ΑΡΧΕΙΑ ΚΑΙ ΧΕΙΡΙΣΜΟΣ ΣΦΑΛΜΑΤΩΝ

Δημήτρης Κ. Ιακωβίδης

Στόχος αυτής της ενότητας είναι η γνωριμία με τη διαχείριση, την αποθήκευση και ανάγνωση αρχείων, όπως επίσης και η εισαγωγή στο χειρισμό σφαλμάτων που προκαλούν εξαιρέσεις κατά το χρόνο εκτέλεσης ενός προγράμματος.

13.1 Διαχείριση αρχείων

Η Java μας προσφέρει δυνατότητες διαχείρισης αρχείων, αντίστοιχες με αυτές που μας δίνονται μέσω του λειτουργικού μας συστήματος, όπως π.χ. μας επιτρέπει η εξερεύνηση των Windows (Windows Explorer). Δηλαδή, μπορούμε να διαγράφουμε, να μετονομάζουμε αρχεία, να διαβάζουμε τα περιεχόμενα φακέλων κοκ.

Όλες αυτές οι δυνατότητες προσφέρονται μέσω μεθόδων της κλάσης File. Η File όπως και όλες οι κλάσεις που αφορούν αρχεία βρίσκονται στο πακέτο java.io. Παραδείγματα χρήσης των μεθόδων της File εικονίζονται στο Σχ.1. Μπορείτε να αναζητήσετε τις μεθόδους που παρουσιάζονται στο Σχ.1 αλλά και τις άλλες μεθόδους της κλάσης File στο Application Programming Interface (API) της Java.

```
1    import java.io.*;
2
3    class FileManagement
4    {
5         public static void main(String[] args)
6         {
7              // Delete test1.txt
8              File f1 = new File("test1.txt");
9              boolean b1 = f1.delete();
10
11             // Rename test2.txt to test1.txt
12             File f2 = new File("test2.txt");
13             boolean b2 = f2.renameTo(f1);
14
15             // Print the contents of the current folder (.)
16             File dir = new File(".");
17
18             // Check if dir is a directory
19             if (dir.isDirectory())
20             {
21                  // Get the filenames in the folder
22                  String[] contents = dir.list();
23
24                  for (int i = 0; i < contents.length; i++)
25                  {
```

```
26                              System.out.println(contents[i]);
27                          }
28                      }
29                  }
30              }
```

Σχήμα 1. Επίδειξη χρήσης αντιπροσωπευτικών μεθόδων της κλάσης File, για διαγραφή (delete), μετονομασία (rename) αρχείων, και προβολή των περιεχομένων ενός φακέλου (directory) – αφού πρώτα ελεγχθεί αν είναι όντως φάκελος. Για να εκτελεσθεί σωστά, τα αρχεία test1.txt και test2.txt θα πρέπει να υπάρχουν στο φάκελο που εργάζεστε.

13.2 Εξαιρέσεις

Στη Java (αλλά και σε άλλες γλώσσες προγραμματισμού) δίνεται η δυνατότητα διαχείρισης των σφαλμάτων που μπορεί να προκύψουν κατά την εκτέλεσης ενός προγράμματος (error handling). Τα σφάλματα αυτά, ως επί το πλείστον χαρακτηρίζονται ως εξαιρέσεις και μπορούμε να προλάβουμε, ώστε να μη διακοπεί ανεξέλεγκτα η ομαλή ροή του προγράμματός μας.

Για το σκοπό αυτό «δοκιμάζουμε» (**try**) αν ένα μπλοκ κώδικα που μας ενδιαφέρει «πετάει» (**throws**) εξαίρεση (**exception**). Αν «πετάξει» εξαίρεση την «πιάνουμε» (**catch**) ώστε να εκτελεσθεί ένας εναλλακτικός κώδικας που έχουμε προβλέψει, αντί του κώδικα που «πετάει» την εξαίρεση. Ο τύπος της εξαίρεσης που μπορεί να «πεταχτεί» περιγράφεται από μια κλάση, π.χ., η κλάση **IOException** αναπαριστά μια εξαίρεση που οφείλεται σε πρόβλημα στην είσοδο/έξοδο (Input/Output, IO) του προγράμματος. Η κλάση αυτή, όπως και άλλες κλάσεις που αναπαριστούν εξαιρέσεις, είναι υποκλάσεις της γενικότερης (υπερ-) κλάσης **Exception**.

Ένα χαρακτηριστικό της Java είναι ότι μας επιβάλει να διαχειριζόμαστε τα σφάλματα αυτού του τύπου στην περίπτωση που μια μέθοδος υπάρχει περίπτωση να «πετάξει» εξαίρεση. Αν μια μέθοδος «πετάει» εξαίρεση ή όχι μπορούμε να το διαπιστώσουμε αν εξερευνήσουμε το API της Java, όπου οι μέθοδοι αυτές αναφέρουν στη δήλωσή τους ότι πετούν "throws" ένα είδος εξαίρεσης. Οι λεπτομέρειες για το μηχανισμό αυτό περιγράφονται αναλυτικότερα σε επόμενη ενότητα.

Παραδείγματα του μηχανισμού διαχείρισης εξαιρέσεων με μπλοκ try-catch μελετώνται παρακάτω στα πλαίσια των αρχείων. Στα παραδείγματα αυτά οι πληροφορίες των σφαλμάτων που συμβαίνουν αποθηκεύονται στο **αντικείμενο e** που υποδεικνύεται στο μπλοκ "catch", π.χ. **catch(IOException e)**. Για να διαβάσουμε το αντίστοιχο μήνυμα σφάλματος χρησιμοποιούμε τη μέθοδο **getMessage()** της αντίστοιχης κλάσης εξαίρεσης.

13.3 Αρχεία κειμένου

Αρχεία κειμένου είναι όσα μπορούμε να ανοίξουμε με έναν απλό κειμενογράφο, όπως είναι το σημειωματάριο (notepad), και το περιεχόμενό τους είναι κατανοητό από τον άνθρωπο. Στο Σχ.2 παρουσιάζεται παράδειγμα εγγραφής σε αρχείο κειμένου, ενώ στο Σχ.3 παρουσιάζεται παράδειγμα ανάγνωσης ενός (ολόκληρου) αρχείου κειμένου.

Ένα αρχείο κειμένου μπορούμε να το διαβάσουμε γραμμή-γραμμή με τη μέθοδο readLine, έως ότου επιστρέψει το κενό (null) αλφαριθμητικό.

Οι κλάσεις BufferedWriter/BufferedReader χρησιμοποιούνται για πιο αποδοτική (ταχύτερη) εγγραφή/ανάγνωση κειμένου από το δίσκο.

```
1    import java.io.*;
2
3    class WriteTextFile
4    {
5          public static void main(String[] args)
6          {
7                try
8                {
9                      FileWriter f =
10                           new FileWriter("myTextFile.txt");
11                     BufferedWriter b = new BufferedWriter(f);
12                     b.write("This file was written by
13                           my class.");
14                     b.close();
15
16               } catch(IOException e)
17               {
18                     System.out.println(e.getMessage());
19               }
20         }
21   }
```

Σχήμα 2. Εγγραφή αλφαριθμητικού σε αρχείο κειμένου.

```
1    import java.io.*;
2
3    class ReadTextFiles
4    {
5          public static void main(String[] args)
6          {
7                try
8                {
9                      FileReader f =
10                           new FileReader("myTextFile.txt");
11                     BufferedReader b = new BufferedReader(f);
12                     String s = "";
13                     while(s != null)
14                     {
15                           s = b.readLine();
16                           System.out.println(s);
17                     }
18                     b.close();
19
20               } catch(IOException e)
21               {
22                     System.out.println(e.getMessage());
23               }
24         }
25   }
```

Σχήμα 3. Ανάγνωση αρχείου κειμένου.

13.4 Δυαδικά αρχεία

Αν κάποιος ανοίξει ένα δυαδικό (binary) αρχείο, π.χ., μια εικόνα jpg, με κειμενογράφο, δε θα καταλάβει το περιεχόμενό του. Τα αρχεία αυτά χρησιμοποιούνται συνήθως για την αποδοτικότερη εγγραφή και ανάγνωση δεδομένων που δεν είναι (μόνο) κείμενο. Για να μπορέσει κάποιος να διαβάσει ένα δυαδικό αρχείο θα πρέπει να γνωρίζει εκ των προτέρων τη δομή/μορφή (format) του, π.χ. στην αρχή έχει 2 ακέραιους, μετά έχει έναν πραγματικό αριθμό διπλής ακρίβειας και στο τέλος έχει έναν χαρακτήρα.

13.4.1 Δυαδικά αρχεία που περιέχουν μόνο bytes δεδομένων

Στο Σχ.4 παρουσιάζεται παράδειγμα εγγραφής και ανάγνωσης ενός μόνο byte σε δυαδικό αρχείο. Ένα byte είναι ένας αριθμός από το 0 έως το 255 (για την αναπαράσταση αυτού το αριθμού χρησιμοποιείται ο τύπος int, γιατί ο πρωταρχικός

τύπος byte που προσφέρει η Java αναφέρεται σε προσημασμένους αριθμούς από το -128 έως το 127).

Ένα αρχείο δεδομένων που περιέχει μόνο bytes μπορούμε να το διαβάσουμε byte-byte με τη μέθοδο read, έως ότου επιστρέψει τον ακέραιο -1, ο οποίος σηματοδοτεί το τέλος του αρχείου.

Για το λόγο αυτό το -1 ονομάζεται **τιμή φρουρός** (sentinel value).

Οι κλάσεις BufferedOutputStream/BufferedInputStream χρησιμοποιούνται για πιο αποδοτική (ταχύτερη) εγγραφή/ανάγνωση δεδομένων από το δίσκο.

```
1    import java.io.*;
2
3    class ByteBinaryFiles
4    {
5          public static void main(String[] args)
6          {
7                // Write a byte to disk
8                try
9                {
10                     FileOutputStream f =
11                       new FileOutputStream("myTextFile.dat");
12                     BufferedOutputStream b =
13                       new BufferedOutputStream(f);
14                     int data = 20;
15                     b.write(data);
16                     b.close();
17
18               } catch(IOException e)
19               {
20                     System.out.println(e.getMessage());
21               }
22
23               // Read the byte from disk
24               try
25               {
26                     FileInputStream f =
27                       new FileInputStream("myTextFile.dat");
28                     BufferedInputStream b =
29                       new BufferedInputStream(f);
30                     int data = b.read();
31                     System.out.println(data);
32                     b.close();
33
34               } catch(IOException e)
35               {
36                     System.out.println(e.getMessage());
37               }
38         }
39   }
```

Σχήμα 4. Εγγραφή και ανάγνωση ενός byte σε δυαδικό αρχείο.

13.4.2 Δυαδικά αρχεία που περιέχουν όχι μόνο bytes δεδομένων

Σε ένα δυαδικό αρχείο γενικά μπορούμε να αποθηκεύουμε όχι μόνο bytes αλλά και άλλους τύπους δεδομένων, όπως int, float, double κλπ. Η διαδικασία είναι παρόμοια με τη διαδικασία που περιγράφηκε στην προηγούμενη ενότητα, με τη διαφορά ότι χρησιμοποιούνται επιπλέον οι κλάσεις DataOutputStream/ DataInputStream αντίστοιχα, οι οποίες έχουν κατάλληλες μεθόδους, π.χ. writeInt/readInt και writeDouble/readDouble για την εγγραφή/ανάγνωση ακεραίων και πραγματικών αριθμών διπλής ακρίβειας, αντίστοιχα. Πληροφορίες για περισσότερες μεθόδους που υποστηρίζουν αυτές οι κλάσεις μπορείτε να βρείτε στο API της Java.

Το τέλος ενός αρχείου δεδομένων που μπορεί να μην περιέχει μόνο bytes, μπορούμε να το εντοπίσουμε μέσω της εξαίρεσης (EOFException) που θα προκληθεί όταν φτάσουμε σε αυτό.

Στο παράδειγμα του Σχ.5, η ανάγνωση του δυαδικού αρχείου γίνεται ατέρμονα έως ότου προκληθεί εξαίρεση EOFException, οπότε και κλείνει το αρχείο.

```
1    import java.io.*;
2    import java.util.Random;
3
4    class BinaryFilesNotOnlyByte
5    {
6          public static void main(String[] args)
7          {
8                Random generator = new Random();
9
10               // Write a set of random numbers to disk
11               try
12               {
13                     FileOutputStream f =
14                           new FileOutputStream("myFile.dat");
15                     BufferedOutputStream b =
16                           new BufferedOutputStream(f);
17                     DataOutputStream d =
18                           new DataOutputStream(b);
19
20                     for (int i = 0; i < 100; i++)
21                     {
22                           double dataDouble =
23                                 generator.nextDouble();
```

```
24                              d.writeDouble(dataDouble);
25                        }
26                        d.close();
27
28                  } catch(IOException e)
29                  {
30                        System.out.println(e.getMessage());
31                  }
32
33                  // Read the saved set of numbers from disk
34                  try
35                  {
36                        FileInputStream f =
37                              new FileInputStream("myFile.dat");
38                        BufferedInputStream b =
39                              new BufferedInputStream(f);
40                        DataInputStream d =
41                              new DataInputStream(b);
42
43                        // Nested try-catch detecting end of file
44                        try
45                        {
46                              int c = 1;
47                              // Infinite loop
48                              for (;;)
49                              {
50                                    double dataDouble;
51                                    dataDouble = d.readDouble();
52                                    System.out.println(c + " " +
53                                          dataDouble);
54                                    c++;
55                              }
56                        } catch (EOFException eof)
57                        {
58                              d.close();
59                        }
60
61                  } catch(IOException e)
62                  {
63                        System.out.println(e.getMessage());
64                  }
65            }
66      }
```

Σχήμα 5. Εγγραφή και ανάγνωση τυχαίων αριθμών τύπου double σε δυαδικό αρχείο.

13.5 Κατασκευή εξαιρέσεων

Σε προηγούμενη ενότητα μελετήθηκε η χρήση του **μπλοκ try-catch** για τη σύλληψη εξαιρέσεων (exceptions) στο χρόνο εκτέλεσης ενός προγράμματος. Για το χειρισμό των σφαλμάτων στα προγράμματά μας, η Java μας δίνει τη δυνατότητα να κατασκευάζουμε δικές μας εξαιρέσεις εξαιρέσεων. Αυτό μπορεί να γίνει απλά,

επεκτείνοντας την κλάση Exception ή οποιαδήποτε υποκλάση της (ανάλογα με το είδος της εξαίρεσης που επιθυμούμε να κατασκευάσουμε), π.χ., την κλάση **IOException** του πακέτου **java.io**, αν η εξαίρεσή μας αφορά κάποια περίπτωση εισόδου/εξόδου αρχείων.

```
1    public class NegativeNumberException extends Exception
2    {
3        public NegativeNumberException( )
4        {
5            super("Negative Number Exception!");
6        }
7
8        public NegativeNumberException(String message)
9        {
10           super(message);
11       }
12   }
```

Σχήμα 6. Μια δική μας κλάση εξαιρέσεων (η οποία χρησιμοποιείται σε παρακάτω παράδειγμα για να υποδεικνύει την παρουσία ενός αρνητικού αριθμού).

13.6 Χειρισμός εξαιρέσεων

Μπορούμε να προκαλέσουμε μια εξαίρεση σε οποιοδήποτε σημείο ενός προγράμματος που κατασκευάζουμε με την εντολή **throw** («πέταξε»/προκάλεσε).

Αν η **throw** καλείται μέσα σε μέθοδο τότε θα πρέπει στην επικεφαλίδα της συνάρτησης να δηλώνεται ότι η μέθοδος αυτή, **throws** (προκαλεί) εξαιρέσεις.

Για να χρησιμοποιήσουμε μια μέθοδο που προκαλεί εξαιρέσεις πρέπει οπωσδήποτε να την καλέσουμε μέσα από ένα μπλοκ **try-catch**, το οποίο κάνει **δοκιμή (try)** αν ο κώδικας που περιλαμβάνει θα προκαλέσει εξαίρεση, και αν συμβεί θα τη **συλλάβει (catch)**. Διαφορετικά δε μεταγλωττίζεται το πρόγραμμα.

Ένα μπλοκ **try-catch** μοιάζει με ένα μπλοκ **switch-case**. Μπορεί να φιλοξενεί πολλά μπλοκ **catch** για διαφορετικά είδη εξαιρέσεων και ένα μπλοκ **finally** (προαιρετικό). Κάθε μπλοκ **catch** εκτελείται μόνο για ένα συγκεκριμένο είδος εξαίρεσης, ενώ το μπλοκ **finally** εκτελείται πάντα. Ένα χαρακτηριστικό παράδειγμα χειρισμού εξαιρέσεων (συμπεριλαμβανομένης της εξαίρεσης του Σχ.6) εικονίζεται στο Σχ.7. Στο παράδειγμα αυτό αναλόγως με την τιμή που δίνεται ως όρισμα στη μέθοδο **exerciseMethod** εκτελείται και διαφορετική εξαίρεση. Για να εκτελεστεί σωστά

πρέπει να έχει υλοποιηθεί πρώτα η κλάση του Σχ.6, γιατί χρησιμοποιείται στη γραμμή 23 του Σχ.7.

```
1    public class FinallyDemo
2    {
3        public static void main(String[] args)
4        {
5            try
6            {
7                exerciseMethod(-2);
8            }
9            catch(Exception e)
10           {
11               System.out.println("Caught in main.");
12           }
13       }
14
15       public static void exerciseMethod(int n) throws
16                                                    Exception
17       {
18           try
19           {
20               if (n > 0)
21                   throw new Exception( );
22               else if (n < 0)
23                   throw new NegativeNumberException( );
24               else
25                  System.out.println("No Exception.");
26               System.out.println("Still in sampleMethod.");
27           }
28           catch(NegativeNumberException e)
29           {
30               System.out.println("Caught in sampleMethod.");
31                 System.out.println(e.getMessage());
32           }
33           finally
34           {
35               System.out.println("In finally block.");
36           }
37           System.out.println("After finally block.");
38       }
39   }
```

Σχήμα 7. Παράδειγμα χειρισμού εξαιρέσεων.

13.7 Πότε ένα αντικείμενο μπορεί να εγγραφεί σε αρχείο;

Για να μπορεί ένα αντικείμενο μιας κλάσης να εγγραφεί σε αρχείο, **πρέπει να υλοποιεί τη διεπαφή Serializable**. Η διεπαφή αυτή υπάρχει στο πακέτο java.io και ο ρόλος της είναι απλά και μόνο να υποδεικνύει ότι ένα αντικείμενο μπορεί να εγγραφεί σε αρχείο. Για παράδειγμα, αν επιθυμούμε να έχουμε μια κλάση

ημερομηνιών (Date) και να μας επιτρέπεται να γράφουμε τις ημερομηνίες αυτές σε αρχείο, η κλάση των ημερομηνιών **θα πρέπει να υλοποιεί τη διεπαφή Serializable**, όπως εικονίζεται στο Σχ.8.

```
1    import java.io.Serializable;
2
3    public class Date implements Serializable
4    {
5          int day, month, year;
6
7          public Date(int d, int m, int y)
8          {
9                day = d; month = m; year = y;
10         }
11
12         public String toString()
13         {
15               return day + "/" + month + "/" + year;
16         }
17
18   }
```

Σχήμα 8. Μια κλάση ημερομηνιών που μπορεί να εγγραφεί σε αρχείο γιατί υλοποιεί τη διεπαφή Serializable.

13.8 Αρχεία αντικειμένων

Αντικείμενα μπορούν να εγγραφούν σε **δυαδικά αρχεία**, όπως αυτά που αποθηκεύουν διάφορους τύπους μεταβλητών. Για την εγγραφή και την ανάγνωση τέτοιων αρχείων χρησιμοποιούνται οι κλάσεις **ObjectOutputStream** και **ObjectInputStream** αντίστοιχα, οι οποίες ανήκουν στο πακέτο **java.io**. Στο Σχ.9 εικονίζεται ένα παράδειγμα εγγραφής και αποθήκευσης ενός αντικειμένου τύπου Date (όπως ορίσθηκε Σχ.8) στο δίσκο, σε ένα αρχείο με όνομα "TestObjectFile.java".

```
1    import java.io.*;
2
3    public class ObjectFileDemo
4    {
5          public static void main(String[] args)
6          {
7                // File write
8                try
9                {
10                     ObjectOutputStream oos =
11                     new ObjectOutputStream(
```

```
12          new FileOutputStream("TestObjectFile.dat"));
13                      Date d = new Date(3,2,2014);
15                      oos.writeObject(d);
16                      oos.close();
17
18                } catch (IOException e)
19                {
20                      System.out.println(e);
21                }
22
23                // File read
24                try
25                {
26                      ObjectInputStream oos =
27                      new ObjectInputStream(
28          new FileInputStream("TestObjectFile.dat"));
29                      Date d;
30                      d = (Date)oos.readObject();
31                      System.out.println(d);
32                      oos.close();
33
34                } catch (Exception e)
35                {
36                      System.out.println(e);
37                }
38
39          }
40    }
```

Σχήμα 9. Εγγραφή και ανάγνωση ενός αντικειμένου τύπου Date (Σχ.8)

Αν επιθυμούμε να ανιχνεύσουμε αυτόματα το τέλος ενός τέτοιου αρχείου τότε χρησιμοποιούμε ένα μπλοκ **try-catch** που συλλαμβάνει εξαιρέσεις τύπου **EOFException**, με τον ίδιο τρόπο που κάναμε και στα δυαδικά αρχεία οποιουδήποτε τύπου σε προηγούμενη ενότητα.

Ασκήσεις

1. Να κατασκευαστεί ένα πρόγραμμα που θα ζητά από το χρήστη μια διαδρομή φακέλου στο δίσκο. Αφού ο χρήστης γράψει τη διαδρομή στο πληκτρολόγιο και πιέσει το πλήκτρο Enter, θα τυπώνει τα περιεχόμενα του φακέλου στην οθόνη.

2. Να κατασκευαστεί εφαρμογή που διαβάζει ένα αρχείο κειμένου από το δίσκο και το εμφανίζει στην οθόνη. Το όνομα του αρχείου να δίνεται ως όρισμα από τη γραμμή εντολών.

3. Να κατασκευαστεί εφαρμογή που γράφει ένα αρχείο τυχαίων αριθμών τύπου double στο δίσκο. Το πλήθος των αριθμών που αποθηκεύεται θα πρέπει να είναι και αυτό τυχαίο. Στην αρχή του αρχείου θα πρέπει να αποθηκεύεται το πλήθος των αριθμών. Αμέσως μετά, θα πρέπει να αποθηκεύονται οι αριθμοί.

4. Να κατασκευαστεί εφαρμογή που διαβάζει ένα αρχείο αριθμών τύπου double από το δίσκο. Το αρχείο θα πρέπει να έχει τη μορφή που υποδεικνύεται στην προηγούμενη άσκηση (δηλ. πρώτα θα διαβάζεται το πλήθος των αποθηκευμένων αριθμών από το αρχείο και κατόπιν θα διαβάζονται οι αποθηκευμένοι αριθμοί). Οι αριθμοί (τύπου double) που διαβάζονται από το αρχείο θα πρέπει να αποθηκεύονται σε αντίστοιχο πίνακα, ο οποίος στο τέλος του προγράμματος θα πρέπει εκτυπώνεται στην οθόνη.

5. Να υλοποιήσετε το παράδειγμα των Σχ. 6-7.

6. Να τροποποιήσετε το παράδειγμα των Σχ.6-7 ώστε να εισάγει ο χρήστης την τιμή στη μέθοδο exerciseMethod, και να εμφανίζει αν ο αριθμός που εισήγαγε είναι θετικός ή αρνητικός προκαλώντας κατάλληλη εξαίρεση (να κατασκευάσετε μια PositiveNumberException).

7. Να υλοποιήσετε το παράδειγμα των Σχ. 8-9.

8. Να τροποποιήσετε το παράδειγμα των Σχ.8-9 για την αποθήκευση ενός αντικειμένου μιγαδικού αριθμού.

9. Να τροποποιήσετε το παράδειγμα της προηγούμενης άσκησης για την αποθήκευση μιας λίστας (ArrayList) μιγαδικών αριθμών.