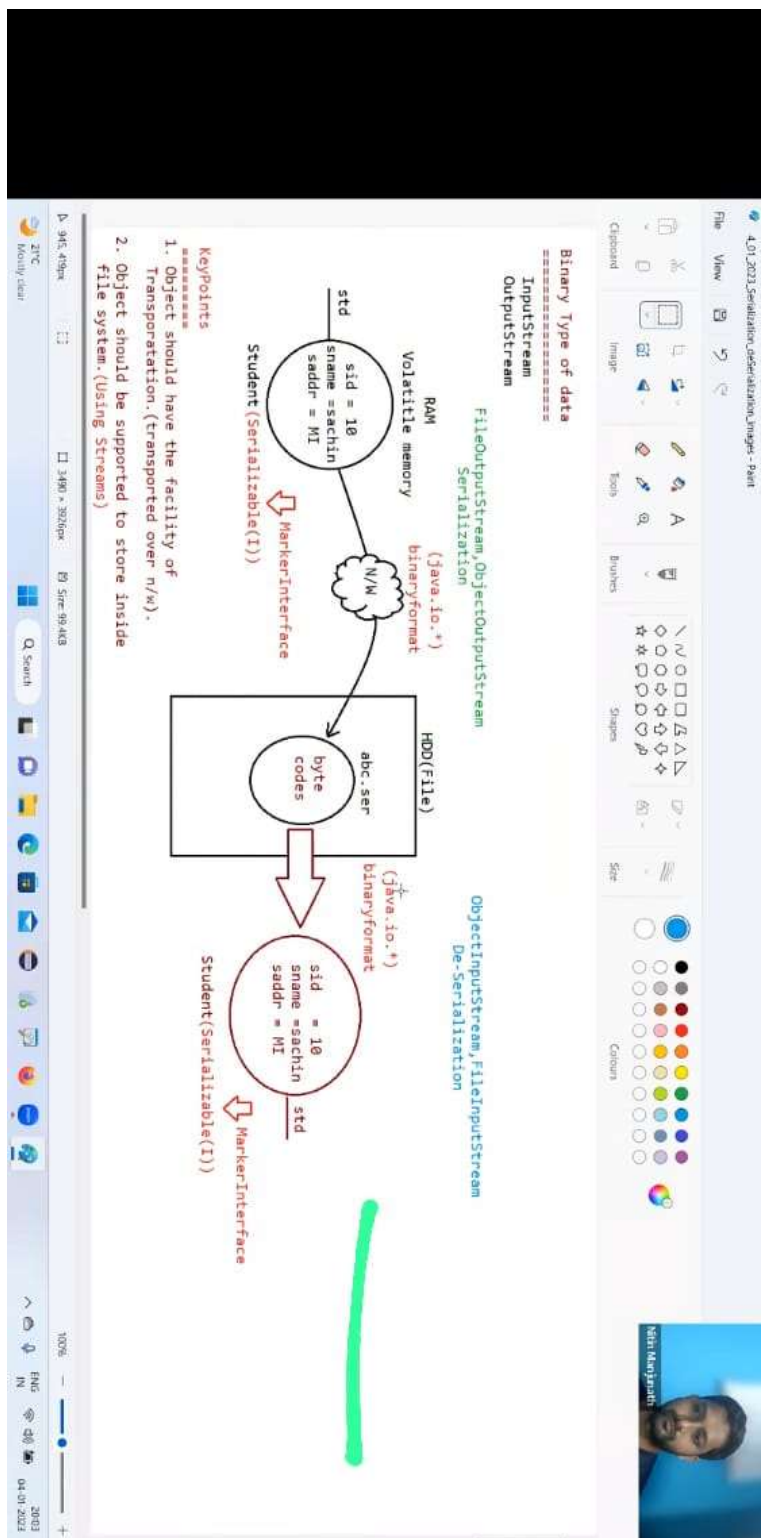


45Java_Serialization_Deserialization



10. Difference between Serialization & Externalization

11. SerializableUID

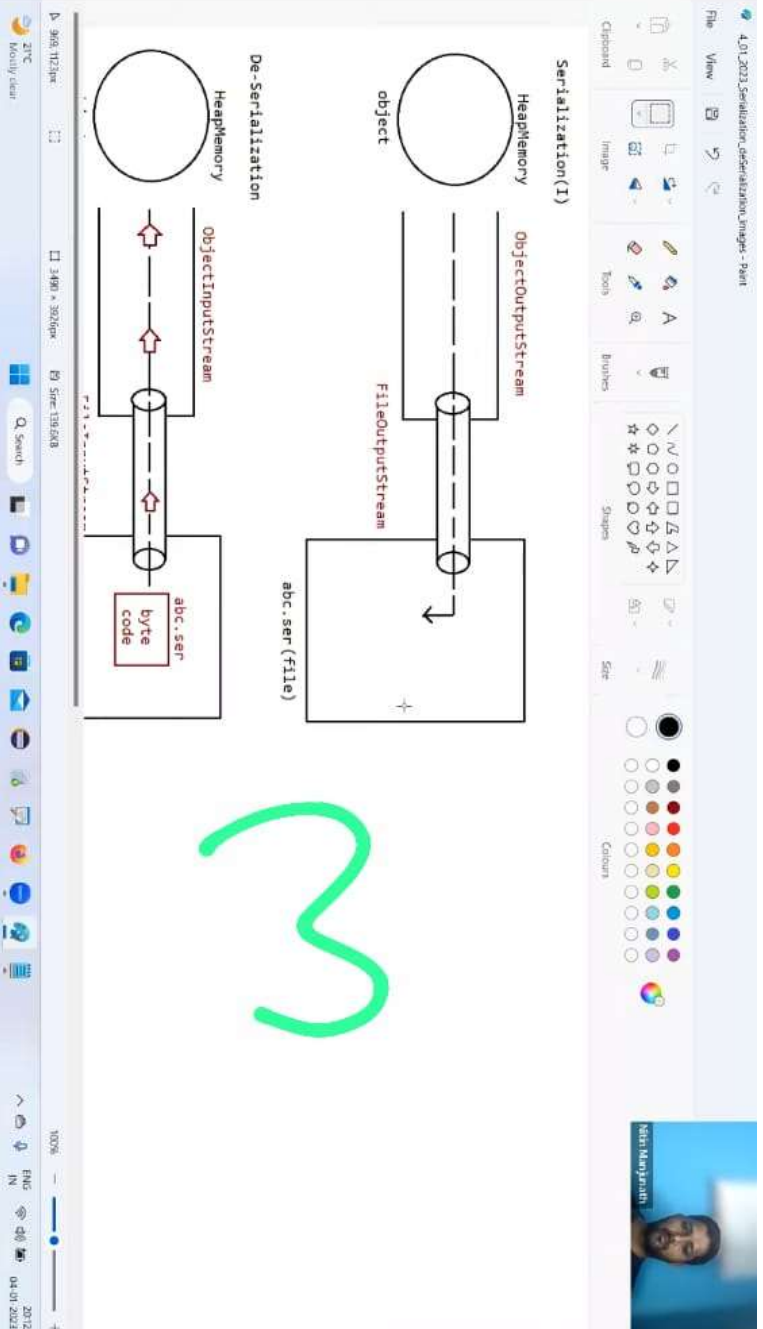
Serialization: (1.1 v)

- => The process of saving (or) writing state of an object to a file is called serialization but strictly speaking it is the process of converting an object from java supported form to either network supported form (or) file supported form.
- => By using FileOutputStream and ObjectOutputStream classes we can achieve serialization process.
 - |=> writeObject(Object obj)

De-Serialization:

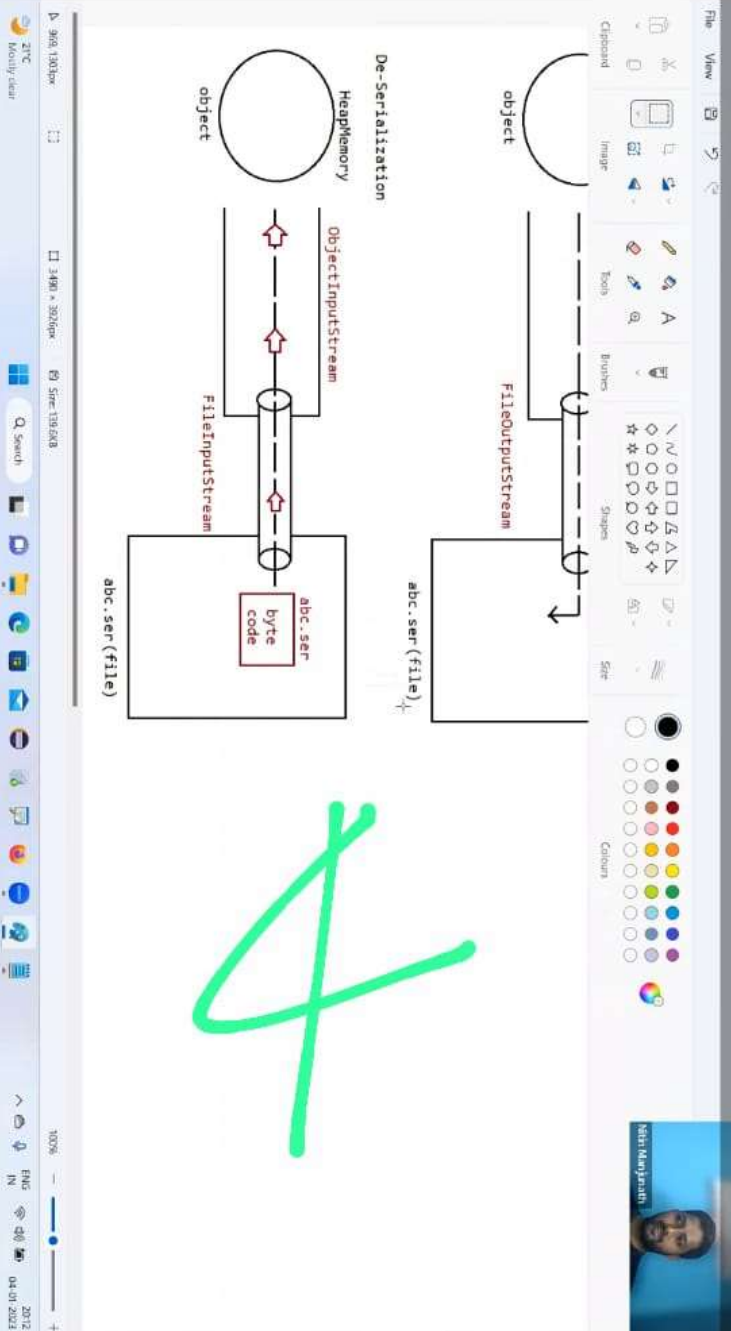
- => The process of reading state of an object from a file is called DeSerialization but strictly speaking it is the process of converting an object from file supported form (or) network supported form to java supported form.
- => By using FileInputStream and ObjectInputStream classes we can achieve DeSerialization.
 - |=> readObject()

2



8:24 AM

Serialisation, Deserialisation Images - Paint



LTE .all K/s 53

Command Prompt

D:\100parations>javap java.io.ObjectOutputStream

Compiled from "ObjectOutputStream.java"

public class java.io.ObjectOutputStream extends java.io.OutputStream implements java.io.ObjectOutput, java.

tants {

public java.io.ObjectOutputStream(java.io.OutputStream) throws java.io.IOException;

protected java.io.ObjectOutputStream() throws java.io.IOException, java.lang.SecurityException;

public void useProtocolVersion(int) throws java.io.IOException;

public final void writeObject(java.lang.Object) throws java.io.IOException;

protected void writeObjectOverride(java.lang.Object) throws java.io.IOException;

public void writeUnshared(java.lang.Object) throws java.io.IOException;

public void defaultWriteObject() throws java.io.IOException;

public java.io.ObjectOutputStream\$PutField putFields() throws java.io.IOException;

public void writeFields() throws java.io.IOException;

public void reset() throws java.io.IOException;

protected void annotateClass(java.lang.Class?) throws java.io.IOException;

protected void annotateProxyClass(java.lang.Class?) throws java.io.IOException;

protected java.lang.Object replaceObject(java.lang.Object) throws java.io.IOException;

protected boolean enableReplaceObject(boolean) throws java.lang.SecurityException;

protected void writeStreamHeader() throws java.io.IOException;

protected void writeClassDescriptor(java.io.ObjectStreamClass) throws java.io.IOException;

public void write(int) throws java.io.IOException;

public void write(byte[]) throws java.io.IOException;

public void write(byte[], int, int) throws java.io.IOException;

public void flush() throws java.io.IOException;

protected void drain() throws java.io.IOException;

public void close() throws java.io.IOException;

public void writeBoolean(boolean) throws java.io.IOException;

public void writeByte(int) throws java.io.IOException;

public void writeShort(int) throws java.io.IOException;

public void writeChar(int) throws java.io.IOException;

public void writeInt(int) throws java.io.IOException;



✓

27°C
Monday, June

Search



FMG
IN

2016
04-01-2023

Command Prompt

X + V

D:\IOoperations>javap java.io.FileOutputStream

Compiled from "FileOutputStream.java"

```
public class java.io.FileOutputStream extends java.io.OutputStream {  
    public java.io.FileOutputStream(java.lang.String) throws java.io.FileNotFoundException;  
    public java.io.FileOutputStream(java.io.File) throws java.io.FileNotFoundException;  
    public java.io.FileOutputStream(java.io.File, boolean) throws java.io.FileNotFoundException;  
    public java.io.FileOutputStream(java.io.FileDescriptor);  
    public void write(int) throws java.io.IOException;  
    public void write(byte[]) throws java.io.IOException;  
    public void write(byte[], int, int) throws java.io.IOException;  
    public void close() throws java.io.IOException;  
    public final java.io.FileDescriptor getFD() throws java.io.IOException;  
    public java.nio.channels.FileChannel getChannel();  
    protected void finalize() throws java.io.IOException;  
    static void access$000(java.io.FileOutputStream) throws java.io.IOException;  
    static {};  
}
```

D:\IOoperations>



6

ZTC
Bollywood

Q Search



EN5
IN

04-01-2023

Command Prompt

D:\100operations>java java.io.ObjectOutputStream

Compiled from "ObjectOutputStream.java"

```
public class java.io.ObjectOutputStream extends java.io.OutputStream implements java.io.ObjectOutput, java.io.ObjectStreamable {
    public java.io.ObjectOutputStream(java.io.OutputStream) throws java.io.IOException, java.lang.SecurityException;
    protected java.io.ObjectOutputStream() throws java.io.IOException, java.lang.SecurityException;
    public void useProtocolVersion(int) throws java.io.IOException;
    public final void writeObject(java.lang.Object) throws java.io.IOException;
    protected void writeObjectOverride(java.lang.Object) throws java.io.IOException;
    public void writeUnshared(java.lang.Object) throws java.io.IOException;
    public void defaultWriteObject() throws java.io.IOException;
    public java.io.ObjectOutputStream.putFields() throws java.io.IOException;
    public void writeFields() throws java.io.IOException;
    public void reset() throws java.io.IOException;
    protected void annotateClass(java.lang.Class?) throws java.io.IOException;
    protected void annotateProxyClass(java.lang.Class?) throws java.io.IOException;
    protected java.lang.Object replaceObject(java.lang.Object) throws java.io.IOException;
    protected boolean enableReplaceObject(boolean) throws java.lang.SecurityException;
    protected void writeStreamHeader() throws java.io.IOException;
    public void write(int) throws java.io.IOException;
    public void write(byte[]) throws java.io.IOException;
    public void write(byte[], int, int) throws java.io.IOException;
    public void flush() throws java.io.IOException;
    protected void drain() throws java.io.IOException;
    public void close() throws java.io.IOException;
    public void writeBoolean(boolean) throws java.io.IOException;
    public void writeByte(int) throws java.io.IOException;
    public void writeShort(int) throws java.io.IOException;
    public void writeChar(int) throws java.io.IOException;
    public void writeInt(int) throws java.io.IOException;
    public void writeLong(long) throws java.io.IOException;
    public void writeFloat(float) throws java.io.IOException;
```



Nishu Mangrulkar

ZTC
Brosify Lear

Search

FM5

IN

20:20
04-01-2023

Compiled from "ObjectInputStream.java"

```
public class java.io.ObjectInputStream extends java.io.InputStream implements java.io.ObjectInput, java.io.ObjectStreamConstants {
    public java.io.ObjectInputStream(java.io.InputStream) throws java.io.IOException;
    protected java.io.ObjectInputStream() throws java.io.IOException;
    protected final java.lang.Object readObject() throws java.io.IOException;
    protected java.lang.Object readObjectOverride() throws java.io.IOException;
    protected java.lang.Object readUnshared() throws java.io.IOException;
    public java.lang.Object readObject() throws java.io.IOException;
    public void defaultReadObject() throws java.io.IOException;
    public java.io.ObjectInputStream$StreamField readField() throws java.io.IOException;
    public void registerValidation(java.io.ObjectInputValidation) throws java.io.IOException;
    protected java.lang.Class? resolveProxyClass(java.lang.String[]) throws java.io.IOException;
    protected java.lang.Object resolveObject(java.lang.Object) throws java.io.IOException;
    protected boolean enableResolveObject(boolean) throws java.io.IOException;
    protected void readStreamHeader() throws java.io.IOException;
    protected java.io.ObjectStreamClass readClassDescriptor() throws java.io.IOException;
    public int read() throws java.io.IOException;
    public int read(byte[], int, int) throws java.io.IOException;
    public int available() throws java.io.IOException;
    public void close() throws java.io.IOException;
    public boolean readBoolean() throws java.io.IOException;
    public byte readByte() throws java.io.IOException;
    public int readSignedByte() throws java.io.IOException;
    public char readChar() throws java.io.IOException;
    public short readShort() throws java.io.IOException;
    public int readSignedShort() throws java.io.IOException;
    public int readInt() throws java.io.IOException;
    public long readLong() throws java.io.IOException;
    public float readFloat() throws java.io.IOException;
    public double readDouble() throws java.io.IOException;
    public void readFully(byte[]) throws java.io.IOException;
    public int readFully(byte[], int, int) throws java.io.IOException;
    public int skipBytes(int) throws java.io.IOException;
    public java.lang.String readLine() throws java.io.IOException;
    public java.lang.String readUTF() throws java.io.IOException;
    public java.lang.String readUTF8() throws java.io.IOException;
    static void access$800(java.io.ObjectInputStream, sun.misc.ObjectInputFilter);
    static sun.misc.ObjectInputFilter access$1800(java.io.ObjectInputStream);
    static int access$500(java.io.ObjectInputStream);
```

Nati Hanyuan

The screenshot displays a Java IDE with the following code in the editor:

```
1 import java.io.*;
2 /*
3  public java.io.ObjectOutputStream(java.io.OutputStream) throws java.io.IOException;
4  public java.io.ObjectOutputStream(java.lang.String) throws java.io.FileNotFoundException;
5  public final void writeObject(java.lang.Object) throws java.io.IOException;
6
7  public java.io.ObjectInputStream(java.io.InputStream) throws java.io.IOException;
8  public java.io.FileInputStream(java.lang.String) throws java.io.FileNotFoundException;
9  public final java.lang.Object readObject() throws java.io.IOException, java.lang.ClassNotFoundException;
10 */
11
12 class Dog implements Serializable
13 {
14     static{
15         System.out.println("static block gets executed...");
16     }
17     Dog(){
18         System.out.println("Object is created...");
19     }
20
21     int i = 10;
22     int j = 20;
23 }
```

A green handwritten checkmark is visible in the bottom right corner of the code area. The IDE's interface includes a top menu bar with options like File, Edit, View, Search, Environment, Project, Tools, Browser, Editor, Window, and Help. The left sidebar shows a Project Explorer with a directory structure containing 'src' and 'lib' folders. The bottom status bar indicates the file is 'Serializable.java' and the cursor is at line 20, column 4.

File Edit View Search Run Window Help

Project Tools Browser Register Window Help

Directory Content Functions

D:\New Volume

13={

14={

15 static{

16 system.out.println("static block gets executed...");

17 }

18 Dog(){

19 system.out.println("Object is created...");

20 }

21 int i = 10;

22 int j = 20;

23 }

24 }

25 class Test

26 {

27 public static void main(String[] args)throws Exception

28 {

29 Dog d = new Dog();

30 }

31 System.out.println("Serialization started....");

32 String fileName = "abc.ser";

33 FileOutputStream fos = new FileOutputStream(fileName);

34 ObjectOutputStream oos = new ObjectOutputStream(fos);

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 }

<

31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
48
49
50
51

```
System.out.println("Serialization started.....");
String fileName = "abc.ser";
FileOutputStream fos = new FileOutputStream(fileName);
ObjectOutputStream oos = new ObjectOutputStream(fos);
oos.writeObject(d);
System.out.println("Serialized Object reference is ::"+d);
System.out.println("Serialization ended.....");

//To pause the execution till we press some key from keyboard
System.in.read();

System.out.println("De-Serialization started.....");
FileInputStream fis = new FileInputStream("abc.ser");
ObjectInputStream ois = new ObjectInputStream(fis);
Object obj=ois.readObject();
Dog d1 = (Dog)obj;
System.out.println("De-Serialized Object reference is ::"+d1);
System.out.println("De-Serialization ended.....");
}
//JVM shutdown now
```

1234567

20°C
Boudhatar

Q Search

in 39 607.74 54 00 PC ANS

04-07-2023

Neer Marjani

File Edit View

Serialization_PackSerialization_Power

Deserialization started

Deserialization ended

Dog object data

10 20

Cat object data

100 200

Note:

1. We can perform Serialization only for Serializable objects.
2. An object is said to be Serializable if and only if the corresponding class implements Serializable interface.
3. Serializable interface present in java.io package and does not contain any methods. It is marker interface. The required ability will be provided automatically by JVM.
4. We can add any no. Of objects to the file and we can read all those objects from the file but in which order we wrote objects in the same order only the objects will come back. That is order is important.
5. If we are trying to serialize a non-serializable object then we will get RuntimeException saying "NotSerializableException".



4.01.2023,Serialization,deSerialization,Images - Paint

FileView

Clipboard

Image

Tools

Brushes

Shapes

Size

Colors

Console

14

B

I

U

S

Background fill

Serialization(1)

HeapMemory

ObjectOutputStream

#1

FileOutputStream

abc.ser (file)

Object #3

i = 10

j = 20

De-Serialization

HeapMemory

ObjectInputStream

FileInputStream

abc.ser

byte code

Object #1

i = 10

j = 20

13

20°C

Brandy bear

Search

1488 x 3026px

File Src: 140.77.8

100%

20:57

04-01-2023

Nar Marjan

40 = {
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60

1234567

1234567

Dog d = new Dog();
Cat d = new Cat();

System.out.println("Serialization started.....");
String fileName = "abc.ser";
FileOutputStream fos = new FileOutputStream(fileName);
ObjectOutputStream oos = new ObjectOutputStream(fos);
oos.writeObject(d);
System.out.println("Serialized Object reference is ::"+d);
System.out.println("Serialization ended.....");

//To pause the execution till we press some key from keyboard
System.in.read();

System.out.println("De-Serialization started.....");
FileInputStream fis = new FileInputStream("abc.ser");
ObjectInputStream ois = new ObjectInputStream(fis);
Object obj=ois.readObject();
Dog d1 = (Dog)obj;
System.out.println("De-Serialized Object reference is ::"+d1);
Custom out.println("De-Serialized Object reference is ::"+d1);

20°C
Boudydear

14:42 100% 31 07 00 PC ANSI 04-01-2023

Mr. Karjunn

8:52 AM

```
File Edit View Search Document Project Tools Browser Frame Window Help
D:\New Volume
Directory Clipboard Functions
D:\
IOOperations
Test.java
61
62
63
64
65
66
67
68
69
55 System.out.println("De-Serialization started.....");
56 FileInputStream fis = new FileInputStream("abc.ser");
57 ObjectInputStream ois = new ObjectInputStream(fis);
58 Dog d1=(Dog)ois.readObject();
59 Cat c1=(Cat)ois.readObject();
60
61
62 System.out.println("Dog object data is:: "+d1.i+"---> "+d1.j);
63 System.out.println("Cat object data is:: "+c1.i+"---> "+c1.j);
64 System.out.println("De-Serialization ended.....");
65
66 }
67 //JVM shutdown now
68 }
69 }
```

System.out.println("De-Serialization started.....");
FileInputStream fis = new FileInputStream("abc.ser");
ObjectInputStream ois = new ObjectInputStream(fis);
Dog d1=(Dog)ois.readObject();
Cat c1=(Cat)ois.readObject();

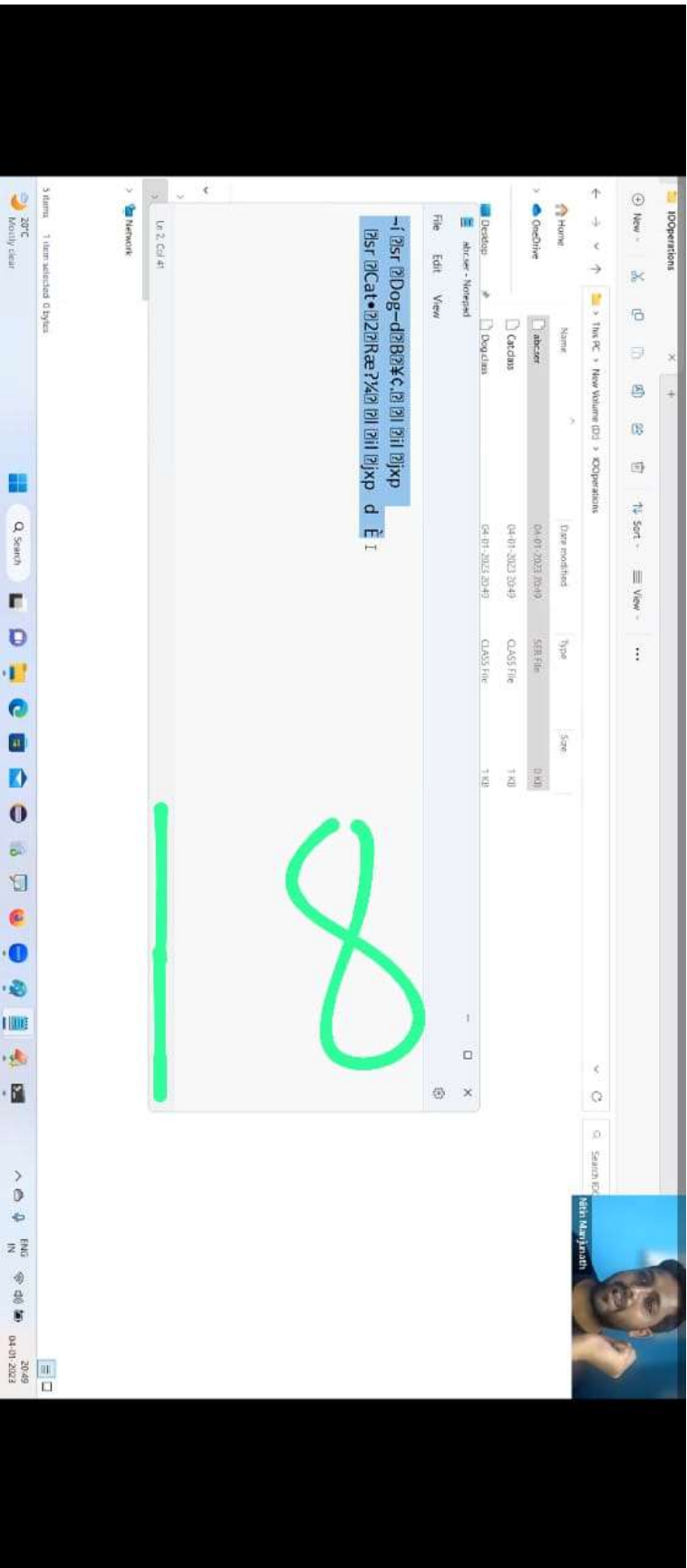
System.out.println("Dog object data is:: "+d1.i+"---> "+d1.j);
System.out.println("Cat object data is:: "+c1.i+"---> "+c1.j);
System.out.println("De-Serialization ended.....");

}
//JVM shutdown now
}
}

1 2 3 4 5 6 7

in 62 cod 70 09 31 PC ANS
FMS 2048
IN 04-07-2023

49





Nishit Manojanathi



Nishit Manojanathi



Nishit Manojanathi



Nishit Manojanathi



Nishit Manojanathi



Nishit Manojanathi

19

Note:

1. We can perform Serialization only for Serializable objects.
2. An object is said to be Serializable if and only if the corresponding class implements Serializable interface.
3. Serializable interface present in java.io package and does not contain any methods. It is marker interface. The required ability will be provided automatically by JVM.
4. We can add any no. Of objects to the file and we can read all those objects from the file but in which order we wrote objects in the same order only the objects will come back. That is order is important.
5. If we are trying to serialize a non-serializable object then we will get RuntimeException saying "NotSerializableException".

Transient keyword:

1. transient is the modifier applicable only for variables, but not for classes and methods.
2. While performing serialization if we don't want to save the value of a particular variable to meet security constant such type of variable, then we should declare that variable with "transient" keyword.
3. At the time of serialization JVM ignores the original value of transient variable and save default value to the file.
4. That is transient means "not to serialize".

eg#1.

```
import java.io.FileOutputStream;  
import java.io.IOException;  
import java.io.ObjectOutputStream;  
import java.io.FileInputStream;
```



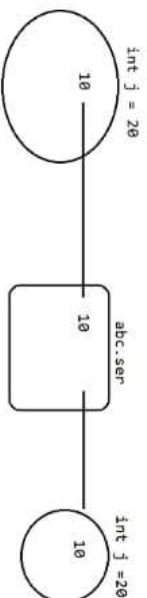


4.01.2023.Serialization.DiceGame.Images - Paint

File View



```
class Dog implements Serializable
{
    final transient int i = 10;
    static transient int j = 20;
}
```



transient -> variable don't participate in serialization

final and transient -> final means variable won't come into picture it is directly the values
static and transient -> variable which is static is not a part of Object, so it is not a part of Serialization.

21



Declaration output

=====

1.

int i=10;

int j=20;

output

10 ----- 20

2.

transient int i=10;

int j=20;

output

0-----20

3.

transient int i=10;

23



int j=20;

output

0-----20
 |

3. transient int i=10;
transient static int j=20;

output

0-----20

4. transient final int i=10;
transient int j=20;

5. transient final int i=10;
transient static int j=20;

24



output

0-----20

4.

transient final int i=10;
transient int j=20;

output

10-----0

5.

transient final int i=10;
transient static int j=20;
}

output

10-----20

25





```
Object o = oos.readObject();
```

```

if (o instanceof Dog)
    // perform Dog related operation

if (o instanceof Cat)
    // perform Cat related operation

if (o instanceof Rat)
    // perform Rat related operation

```

26

```
}  
    if(obj instanceof Cat){  
        Cat C=(Cat)obj;  
        //perform operation related to Cat  
    }  
    if(obj instanceof Rat){  
        Rat r=(Rat)obj;  
        //perform operation related to Rat  
    }  
}
```

Object graph in serialization:

1. Whenever we are serializing an object the set of all objects which are reachable from that object will be serialized automatically. This group of objects is nothing but object graph in serialization.
2. In object graph every object should be Serializable otherwise we will get runtime exception saying "NotSerializableException".

eg#1.
import java.io.Serializable;

27



File Edit View Search Run Window Help

Directory: C:\Users\... \Desktop

1 2 3 4 5 6 7


```
7 public java.io.ObjectInputStream(java.io.InputStream) throws java.io.IOException;
8 public java.io.FileInputStream(java.lang.String) throws java.io.FileNotFoundException;
9 public final java.lang.Object readObject() throws java.io.IOException, java.lang.ClassNotFoundException;
10 */
11
12 class Dog implements Serializable
13 {
14     Cat c = new Cat();
15 }
16 class Cat implements Serializable
17 {
18     Rat r = new Rat();
19 }
20 class Rat implements Serializable
21 {
22     int j = 20;
23 }
24 class Test
25 {
26     public static void main(String[] args) throws Exception
27     {
28         Dog d = new Dog();
29     }
30 }
```

Test.java

Test.java

10:44 10:34 31 2F PC ANS 27:50 04-07-2023

28



File Edit View Search Document Project Tools Browser Engine Window Help

Directory Content Functions

D:\New Volume

D:\

IOOperations

Test.java

22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42

```
int j = 20;
class Test
{
    public static void main(String[] args) throws Exception
    {
        Dog d = new Dog();

        System.out.println("Serialization started.....");
        String fileName = "abc.ser";
        FileOutputStream fos = new FileOutputStream(fileName);
        ObjectOutputStream oos = new ObjectOutputStream(fos);
        oos.writeObject(d);
        System.out.println("Serialization ended.....");

        //To pause the execution till we press some key from keyboard
        System.in.read();

        System.out.println("De-Serialization started.....");
        FileInputStream fis = new FileInputStream("abc.ser");
        ObjectInputStream ois = new ObjectInputStream(fis);
        Dog d1 = (Dog)ois.readObject();
```

Test.java


File Edit View Search Document Project Tools Browser Engine Window Help

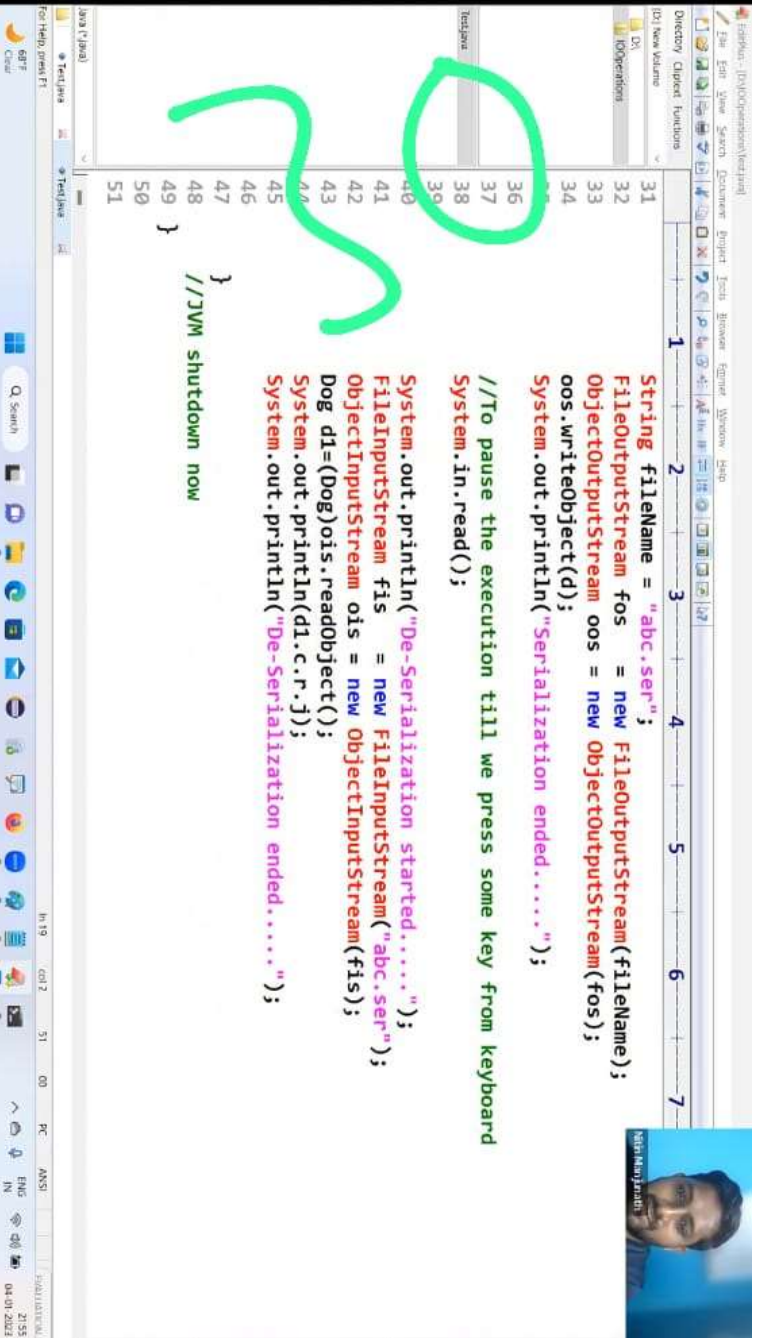
Q Search

Taskbar icons

16:27 60% 31 00 PC ANS

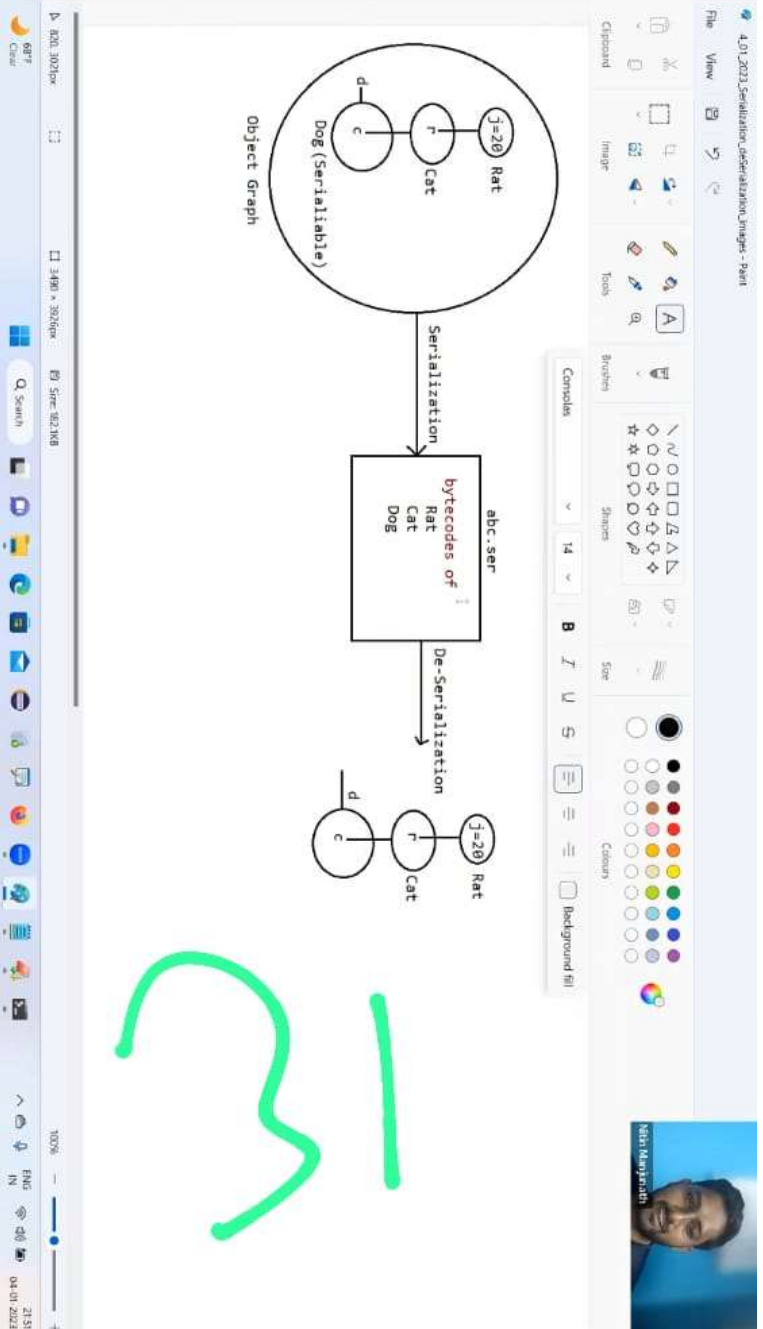
04-07-2023 21:52





```
31 String fileName = "abc.ser";
32 FileOutputStream fos = new FileOutputStream(fileName);
33 ObjectOutputStream oos = new ObjectOutputStream(fos);
34 oos.writeObject(d);
35 System.out.println("Serialization ended.....");
36
37 //To pause the execution till we press some key from keyboard
38 System.in.read();
39
40 System.out.println("De-Serialization started.....");
41 FileInputStream fis = new FileInputStream("abc.ser");
42 ObjectInputStream ois = new ObjectInputStream(fis);
43 Dog d1=(Dog)ois.readObject();
44 System.out.println(d1.c.r.j);
45 System.out.println("De-Serialization ended.....");
46
47 }
48 //JVM shutdown now
49 }
50
51
```

IN 19 607 2 31 00 PC ANS 2155 04-07-2023



```
10 */
11
12 class Account implements Serializable
13 {
14     String userName = "sachin";
15     transient String password = "tendulkar"; // loss of information
16 }
17
18 class Test
19 {
20     public static void main(String[] args) throws Exception
21     {
22         Account account = new Account();
23
24         System.out.println("Serialization started.....");
25         String fileName = "abc.ser";
26         FileOutputStream fos = new FileOutputStream(fileName);
27         ObjectOutputStream oos = new ObjectOutputStream(fos);
28         oos.writeObject(account);
29         System.out.println("Serialization ended.....");
30
31         //To run the execution till we move from keyboard
```

3



```
22 Account account = new Account();
23
24
25 System.out.println("Serialization started.....");
26 String fileName = "abc.ser";
27 FileOutputStream fos = new FileOutputStream(fileName);
28 ObjectOutputStream oos = new ObjectOutputStream(fos);
29 oos.writeObject(account);
30 System.out.println("Serialization ended.....");
31
32 //To pause the execution till we press some key from keyboard
33 System.in.read();
34
35 System.out.println("De-Serialization started.....");
36 FileInputStream fis = new FileInputStream("abc.ser");
37 ObjectInputStream ois = new ObjectInputStream(fis);
38 Account acc=(Account)ois.readObject();
39
40 System.out.println("Username is :: "+acc.userName);
41 System.out.println("Password is :: "+acc.password);
42 System.out.println("De-Serialization ended.....");
```



4/1/2023,Serialization,Serialization,Images - Paint

FileView

Clipboard

Image

Tools

Brushes

Shapes

Size

Colors

Object Graph

Console

14

B

I

U

S

Background fill

Serialization

=====

class Account implements Serializable

{

String username = "sachin";

transient String password = "tendulkar";

}

CustomSerialization

=====

Serialization

=====

username: sachin

password: tendulkar

Serialization

=====

abc.ser

=====

username: sachin

password: null

Deserialization

=====

username: sachin

password: null

34

69°F

Cloud

1440 x 4020px

File Src: 244KB

Q Search

Taskbar

100%

FM5

04-01-2023

Nitin Dhanrajani

4.01.2023,Serialization,deSerialization,Images - Paint

FileView

Clipboard

Image

Tools

Brushes

Shapes

Size

Colors

69°F

Cloud

Search

File Explorer

Edge

Mail

Calendar

Photos

Music

Settings

Control Panel

Power

Taskbar

System Tray

69°F

Cloud

Search

File Explorer

Edge

Mail

Calendar

Photos

Music

Settings

Control Panel

Power

Taskbar

System Tray

100%

22:08

04-07-2023

String username = "sachin";
transient String password = "tendulkar";
}

CustomSerialization
=====

Actual Object

username = sachin
password = tendulkar

Serialization

abc.ser

username:sachin
password:null
tendulkar

Deserialization

username: sachin
password: null

Deserialized Object

password = tendulkar

Serialization

username:sachin
password:null

Deserialization

username: sachin
password: null

3

Nitin Marjoram

4.01.2023,Serialization,DeSerialization,Images - Paint

File View

Clipboard Image Tools Brushes Shapes Size Colors

username = sachin
password = tendulkar

Serialization

123tendulkar

DeSerialization

password: 123

Serialization

=====

1. Default Serialization should happen (password = null, username=sachin)
2. write the encrypted password data as shown below
encryp = "123"+password;
3. Now write the encrypted password also to the serialized Object

De-Serialization

=====

1. Default De-Serialization should happen (password = null, username=sachin)
2. Read encrypted password and decrypt the encrypted password
3. Attach it to password variable with decrypted value.

100%

697 697

Search

Taskbar icons

100% 697 697

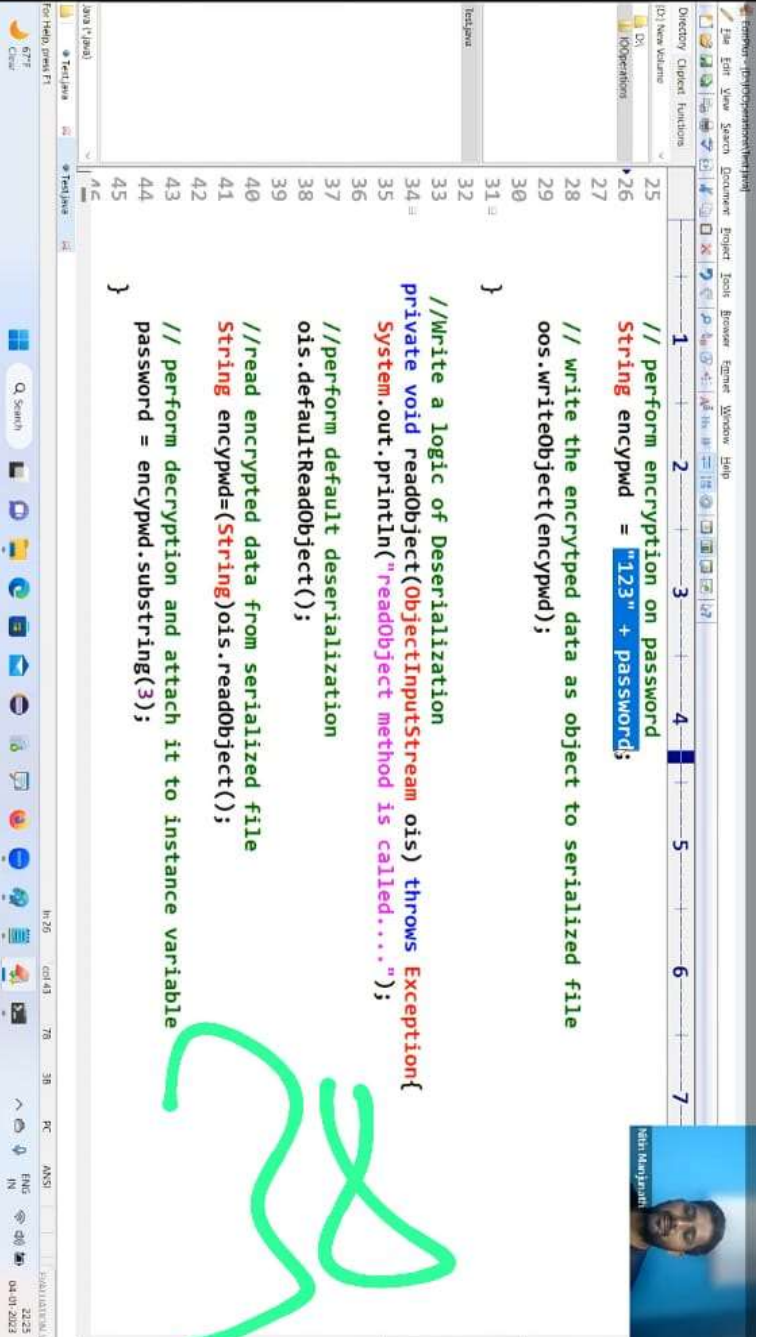
22:23 04-01-2023

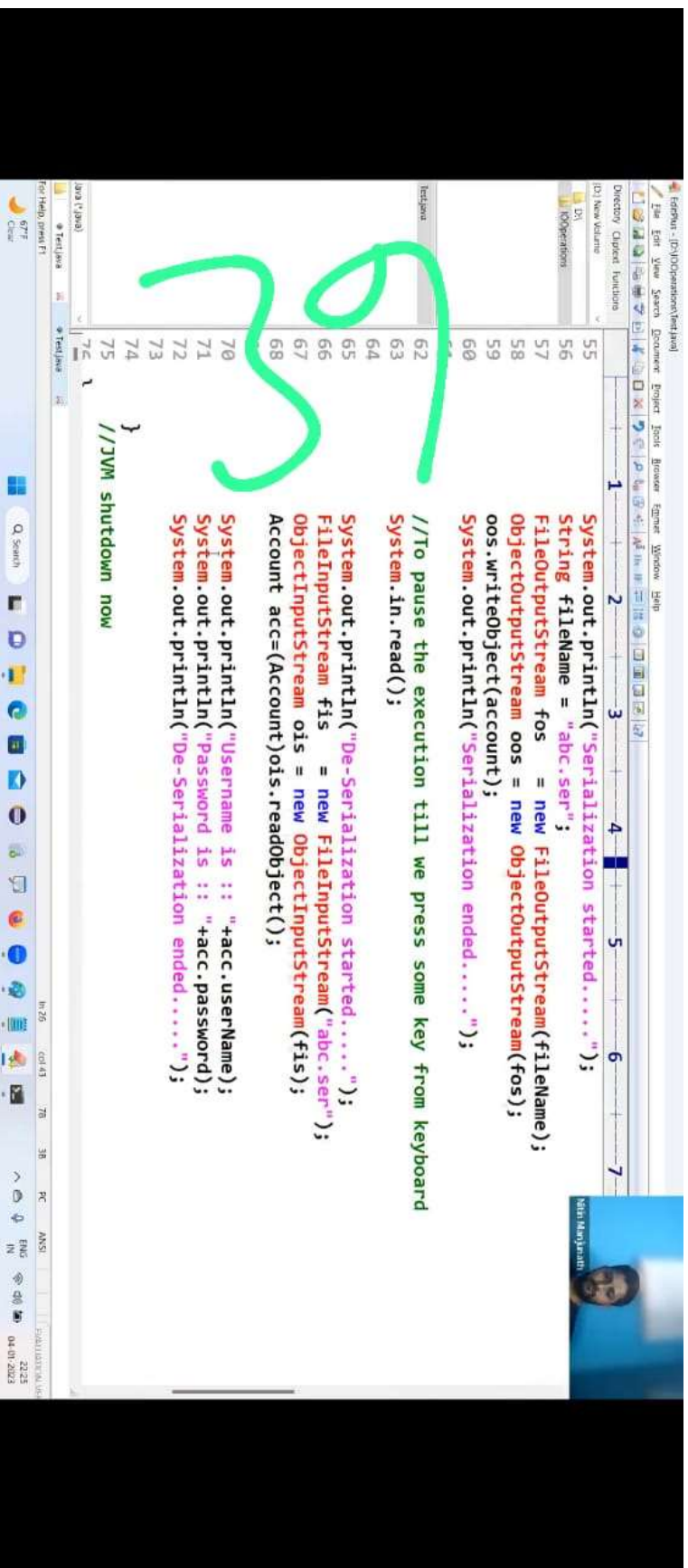
Nar Marjan

```
13 class Account implements Serializable
14 {
15     String userName = "sachin";
16     transient String password = "tendulkar"; // loss of information
17
18     // Write a logic of Serialization
19     private void writeObject(ObjectOutputStream oos) throws Exception{
20         System.out.println("writeObject method is called....");
21
22         // perform default serialization
23         oos.defaultWriteObject();
24
25         // perform encryption on password
26         String encypwd = "123" + password;
27
28         // write the encrypted data as object to serialized file
29         oos.writeObject(encypwd);
30
31     }
32
33     // Write a logic of Deserialization
34     private void readObject(ObjectInputStream ois) throws ClassNotFoundException, IOException{
35
36     }
37 }
```

Handwritten green mark resembling a stylized 'S' or a large checkmark.







=> We can recover this loss of information by using customized serialization.

We can implements customized serialization by using the following two methods.

1. private void writeObject(ObjectOutputStream os) throws Exception.

=> This method will be executed automatically by jvm at the time of serialization.

=> It is a callback method. Hence at the time of serialization if we want to perform any extra work we have to define that in the method only. (prepare encrypted password and write encrypted password separate to the file)

2. private void readObject(ObjectInputStream is) throws Exception.

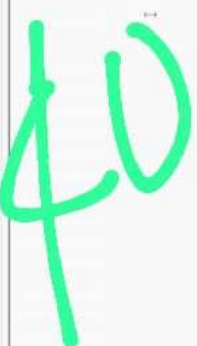
=> This method will be executed automatically by JVM at the time of Deserialization.

Hence at the time of Deserialization if we want to perform any extra activity we have to define that in this method only. (read encrypted password , perform decryption and assign decrypted password to the current object password variable)

eg#1.

```
import java.io.Serializable;  
import java.io.FileOutputStream;  
import java.io.ObjectOutputStream;  
import java.io.FileInputStream;
```

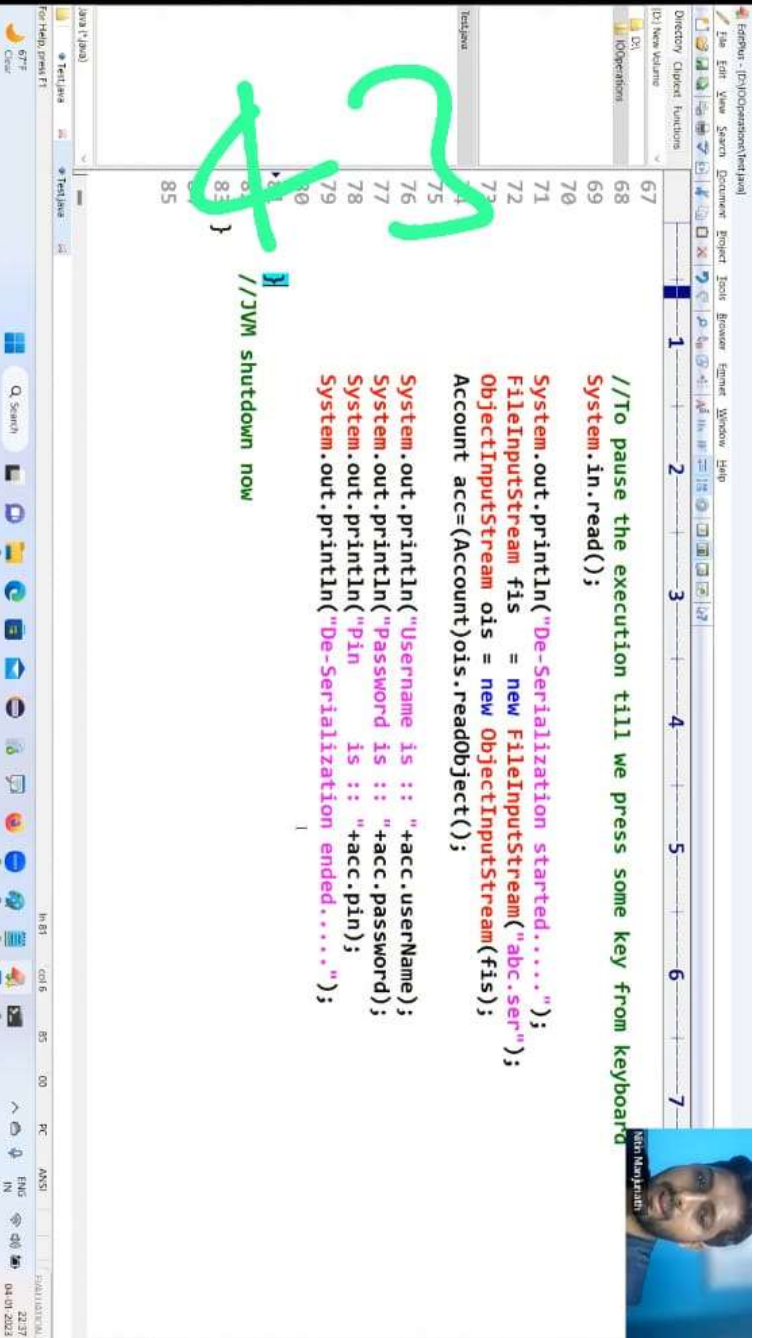
1




```
Editor - (D:\Operations\test.java)
File Edit View Search Runway Project Tools Browser Editor Window Help
(D:) New Volume
Directory Content Functions
DS
IOOperations
test.java
16 transient String password = "tendulkar";//loss of information
17 transient int pin=4444;//loss of information
18
19
20
21 //Write a logic of Serialization
22 private void writeObject(ObjectOutputStream oos) throws Exception{
23     System.out.println("writeObject method is called...");
24
25     // perform default serialization
26     oos.defaultWriteObject();
27
28     // perform encryption on password
29     String encypwd = "123" + password;
30     int encypin = 1111 + pin;
31
32     // write the encrypted data as object to serialized file
33     oos.writeObject(encypwd);
34     oos.writeInt(encypin);
35 }
36
37
38 //Unit test of Serialization
39
```







```
67 //To pause the execution till we press some key from keyboard
68 System.in.read();
69
70
71 System.out.println("De-Serialization started.....");
72 FileInputStream fis = new FileInputStream("abc.ser");
73 ObjectInputStream ois = new ObjectInputStream(fis);
74 Account acc=(Account)ois.readObject();
75
76 System.out.println("Username is :: "+acc.userName);
77 System.out.println("Password is :: "+acc.password);
78 System.out.println("Pin is :: "+acc.pin);
79 System.out.println("De-Serialization ended.....");
80
81
82 //JVM shutdown now
83 }
84
85
```

IDE Interface Details:

- File Explorer: D:\New Volume
- Project Explorer: Test.java
- Run Console: 1, 2, 3, 4, 5, 6, 7
- Output Console: (Empty)
- Taskbar: Windows Start button, Search, Task View, File Explorer, Edge, VS Code, and other applications.
- System Tray: Network, Volume, and Date/Time (04-01-2023, 22:37).

Command Prompt

X + V

D:\IOoperations>javac Test.java

D:\IOoperations>java Test

Serialization started.....

writeObject method is called....

Serialization ended.....

De-Serialization started.....

readObject method is called....

Username is :: sachin

Password is :: tendulkar

Pin is :: 4444

De-Serialization ended.....

D:\IOoperations>|

44



677
Cell

Search



ENG IN 22:38 04-01-2023