, where we want to store this content.

So, as you see it is in the D drive under these D drive there is a directive demonstration, under this demonstration XII, and then this is basically the target file and test dot txt. So, you should explicitly mention path of the file, where you want to store your data. So, this is a way that we can do it. Now, here you see for this object fout, fout therefore object is basically the connection from your program to this file test dot txt.

Now, from the programme we can write 65. Now, 65 is basically here in integer form, but it will be converted to and byte form as you know the 65 byte in the byte form, it is basically capital A character. So, ultimately although we give the 65 your program the system will convert into it a byte that means, 65 in the byte is basically the ASCII code is a. So, a will be store actually in the file. And then finally, once the file is open, we have to close the file it is always, you should close your file always wants your task is over, and then finally it print the writing is over.

Now, this program as you see this program will write 65 in the byte code form into a file test.txt. So, after the successful execution of this program, we will see the output file here test.txt that 65 is told there or not. So, this program is run successfully, and then output file is here. The test dot txt is now displayed as we see it basically store A.

(Refer Slide Time: 10:41)



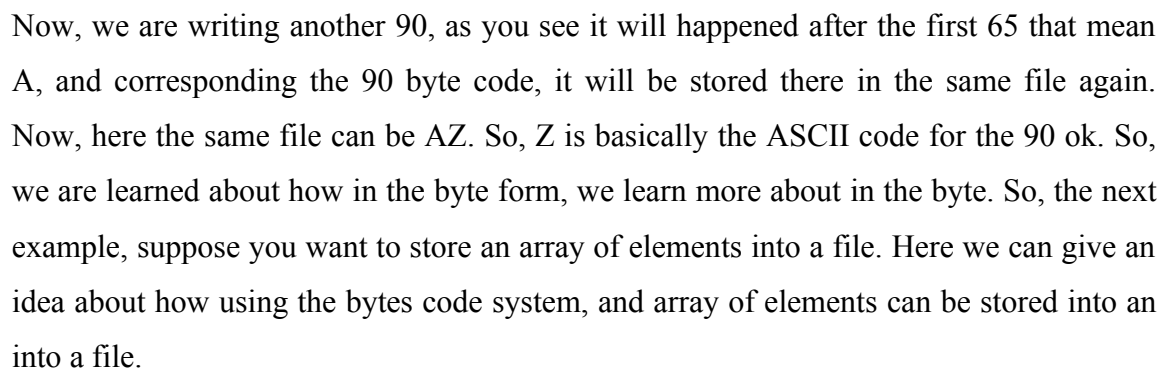
The screenshot displays a Java IDE with a code editor and a terminal window. The code editor shows a Java program that writes the character 'A' to a file named 'test.txt'. The terminal window shows the command to run the program and the output 'Writing is over'.

```
1  /* Java FileOutputStream : Write a byte */
2
3  import java.io.*;
4
5  public class Demo {
6      public static void main(String[] args) {
7          try {
8              File f = new File("D:\\Demonstration\\XII\\test.txt");
9              FileOutputStream fout = new FileOutputStream(f);
10             fout.write("A");
11             fout.close();
12             System.out.println("Writing is over");
13         } catch (Exception e) {
14             e.printStackTrace();
15         }
16     }
17 }
18
19
```

Terminal Output:

```
D:\Demonstration\XII>javac Demonstration_122a.java
D:\Demonstration\XII>java Demonstration_122a
Writing is over
```

(Refer Slide Time: 10:59)

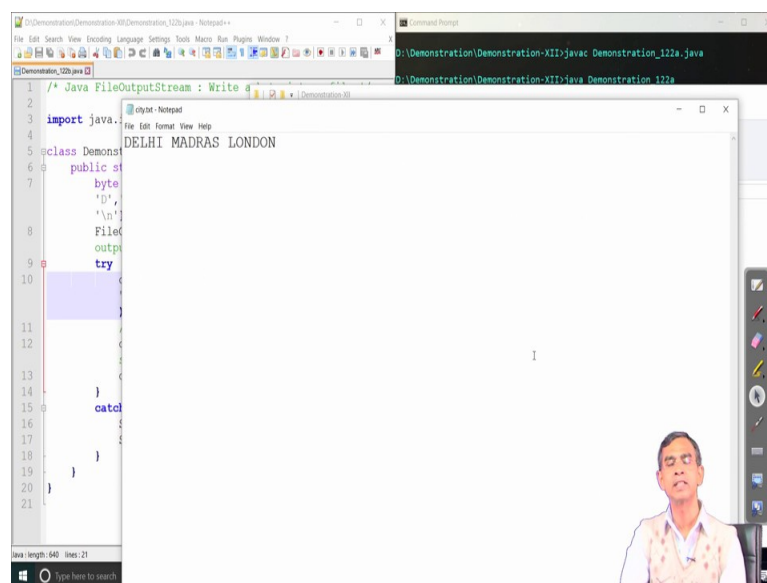


Anyway, so this basically now we create one output objects, so output file is a type of FileOutputStream very similar to the previous one. And then we decide it is a create the object, and finally we make a connection that where this value will store. So, for this object output file, we establish a connection to this is the location. So, the location is as

we see location is city dot txt that means, this is the output destination target, where the output the entire things whichever here will be there.

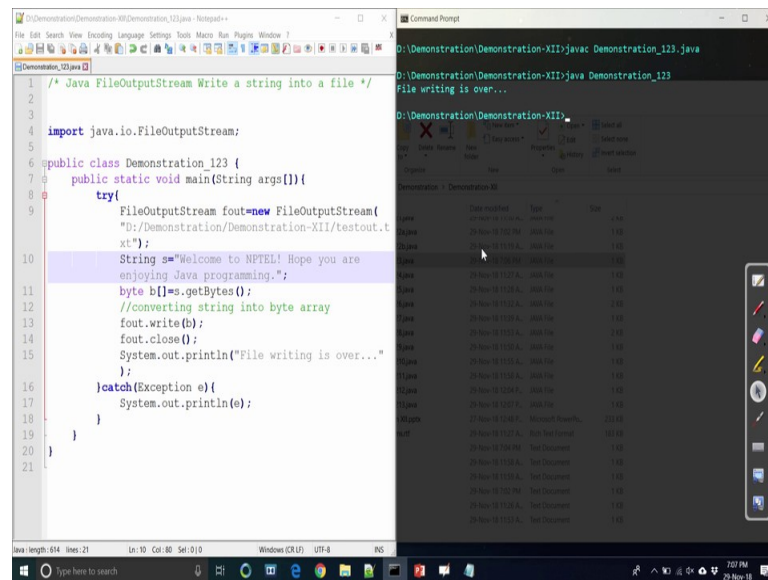
And then finally, you write the entire array here, so passing the array name here and finally close, these are the same as earlier now ok. So, this way the entire array elements will be stored into the file. Now, let us run the program, and final you will be able to see the file in this case city dot txt, which store all the values that is there in the byte array ok. So, file has been written successfully.

(Refer Slide Time: 13:43)

The screenshot shows a Java IDE with a code editor and a terminal. The code editor displays a Java program that writes the contents of a byte array to a file named 'city.txt'. The byte array contains the strings 'DELHI', 'MADRAS', and 'LONDON'. The terminal shows the command to compile and run the program: 'javac Demonstration_122a.java' and 'java Demonstration_122a'. The status bar at the bottom indicates the byte array has a length of 940 and contains 21 elements.

```
1  /* Java FileOutputStream : Write d
2
3  import java.
4
5  class Demonst
6
7  public st
8
9  byte
10 "D",
11 "\n",
12 File
13 outpu
14 try
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
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
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
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
261
```


(Refer Slide Time: 14:07)



The screenshot shows an IDE with two windows. The left window displays a Java file named `Demonstration_123.java` with the following code:

```
1  /* Java FileOutputStream Write a string into a file */
2
3
4  import java.io.FileOutputStream;
5
6  public class Demonstration_123 {
7      public static void main(String args[]){
8          try{
9              FileOutputStream fout=new FileOutputStream(
10                 "D:/Demonstration/Demonstration-XII/testout.txt");
11              String s="Welcome to NPTEL! Hope you are
12                 enjoying java programming.";
13              byte b[]=s.getBytes();
14              //converting string into byte array
15              fout.write(b);
16              fout.close();
17              System.out.println("File writing is over...")
18          }
19          catch(Exception e){
20              System.out.println(e);
21          }
22      }
23  }
```

The right window shows the command prompt output for the command `D:\Demonstration>javac Demonstration_123.java`. The output is:

```
D:\Demonstration>javac Demonstration_123.java
File writing is over...
```

So, an array of elements can be stored, a single element can be stored, and like this, here we are storing all these things in our memory. Now, our next example again using `FileOutputStream` classes that we have discussed that we have to pay to copy some text using again byte array stream into the same file.

Now, here is a program as we see, it is the same `FileOutputStream` class. Here `FileOutputStream`, create the object, and we just connect it to one target `testout.txt`. And here we create a stream here welcome to NPTEL. Here is a small string we have considered, many other large string also we can include say, other things also you can input here.

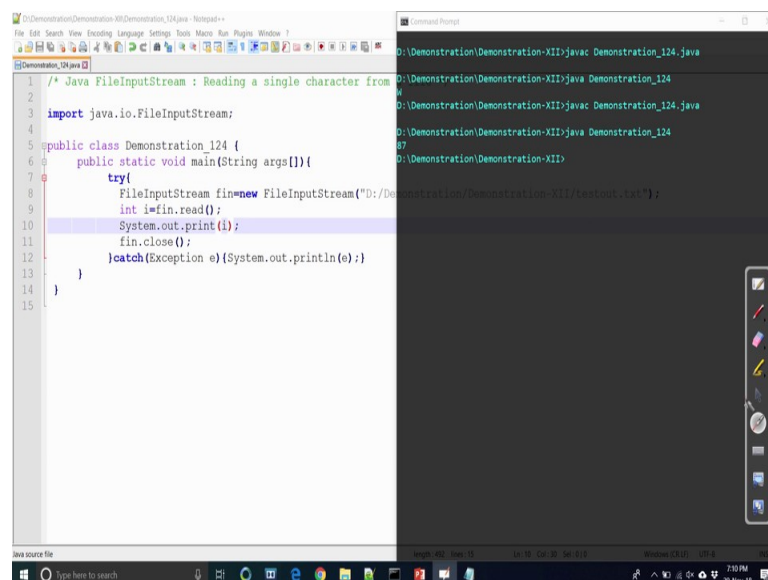
For example, type here welcome to NPTEL, hope you are enjoying java programming hope you are enjoying java programming ok. So, this is the text whatever the text may be here only few characters, very big I mean very large file also can be consider. I will discuss about how a file can be entire file can be copied here, anyway our objecting is that these stray string, which is basically here in the program, I want to store into a file the name of the file is `testout dot txt`.

Now, we first store this string into the form of a bytes. So, here is a mechanism for a string object, how we can convert into the bytes. So, basically string this is the string and we converting into the entire string content is the byte. So, `getBytes` is a method for the string class, which is there in `java dot` (Refer Time: 15:57) package. So, we temporary

stored into an array byte array, it is the similar to the city array we have as discuss there. Then write function write method for this a fout FileOutputStream, which basically write the entire array, it is also same as the earlier one and close. And finally, file writing is over (Refer Time: 16:16), and then it will there.

Now, we see after the successful running of this program. We will see one file testout dot txt has been created, where we could store the stream which has mentioned there fine. Now we are displaying the file content as we see this is the content has been pushed to the file, which we have displaced there. Now, so this is the way that now so far what we have done is that we can create something we can store something from our program to file.

(Refer Slide Time: 16:57)



The screenshot shows a code editor on the left and a command prompt on the right. The code editor displays the following Java code:

```
1  /* Java FileInputStream : Reading a single character from
2
3  import java.io.FileInputStream;
4
5  public class Demonstration_124 {
6      public static void main(String args[]){
7          try{
8              FileInputStream fin=new FileInputStream("D:/Demonstration/Demonstration-XII/testout.txt");
9              int i=fin.read();
10             System.out.print(i);
11             fin.close();
12         }catch(Exception e){System.out.println(e);}
13     }
14 }
15
```

The command prompt shows the execution of the Java program:

```
D:\Demonstration\Demonstration-XII>javac Demonstration_124.java
D:\Demonstration\Demonstration-XII>java Demonstration_124
D:\Demonstration\Demonstration-XII>javac Demonstration_124.java
D:\Demonstration\Demonstration-XII>java Demonstration_124
87
D:\Demonstration\Demonstration-XII>
```

Now, we learn about how to read a content from a file. Now, this very simple program for this purpose for reading some content from the file, we have to have that create object for the file input stream class. So, file input stream class, because we have to go for input process. And then fin is the name of the object of this class we have created, and this is a usual process of creating the object, and giving the explicit mentioning where actually from which file we want to read.

So, in this example we want to read something from testout.txt, which file we have already created like welcome to NPTEL like. And here int i equals fin dot read(), you

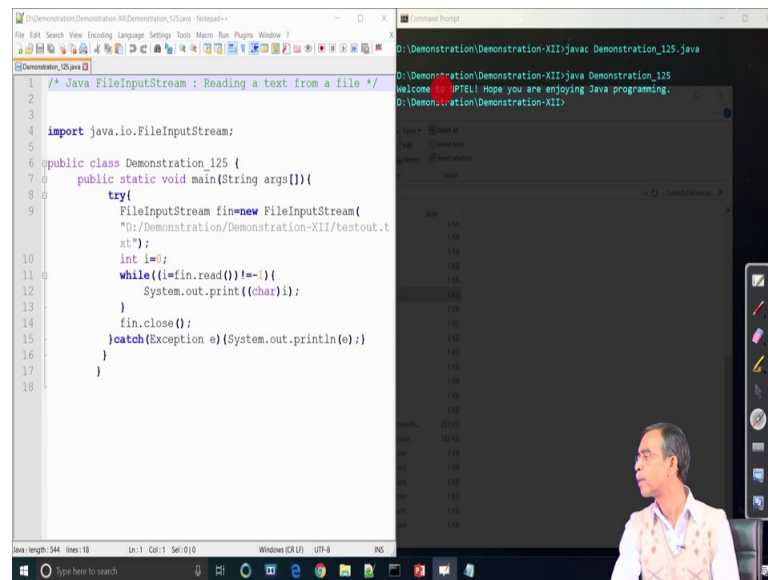
little bit carefully observe for the fin object. A f in object means that is a connection from this program to this file, we inform the method read.

Now, read method always read one character at a time, so this means that the whenever we create the object file object here, it basically point to the first character in that file, and it will basically read that character. And this character actually written by this read method as an ASCII value of the character that means, it will read there. And finally, we can print this character by casting it that means, i its ASCII value will be converting the character corresponding to this, and it will be stored there.

So, you can note that in the previous example testout dot txt, we have store the welcome to the NPTEL. So, W has been store there, and W being the ASCII value, and then it is it will be access it, and then it will display from your program. Now, here you can see here you can see here, you can see W is being printed. Now, if little bit change this program without casting it, you can see here without casting it, we can run the same program again, but only the its integer value rather it is basically not ASCII code, it is the ASCII value.

Now, here we see we print here in case 87 and 87 is basically the ASCII value for the ASCII character W. So, this is basically the way here exactly we save this programme earlier in the byte code form, here we read in that ASCII form byte form, and then printing into its character. So, conversion is it possible. So, this means actually I want emphasize is that whatever the way in the byte or character, you can do it and you can read it also no issue it is there.

(Refer Slide Time: 19:55)



The screenshot shows a Java IDE with two windows. The left window displays the source code for a class named `Demonstration_125`. The code imports `java.io.FileInputStream` and uses a `try` block to create a `FileInputStream` object `fin` pointing to `D:\Demonstration\Demonstration-XII\testout.txt`. It then enters a `while` loop that reads characters from the file until it reaches the end of the file (indicated by `fin.read() != -1`), printing each character. Finally, it closes the file and catches any exceptions. The right window shows the command prompt output after running the program, displaying the text: `Welcome to NPTEL! Hope you are enjoying Java programming.`

```
1  /* Java FileInputStream : Reading a text from a file */
2
3
4  import java.io.FileInputStream;
5
6  public class Demonstration_125 {
7      public static void main(String args[]){
8          try{
9              FileInputStream fin=new FileInputStream(
10                 "D:/Demonstration/Demonstration-XII/testout.t
11                 xt");
12                 int i=0;
13                 while((i=fin.read()) != -1){
14                     System.out.print((char)i);
15                 }
16                 fin.close();
17             }catch(Exception e){System.out.println(e);}
18         }
19     }
20 }
```

```
D:\Demonstration\Demonstration-XII>java Demonstration_125
Welcome to NPTEL! Hope you are enjoying Java programming.
D:\Demonstration\Demonstration-XII>
```

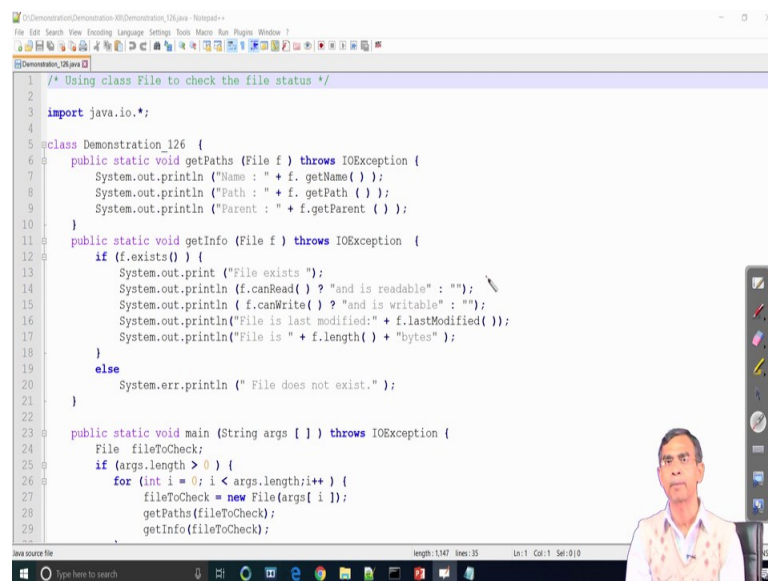
Now, we learn about more about regarding the charactering the stream classes. As we have seen in the last example, we could read only one character from the file. If you want to read the entire character set stream text from a file, what is the program should look like? In this case, again we have to create an object for the `FileInputStream`, because we want to read a file. And here is the object the same way, we created the object a `fin`. And this is the for example, `testout.txt` is the target file from, where you want to read the entire text. As you know in this text, we have written welcome to NPTEL hope you are enjoying java programming like.

Now, here this basically is the loop instead of only single character, we want to read it all the characters which are stored there in this `testout dot file`. Now, a `fin.read`, it is basically to be looped until here is the condition. This is very important not equals to `-1`, so not equals to minus 1 java (`!= -1`) implies that it is the end of file. At every end of file whenever you put file close; the java put a `-1` there indicating that this is the end of the file, so `-1` is a termination condition for a file object.

Now, it will basically read the file content. And here the content which will be read from their `FileInputStream` in the byte form, we convert the character and those things will be converted printed on the screen. And finally, wants the entire file is scanned, it will basically closed. And here you can see it will read the file sequentially the first, then second, and third and so on so on, until it will reach to the end of file.

So, these program again as you can understand anticipate. If we run, it is basically extract all the content, which is there in test out dot txt, and display on your terminal. So, I am running this program, and then will see after running. So, this program has we see here, it basically show the output entire which has been accessed from the text out dot. This is the basically content, which we have stored in the taste out file and this exercise and displayed it.

(Refer Slide Time: 22:03)

A screenshot of a Java IDE window titled "Demonstration_126.java". The code defines a class "Demonstration_126" with two methods: "getPaths" and "getInfo". The "getPaths" method prints the name, path, and parent of a file. The "getInfo" method checks if a file exists and prints its status (readable, writable, last modified, and length). The "main" method takes command-line arguments and iterates through them, creating File objects and calling the "getPaths" and "getInfo" methods. The IDE interface includes a menu bar, toolbar, and a status bar at the bottom showing "length: 1347, lines: 35". A small inset image of a person is visible in the bottom right corner of the IDE window.

```
1  /* Using class File to check the file status */
2
3  import java.io.*;
4
5  class Demonstration_126 {
6      public static void getPaths (File f) throws IOException {
7          System.out.println ("Name : " + f.getName() );
8          System.out.println ("Path : " + f.getPath() );
9          System.out.println ("Parent : " + f.getParent() );
10     }
11     public static void getInfo (File f) throws IOException {
12         if (f.exists() ) {
13             System.out.print ("File exists ");
14             System.out.println (f.canRead() ? "and is readable" : "");
15             System.out.println (f.canWrite() ? "and is writable" : "");
16             System.out.println ("File is last modified:" + f.lastModified());
17             System.out.println ("File is " + f.length() + "bytes");
18         }
19         else
20             System.err.println (" File does not exist." );
21     }
22
23     public static void main (String args [] ) throws IOException {
24         File fileToCheck;
25         if (args.length > 0 ) {
26             for (int i = 0; i < args.length;i++) {
27                 fileToCheck = new File(args[i]);
28                 getPaths(fileToCheck);
29                 getInfo(fileToCheck);
30             }
31         }
32     }
33 }
```

Now, so we are learned about how we can open a file. And then from that file we can read the content in the form of a byte. And our next example is basically regarding the file status checking. So, it is a 12.6 programme, we are going to see about it. Now, here you know so for the status of the file is concerned. Whenever we store a file or operating system maintain all about its file that mean in which directory what is the name of the file, what is the extension, whether the file is in which mode readable or writable, whether file is available or not available, whether file is corrupt, whatever the information it is there. So, it is basically the program, which all these information can be accessed from the program site itself.

Now, here we can see this is the program, and it basically two methods we have discussed here in this class, this is basically our main class. The method is getPaths, and another method is getInfo. Now, so for the getPath method is concerned for a object file object. So, file is basically one class, which is define in java.io package, so .* include

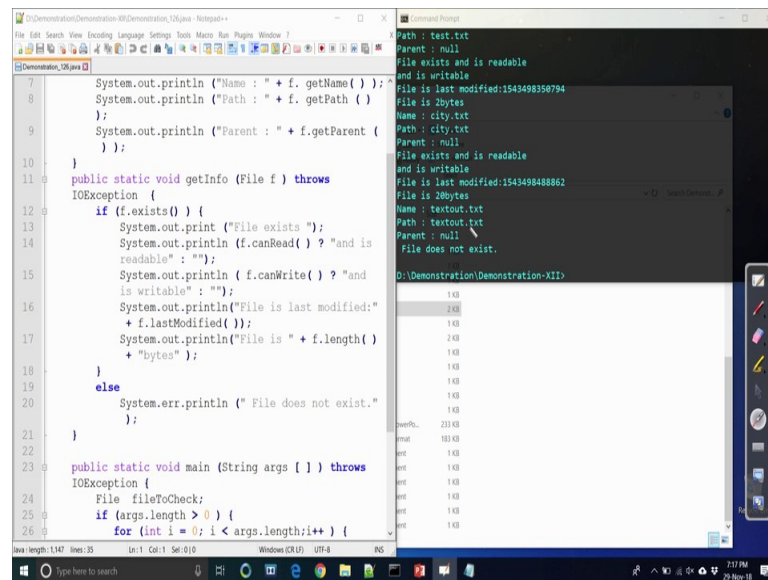
that file package the file class need to be accessed. So, file for this we will pass this file object, whenever you call this getPaths.

And now here you see this getPaths method input simple three statement namely, it will basically give the name of the file `f dot getName()`, the name of the file `f dot getPath()`, which path actually it is stored. And `f dot getParent()` what is the parent, if it is there. And then so far the `getInfo` method is concerned, if the file exist that means, if file is not null, file exist means you can pass information which is no more exist. So, in that case if it is exist, it will just exception.

If it is exist it is true, then it will go for this printing `f dot canRead`, `f dot canWrite` that means, it basically whether this file is readable or write writeable, read mode or write mode, read permission, write permission. And then it also say the last modified whenever you created file, system always store at the last modification of the date of this one so this one. And `f dot length` is basically how many bytes, basically how many byte basically are stored there in this file. So, they are the information.

Now, let us come to the main method here. So, this is the main method this is the main method. Here basically we create one file object name of the object is file to check. And then if `args.length`, basically we can we can pass the command line input, so that we can run this method program for as many file you want to check for it. So, it is basically `args dot length`. And for each file argument we have passed it, it will basically call the `getPaths` and `getInfo` method that means, it will retrieve all the information regarding the file name path and status of it, and finally it will come to the end of this program.

(Refer Slide Time: 25:27)



The screenshot shows an IDE with two windows. The left window displays a Java file named `Demonstration_XII.java` with the following code:

```
7      System.out.println ("Name : " + f. getName() );
8      System.out.println ("Path : " + f. getPath() );
9      System.out.println ("Parent : " + f.getParent (
10     ) );
11
12     public static void getInfo (File f ) throws
13     IOException {
14         if (f.exists() ) {
15             System.out.print ("File exists ");
16             System.out.println (f.canRead() ? "and is
17             readable" : "");
18             System.out.println ( f.canWrite() ? "and
19             is writable" : "");
20             System.out.println("File is last modified:"
21             + f.lastModified());
22             System.out.println("File is " + f.length()
23             + " bytes" );
24         }
25         else
26             System.err.println (" File does not exist."
27             );
28     }
29
30     public static void main (String args [ ] ) throws
31     IOException {
32         File fileToCheck;
33         if (args.length > 0 ) {
34             for (int i = 0; i < args.length;i++ ) {
```

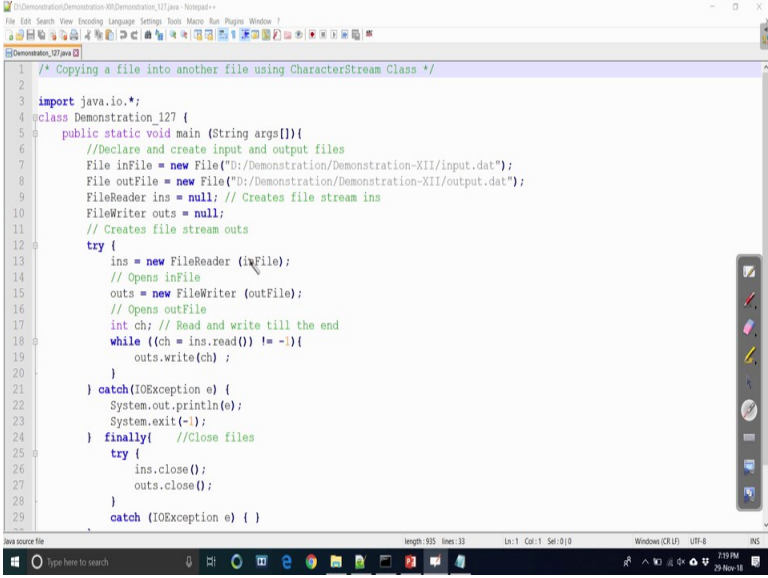
The right window shows the output of the program:

```
Path : test.txt
Parent : null
File exists and is readable
and is writable
File is last modified:1543498350794
File is 2bytes
Name : city.txt
Path : city.txt
Parent : null
File exists and is readable
and is writable
File is last modified:1543498488862
File is 2bytes
Name : testout.txt
Path : testout.txt
Parent : null
File does not exist.
```

Now, will run this program passing three input as a command line, and say the input is input is say city.txt, we know we have created one file test dot txt, testout dot txt. These are three files are already we have created. Now, so we can see ok, we are running this program using passing three input city.txt, test.txt, and textout.txt fine.

Now, here you can see the output as we see path test dot txt parent there is no parent actually indicating null, and then and is writable. Last modified this is basically the system time it is in a some (Refer Time: 26:13) form, and then file size is two bytes. Now, name city text, path is city.txt, parent there is no parent for the, this file. Readable file it is also it is a writable, last modified this one 20 bytes, this also like this. And here you see textout.txt this basically file does not exist, because this file actually it is not there whatever it is here.

(Refer Slide Time: 26:47)



```
1  /* Copying a file into another file using CharacterStream Class */
2
3  import java.io.*;
4  class Demonstration_127 {
5      public static void main (String args[]){
6          //Declare and create input and output files
7          File inFile = new File("D:/Demonstration/Demonstration-XII/input.dat");
8          File outFile = new File("D:/Demonstration/Demonstration-XII/output.dat");
9          FileReader ins = null; // Creates file stream ins
10         FileWriter outs = null;
11         // Creates file stream outs
12         try {
13             ins = new FileReader (inFile);
14             // Opens inFile
15             outs = new FileWriter (outFile);
16             // Opens outFile
17             int ch; // Read and write till the end
18             while ((ch = ins.read()) != -1){
19                 outs.write(ch);
20             }
21         } catch (IOException e) {
22             System.out.println(e);
23             System.exit(-1);
24         } finally{ //Close files
25             try {
26                 ins.close();
27                 outs.close();
28             }
29         } catch (IOException e) { }
```

So, whatever file you have passed to the program, this program from side it can check the status, and can show you. Now, we have discussed about so far the byte stream classes for reading as well as writing. Now, it is our time to learn about other classes, so for the file handling is concerned. They are called FileReader and FileWriter class.

Now, we are going to have a demo. Here in this demo, we will see how we can copy the content of one file into another, so basically making a duplicate, and using FileReader and FileWriter class. So, this program again we make the program for illustration as simple as possible, we can see we fast create one object inFile for the file. This basically creates a connection from this program to this target date input data. Assuming that input.dat file is already there in the system. If it is not there, then it is throw an exception, because file opening will not be possible anyway.

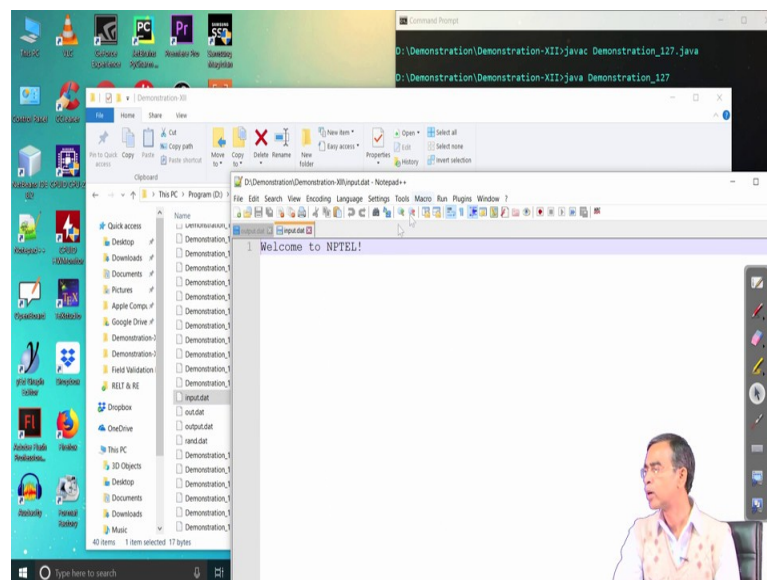
So, this is also another out file, and this basically where the content will store. So, here we have created two file, inFile and outFile for the two purpose; purpose is that reader and writer. So, I just create an object ins, so that in file can be used for the input purpose that means, from in file I will be able to read something. So, inside out are the two objects of type FileReader and FileWriter for the input and output mechanism.

Now, here for ins we create the file reader object for the inFile. So, in this case inFile means, it is basically input data. So, in other words ins means, basically a connection from your program to inFile, inFile means input.dat. Similarly, outs is the connection

from your program to outFile that is the output.dat file. Now, here basically we have to read each and every character those are there input.dat that means, inFile. And then write into the out display on the it is basically read the entire file, and then outs write means it will store into the output file.

Here we see the statement that we have ins read means read one character from the file at a time store as ch the temporary store. And then same ch, it write into the output file outs. So, reading one character from the ins, writing the same character into the outs that mean input date and output date. So, this program and also we have filled up with try and catch you have to always use try and catch block.

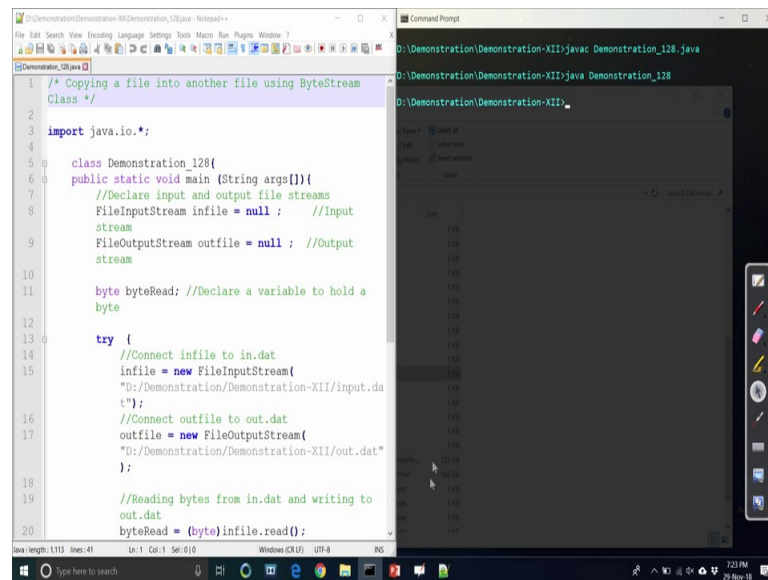
(Refer Slide Time: 29:27)



As there may be some situation unwanted situation, whenever a file is not accessible the file read permission is not there, write permission is there or file is no more memory space is available to avoid so many situations, we have to handle the exception. So, those things can be done by try catch mechanism.

So, you have to handle for every occurrences either creating file or reading or writing, whatever right so they should be put into the try catch flow there. Now, here is basically as we see the output welcome to NPTEL is the input.file. And we copy this the same content, and also create the output file welcome to NPTEL. So, we created the duplicate of one file input dat the name of the duplicated file is output dat.

(Refer Slide Time: 30:29)



```
1  /* Copying a file into another file using ByteStream
2  Class */
3
4  import java.io.*;
5
6  class Demonstration_128{
7      //Declare input and output file streams
8      FileInputStream infile = null ; //Input
9      FileOutputStream outfile = null ; //Output
10     stream
11
12     byte byteRead; //Declare a variable to hold a
13     byte
14
15     try {
16         //Connect infile to in.dat
17         infile = new FileInputStream(
18             "D:/Demonstration/Demonstration-XII/input.dat");
19
20         //Connect outfile to out.dat
21         outfile = new FileOutputStream(
22             "D:/Demonstration/Demonstration-XII/out.dat");
23     }
24
25     //Reading bytes from in.dat and writing to
26     out.dat
27     byteRead = (byte)infile.read();
```

The terminal window shows the command prompt with the following text:

```
D:\Demonstration\Demonstration-XII>javac Demonstration_128.java
D:\Demonstration\Demonstration-XII>java Demonstration_128
```

So, we have learned about how a file copy is possible. Now, here another example in the last example, we have discuss that copying the file using characters, so it will basically character stream classes. We want to do the same thing again, but using byte stream classes. The difference is that in the last example, they will read as a character, but here they will read as a byte that is all.

Now, the mechanism is almost same, but there is again reason that when we have to read character, when we have read byte. Whenever heterogeneous stream is concerned, then you should use the byte stream classes. And homogeneity means in the same system whatever it is there, you have created the file in macro ways, and want to read from the windows, then obviously byte stream is the best procedure. But, if it is the same system (Refer Time: 31:09), then you can use whatever method you can. So, byte stream is always preferable.

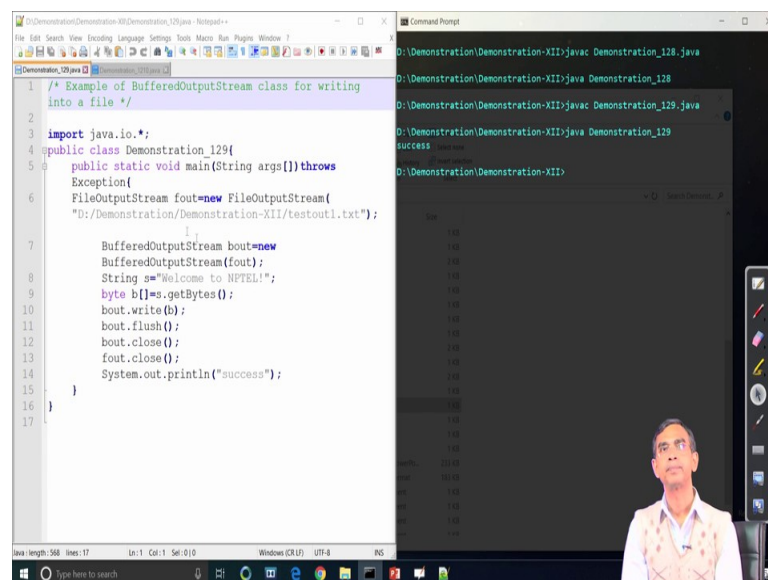
Now, here is a program as we see the same as a inFile, and outFile are the two FileInput Stream and FileOutputStream object. We create the connection inFile, passing this is a target is the input source is the target that means, where the file will be stored there, now here is only mechanism it is difference then the previous one.

So, here byte read temporary store here as a byte in file read, it will read from the in file, store if the byte form, and it is the byte ok. And then by read is again while this is not equal to -1 mean, you have to copy scan the entire file. So, we have to go for loop. And it

read one byte at a time, and the same byte is write into the output file, and again go for reading the next, and then loop continue and until this one is there.

Now, input.dat as you see welcome to NPTEL restored there. And here out.dat file will be created now, if it is successfully copied from that file. So, mechanism is little it different, but the target objective is same object is that we are copying from file to another file yeah, file has been created out.dat file we are going to display the out.dat file yeah, this is the out.dat dat file, you see this is the continue file that we have obtained it after successful compiling competition.

(Refer Slide Time: 32:37)



The screenshot shows a Java IDE on the left and a Command Prompt on the right. The IDE displays a Java file named `Demonstration_129.java` with the following code:

```
1  /* Example of BufferedOutputStream class for writing
2  into a file */
3
4  import java.io.*;
5
6  public class Demonstration_129 {
7      public static void main(String args[]) throws
8      Exception {
9          FileOutputStream fout = new FileOutputStream(
10             "D:/Demonstration/Demonstration-XII/testout1.txt");
11
12             BufferedOutputStream bout = new
13             BufferedOutputStream(fout);
14             String s = "Welcome to NPTEL!";
15             byte b[] = s.getBytes();
16             bout.write(b);
17             bout.flush();
18             bout.close();
19             fout.close();
20             System.out.println("success");
21         }
22     }
23 }
```

The Command Prompt on the right shows the execution of the program:

```
D:\Demonstration\Demonstration-XII>javac Demonstration_129.java
D:\Demonstration\Demonstration-XII>java Demonstration_129
success
```

Now, so this is the idea about, now before concluding our demonstration we want to have two more chance in programming illustration. This is the utility of `BufferedOutputStream` class like also `BufferedOutputStream`, `BufferedInputStream`. In our next two examples, we will demonstrate how the `BufferedOutputStream` and `BufferedInputStream` is possible, it is basically same thing the same idea about byte stream actually, but you use another class that is there already java jdk call the `BufferedOutputStream` class.

Now, let us have a quick look of this program here, we just create file output stream of this, because you want to read something to write into somewhere for the file output stream, we want to write something into testout1 dot txt. This is our target file, where you want to store it from our programme, we want to put somewhere into this target.

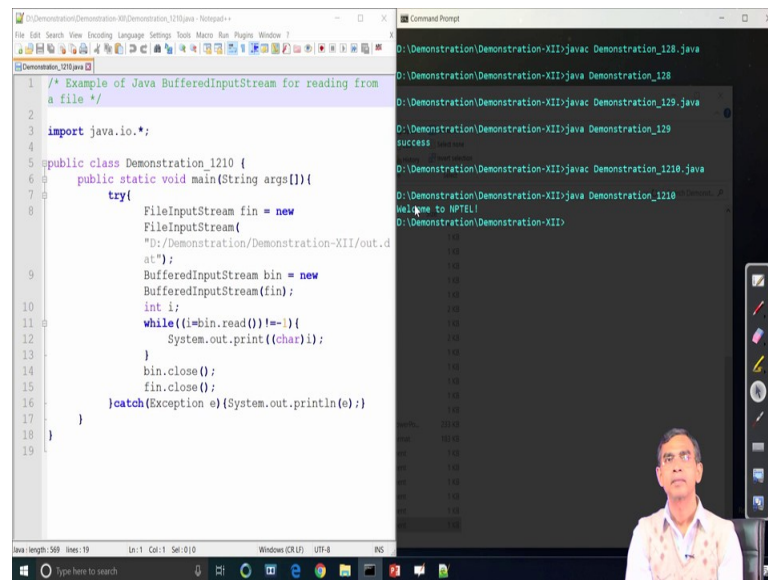
Now, let us see what is the program that you have to put string s welcome to NPTEL, this means that you want to write string into this one, but here we want to store it using this is the object the BufferedOutputStream object call the bout. So, basically bout is a BufferedOutputStream is basically is a connection from the fout. So, the fout to this one that means, it will channelize the content. And then is stored in a BufferedOutputStream form and that in this one.

So, here at this stream s will be from our program, and we convert in the string and here write b byte, and then we will store into the bout form. Now, this idea about that byte buffer output stream classes is there, and this kind of program although we are storing our programme. But, instead of this program if it is a network source at the moment, we do not have any network experience. So, we will not be able to do it with a networking example, but here this s can be forms a network channel.

Now, you will receive the BufferedOutputStream from the network whatever the stream will come, we will buffer it first in our local buffer, and from that buffer we will push into the file target file. So, basically suppose you are downloading an image from the website, and then this image can to be a very large one, so we will buffer into. So, in that case we should use buffered output stream object, so that we can store an image as a buffer form instead of the entire image can be push there, because entire means too large maybe few MB, and you cannot put it there, so it is a buffer form. So, for this purpose, we will do it yes.

Likewise, buffered input basically if we read very large image from the channel network, and writing into a channel network like ok now, let us see here, how we can write this things into this one. And ok, so here the file is successfully content, here we can see directed our output to a file. But, here instead of directing this file, we can direct it into some network port, so that through network this can be transmitted. We will see about the network, whenever you will cover the networking in this module in this course.

(Refer Slide Time: 36:19)



The screenshot shows a Java IDE on the left and a Command Prompt on the right. The IDE contains a Java file named `Demonstration_1210.java` with the following code:

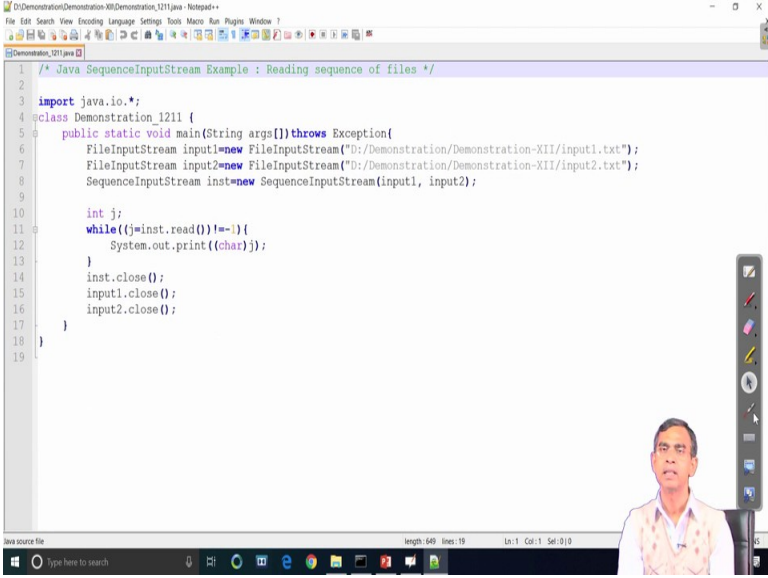
```
1  /* Example of Java BufferedInputStream for reading from
2  a file */
3
4  import java.io.*;
5
6  public class Demonstration_1210 {
7      public static void main(String args[]){
8          try{
9              FileInputStream fin = new
10                 FileInputStream(
11                     "D:/Demonstration/Demonstration-XII/out.d
12                 at");
13             BufferedInputStream bin = new
14                 BufferedInputStream(fin);
15             int i;
16             while((i=bin.read())!=-1){
17                 System.out.print({char}i);
18             }
19             bin.close();
20             fin.close();
21         }catch(Exception e){System.out.println(e);}
22     }
23 }
```

The Command Prompt on the right shows the execution of the program. It displays the output of the `javac` command followed by the execution of `java Demonstration_1210`, which prints the content of the file `out.dat`.

Now, the next example just opposite to this byte output steam buffer, here the byte BufferedInputStream class. Same problem here, basically here we just create the buffer input stream that means, we will read something. Here we will read again from this object out dot dat is already stored there.

So, instead of out.dat here, we can in this source we can maintain the network port number socket all these things. Here we will just read the output at in source of the input. And then it basically read the entire content, and then read in the form of a byte, and then print on the screen. So, it is basically the idea about BufferedInputStream, and then BufferedOutputStream concept yeah this program as you see using the BufferedInputStream, we can access the file.

(Refer Slide Time: 37:19)



```
1  /* Java SequenceInputStream Example : Reading sequence of files */
2
3  import java.io.*;
4  @class Demonstration_1211 {
5      public static void main(String args[]) throws Exception {
6          FileInputStream input1=new FileInputStream("D:/Demonstration/Demonstration-XII/input1.txt");
7          FileInputStream input2=new FileInputStream("D:/Demonstration/Demonstration-XII/input2.txt");
8          SequenceInputStream inst=new SequenceInputStream(input1, input2);
9
10         int j;
11         while((j=inst.read())!=-1){
12             System.out.print((char)j);
13         }
14         inst.close();
15         input1.close();
16         input2.close();
17     }
18 }
19
```

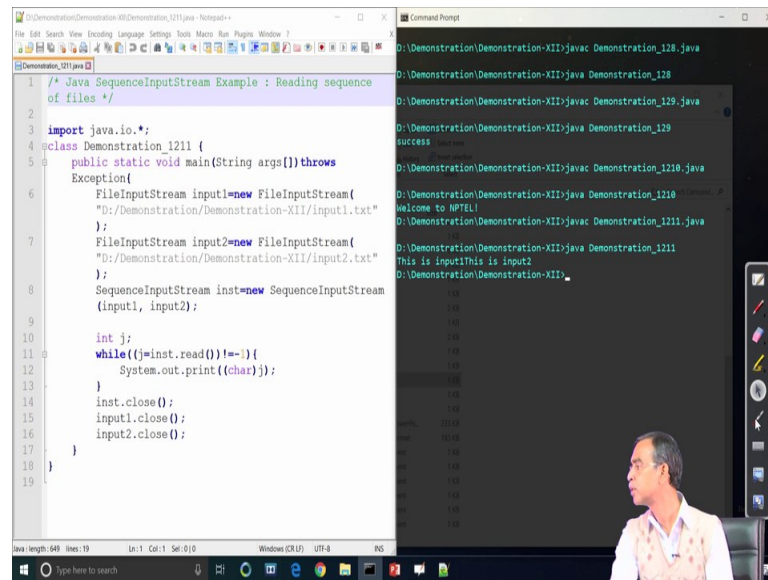
Now, if I ask you the same thing instead of buffer input stream using simple DataInputStream and output stream, you can do the same thing, you can do it actually you have done it already. Now, our next example to illustrate the usage of Sequence InputStream class. Now, so the SequenceInputStream class is basically can handle two more files together. And it will automatically manage it without any intervention within them.

Now, here is the program you see, first we create one input stream FileInputStream namely input1 and another input stream-2. So, here input1, and input2 are the two input target source I can say. And these are two source namely input.txt, input2.txt assuming that there already present in the directory. Now, here you see we create an object call inst, which is of type sequence input stream, this class is again define in java.io package.

So, we can create an object inst, now why we are creating objective you will see, how we can create object passing two input as a overloading constructor of this class input1, input2. Here also we can create input1, input2, input3 also, it will take the overloading constructor will take it. Now, what it will do is basically input1 and input2 the two sources will be scanned one by one, and then the resultant total scanning output will be stored into the inst. And then that inst can be accessed I can read it, and then finally this inst can be display on the screen.

Now, here is basically inst, which is basically the result of sequence input stream, and then will be displayed on the screen. So, it will see the input 1. txt, and input 2.txt, the two content will be displayed on the screen. And finally, we close all the string in input1, and input2 from our system quick.

(Refer Slide Time: 39:31)



The screenshot shows a Java IDE on the left and a Command Prompt on the right. The IDE displays the following code:

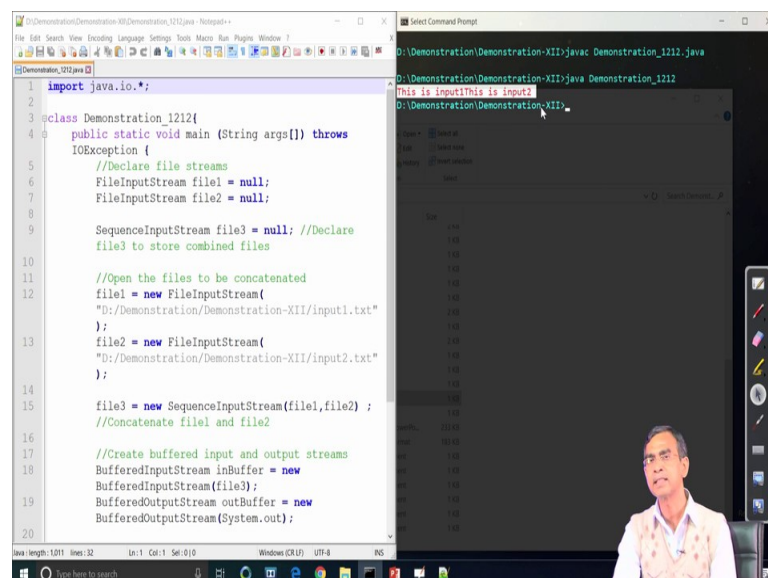
```
1  /* Java SequenceInputStream Example : Reading sequence
2  of files */
3
4  import java.io.*;
5
6  class Demonstration_1211 {
7      public static void main(String args[]) throws
8      Exception{
9          FileInputStream input1=new FileInputStream(
10             "D:/Demonstration/Demonstration-XII/input1.txt"
11         );
12         FileInputStream input2=new FileInputStream(
13             "D:/Demonstration/Demonstration-XII/input2.txt"
14         );
15         SequenceInputStream inst=new SequenceInputStream(
16             input1, input2);
17
18         int j;
19         while((j=inst.read())!=-1){
20             System.out.print((char)j);
21         }
22         inst.close();
23         input1.close();
24         input2.close();
25     }
26 }
```

The Command Prompt shows the execution of the program, displaying the output of the two input files sequentially:

```
D:\Demonstration\Demonstration-XII>java Demonstration_1211
Welcome to NPTEL!
This is input1This is input2
```

As you see here the output as it is shown here, this is a input this is a input two one. So, actually this is the content in the input1.txt and this is a contain input2.txt. So, two files are sequential accessed one after another, and it is basically displayed on this one.

(Refer Slide Time: 39:49)



The screenshot shows a Java IDE on the left and a Command Prompt on the right. The IDE displays the following code:

```
1  import java.io.*;
2
3  class Demonstration_1212{
4      public static void main (String args[]) throws
5      IOException {
6          //Declare file streams
7          FileInputStream file1 = null;
8          FileInputStream file2 = null;
9
10         SequenceInputStream file3 = null; //Declare
11         file3 to store combined files
12
13         //Open the files to be concatenated
14         file1 = new FileInputStream(
15             "D:/Demonstration/Demonstration-XII/input1.txt"
16         );
17         file2 = new FileInputStream(
18             "D:/Demonstration/Demonstration-XII/input2.txt"
19         );
20
21         file3 = new SequenceInputStream(file1,file2) ;
22         //Concatenate file1 and file2
23
24         //Create buffered input and output streams
25         BufferedInputStream inBuffer = new
26         BufferedInputStream(file3);
27         BufferedOutputStream outBuffer = new
28         BufferedOutputStream(System.out);
29     }
30 }
```

The Command Prompt shows the execution of the program, displaying the output of the concatenated files:

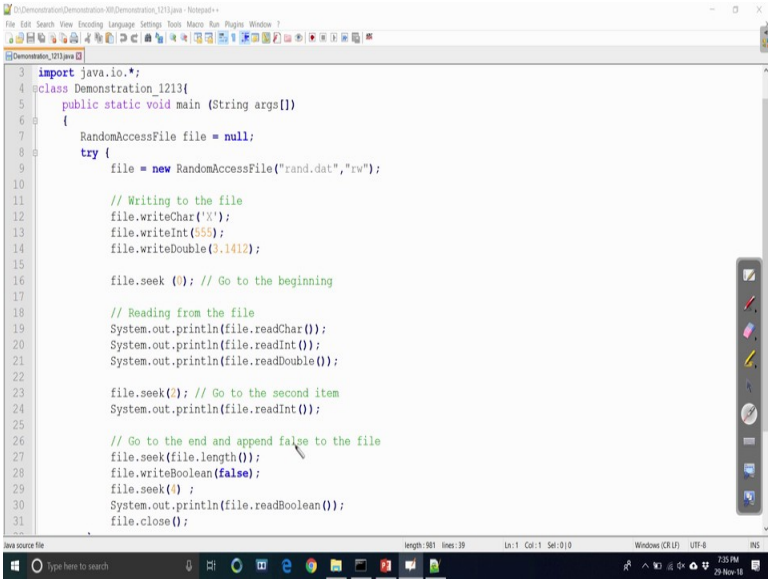
```
D:\Demonstration\Demonstration-XII>java Demonstration_1212
This is input1This is input2
```


So, this is the one usage of the sequence input stream class. Another one application of the sequence input stream class very same concreted margin. Here actually we (Refer Time: 39:58) not stored into the third file. When this program will basically tell you, how the merging result can be stored in the third file is the same as this one. And file1 and file2 just like input file1 input file2, and file 3 is basically the target third file is basically, we will store the concatenation of the two file.

We use the sequence input stream again file 1 and file 2 that means, file 3 will store the merging of the two content. Now, using the buffer input stream, I can read it and then display it. So, here basically in buffer, in out buffer is basically file3 and output is output buffer is basically displaying on the our standard output device.

So, it basically using the by buffer output in form and this program if it is data there, so the concatenation of the two file result will be displayed from the third file on the screen. So, this is a program that we have discuss about using the sequence input stream class example and here is the. So, this is the again program as you see this is an input this is the same thing that is concatenation of the two input file.

(Refer Slide Time: 41:17)



```
3 import java.io.*;
4 class Demonstration_1213{
5     public static void main (String args[])
6     {
7         RandomAccessFile file = null;
8         try {
9             file = new RandomAccessFile("rand.dat", "rw");
10
11             // Writing to the file
12             file.writeChar('X');
13             file.writeInt(555);
14             file.writeDouble(3.1412);
15
16             file.seek (0); // Go to the beginning
17
18             // Reading from the file
19             System.out.println(file.readChar());
20             System.out.println(file.readInt());
21             System.out.println(file.readDouble());
22
23             file.seek(0); // Go to the second item
24             System.out.println(file.readInt());
25
26             // Go to the end and append false to the file
27             file.seek(file.length());
28             file.writeBoolean(false);
29             file.seek(0) ;
30             System.out.println(file.readBoolean());
31             file.close();
32         }
33     }
34 }
```

Now, we will discuss about random access, this is the last content to have the demonstration. Random access is just different mechanism the totally different, then the concept that you have learn so far. In some situation, we have to access the file in a random manner, whatever the things we have discuss that we discuss the file in a

sequence manner that means, one character the next character one by the next byte until the end of file likely, but here we can read at random.

So, this program can have a quick demo about it. And you can use the file class for this, so here we just Random access file rather. There is a class `RandomAccessFile`, which is defining java.io file, we have to create an object for this. So, we create the file is that I `RandomAccessFile`. And this file whenever we created, it can be open in any read write. So, other unlike sequence access, it can be read either read or write, but it can be open both mode.

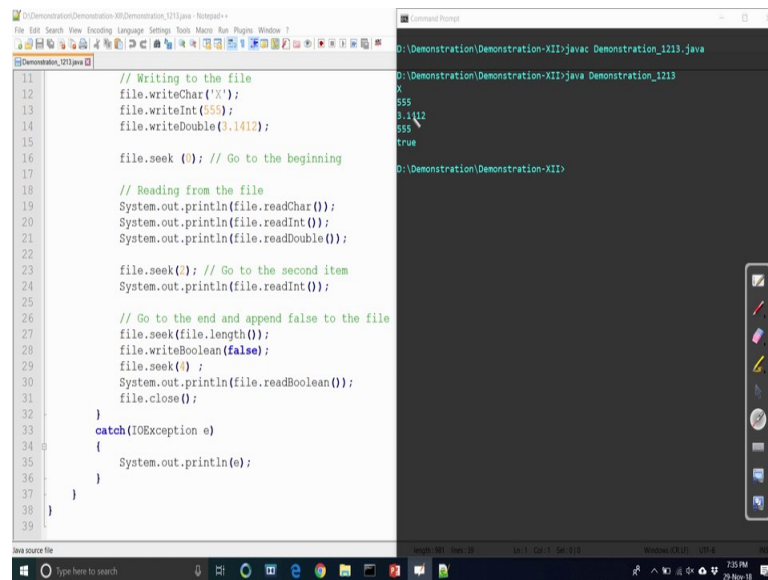
Now, here we create the file object and this is connected to the file. So, you want to randomly access rand data reading, writing on the same file at the same time like. So, here you see we first write something `writeChar`, `writeInt`, and `writeDouble`, and these are the input that we have to write into this that means, if the file is blank or if the file is there if we write it from very beginning, it is basically overwriting. Then entire content will be deleted, and it will store this one.

And here is the file seek is basically positioning the file pointer, so it basically zero indicates that, it will position file pointer at the very beginning that means, where the x is now written there. Now, here `system.out.println (file.readchar())` that means, presently seek 0 that means start here. This statement will read x, and then again after reading x automatically file pointer move to the next one that means, read come here and `readInt`, it will read this one. Go to the next one, `readDouble`, and it will read this one.

So, this is basically reading and writing from the same file as you can see there. Now, here again seek two as you see that file position will be move to the second location. So, 1, 2, then it will go there. And then again we `readInt` that means, you can read the 555 here. And then file dot seek (file dot length), so `file.length` as you know, it will give you the size of the file.

So, if the seek is go there means, it will move the file pointer to the end of the file. And at the end of the file `writeBoolean false` that means, we write a Boolean value and the false. And files dot seek (4), again we go to the forth position 1, 2, 3, 4 means, it will come here. Then we can file read Boolean, so read Boolean means it will false and then finally file close. So, this is basically shows a very quickly shows that how the random access mechanism can be applied to a file object.

(Refer Slide Time: 44:13)



The screenshot displays a Java IDE on the left and a terminal window on the right. The IDE shows the source code for `Demonstration_1213.java`, which demonstrates file I/O operations using `FileWriter` and `FileReader`. The code includes comments for seeking to the beginning, second item, and end of the file. The terminal window shows the output of the program, which reads the file and prints the character 'X', the integer 555, the double 3.1412, and the boolean true.

```
// Writing to the file
file.writeChar('X');
file.writeInt(555);
file.writeDouble(3.1412);

file.seek(0); // Go to the beginning

// Reading from the file
System.out.println(file.readChar());
System.out.println(file.readInt());
System.out.println(file.readDouble());

file.seek(0); // Go to the second item
System.out.println(file.readInt());

// Go to the end and append false to the file
file.seek(file.length());
file.writeBoolean(false);
file.seek(0);
System.out.println(file.readBoolean());
file.close();

}
catch (IOException e)
{
    System.out.println(e);
}
}
```

```
D:\Demonstration\Demonstration-XII>javac Demonstration_1213.java
D:\Demonstration\Demonstration-XII>java Demonstration_1213
X
555
3.1412
555
true
D:\Demonstration\Demonstration-XII>
```

Now, here is a quick execution of the program, so that we can learn about it. And then you can see the difference state that it will show here as you see here, this is the first read integer write, read character, write integer, and then write double, and then again read, and then again finally go to the file position at the end, where the true is store and we read attribute and then get it. So, this way we will be able to access the file in a random way. This is a very small example, we have discussed about it.

Now, I hope you have understood the concept of file input output mechanism in java. Input output mechanism, it just started here. The more input output mechanism will be discussed, whenever we will discussed. The graphical user interface concept there is a lot of input output in a different fashion, the different style we have to follow. But, all those lesson that we have learned, we will be utilize there. In addition to this also few more in those context, we discussed ok, we should wait for the next topic that we are going to cover, it is basically graphical user interface programming our next topics to be covered.

Thank you very much, thanks for your attention.