Προγραμματισμός ΙΙ

6/5/2020

Αρχεία και χειρισμός σφαλμάτων

Διαχείριση αρχείων

- Η Java δίνει δυνατότητες διαχείρισης αρχείων αντίστοιχες με αυτές που δίνουν τα λειτουργικά συστήματα. Όπως πχ μας επιτρέπει ο Windows Explorer.

- Μπορούμε να διαγράψουμε ή να μετονομάσουμε αρχεία, να διαβάζουμε τα περιεχόμενα φακέλων τους κλπ

- Όλες αυτές οι δυνατότητες προσφέρονται μέσω μεθόδων της κλάσης File (βρίσκεται μέσα στο πακέτο **java.io**).

- Σχετικές μέθοδοι μπορούν να αναζητηθούν στο API της Java.

Παράδειγμα διαχείρισης αρχείων

```
import java.io.*;

class FileManagement
{
        public static void main(String[] args)
        {
                //Delete test1.txt
                File f1=new File("test1.txt");
                boolean b1=f1.delete();

                //Rename test2.txt to test1.txt
                File f2=new File("test2.txt");
                boolean b2=f2.renameTo(f1);
```

Παράδειγμα διαχείρισης αρχείων

```
//Print the contents of the current folder()
File dir=new File(“.”);

//Check if dir is a directory
if (dir.isDirectory())
{
        //Get the filename in the folder
        String[] contents=dir.list();

        for (int i=0;i<contents.length;i++)
        {System.out.println(contents[i]);}
}
}}
```

Εξαιρέσεις (exceptions)

- Στη Java, όπως και σε άλλες γλώσσες, υπάρχει η δυνατότητα διαχείρισης σφαλμάτων που προκύπτουν κατά την εκτέλεση (error handling).

- Τα σφάλματα αυτά ονομάζονται εξαιρέσεις και μπορούμε να τα χειριστούμε ώστε να μη διακοπεί ανεξέλεγκτα η ομαλή ροή του προγράμματος.

- Για τον σκοπό αυτό, δοκιμάζουμε (**try**) αν ένα μπλοκ κώδικα που μας ενδιαφέρει «πετάει» (**throws**) εξαίρεση (**exception**).

- Αν «πετάξει» εξαίρεση, την «πιάνουμε» (**catch**), ώστε να εκτελεστεί εναλλακτικός κώδικας που έχουμε προβλέψει.

Εξαιρέσεις (exceptions)

- Ο τύπος της εξαίρεσης που μπορεί να «πεταχτεί» περιγράφεται από μία κλάση.

- Πχ η κλάση **IOException** αναπαριστά μια εξαίρεση που οφείλεται σε πρόβλημα στην είσοδο/έξοδο του προγράμματος.

- Η κλάση αυτή, όπως και άλλες κλάσεις που αναπαριστούν εξαιρέσεις, είναι υποκλάσεις της **Exception**.

- Η Java επιβάλει τη διαχείριση των εξαιρέσεων.

- Οι μέθοδοι που «πετάνε» εξαίρεση, το δηλώνουν στην επικεφαλίδα (με **throws**).

Αρχεία κειμένου

- Αρχεία κειμένου είναι όσα μπορούμε να ανοίξουμε με έναν απλό κειμενογράφο (πχ notepad) και το περιεχόμενό τους είναι κατανοητό από τον άνθρωπο.

- 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 εγγραφή/ανάγνωση κειμένου από τον δίσκο.

Εγγραφή αλφαριθμητικού σε αρχείο κειμένου

```
import java.io.*;

Class WriteTextFile
{
        public static void main(String[] args)
        {
                try
                {
                        FileWriter f=new FileWriter(“myTextFile.txt”);
                        BufferedWriter b=new BufferedWriter(f);
                        b.write(“This file was written by my class.”);
                        b.close();
                } catch(IOException e)
```

Εγγραφή αλφαριθμητικού σε αρχείο κειμένου

```
            } catch(IOException e)
            {

                  System.out.println(e.getMessage());
            }
      }
}
```

Ανάγνωση αρχείου κειμένου

```
import java.io.*;

Class ReadTextFiles
{       public static void main(String[] args)
        {
                try
                {       Filename f=new FileReader(“myTextFile.txt”);
                        BufferedReader b=new BufferedReader(f);
                        String s=“”;
                        while(s!=null){s=b.readLine();System.out.println(s); }
                        b.close();
                } catch(IOException e)
                {System.out.println(e.getMessage());  }}}
```

Δυαδικά αρχεία

- Αν κάποιος ανοίξει ένα δυαδικό (binary) αρχείο, πχ 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 μπορεί κάποιος να διαβάσει ένα δυαδικό αρχείο, θα πρέπει να γνωρίζει εκ των προτέρων τη μορφή (format) του.

Εγγραφή και ανάγνωση byte σε δυαδικό αρχείο

- Ένα byte είναι ένα αριθμός από 0 έως 255 (2^8-1).
- Για την αναπαράστασή του χρησιμοποιείται ο τύπος int, γιατί ο πρωταρχικός τύπος byte που προσφέρει η Java αναφέρεται σε προσημασμένους αριθμούς από -128 έως 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 (τιμή φρουρός).
- Οι κλάσεις BufferedOutputStream/BufferedInputStream χρησιμοποιούνται για πιο αποδοτική εγγραφή/ανάγνωση.

Εγγραφή και ανάγνωση byte σε δυαδικό αρχείο

```
import java.io.*;

class ByteBinaryFiles
{
        public static void main(String[] args)
        {
                //Write a byte to disk
                try
                {FileOutputStream f=new FileOutputStream("myTextFile.dat");
                 BufferedOutputStream b=new BufferedOutputStream(f);
                 int data=20;
                 b.write(data);
```

Εγγραφή και ανάγνωση byte σε δυαδικό αρχείο

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

 //Read the byte from disk
try
{FileInputStream f=new FileInputStream("myTextFile.dat");
 BufferedInputStream b=new BufferedInputStream(f);
 int data=b.read();
 System.out.println(data);
 b.close();
```

Εγγραφή και ανάγνωση byte σε δυαδικό αρχείο

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

Εγγραφή και ανάγνωση άλλων τύπων δεδομένων σε δυαδικό αρχείο

- Σε ένα δυαδικό αρχείο μπορούν να αποθηκευτούν, πέρα από bytes, και άλλοι τύποι δεδομένων, όπως int, float, double κλπ.
- Η διαδικασία είναι παρόμοια με την προηγούμενη, απλώς χρησιμοποιούνται επιπλέον οι κλάσεις DataOutputStream/ DataInputStream.
- Οι κλάσεις αυτές έχουν κατάλληλες μεθόδους, πχ writeInt/readInt και writeDouble/readDouble για την εγγραφή/ανάγνωση ακεραίων και πραγματικών αριθμών διπλής ακρίβειας, αντίστοιχα.
- Το τέλος ενός τέτοιου αρχείου μπορεί να εντοπιστεί μέσω εξαίρεσης (EOFException).

Εγγραφή και ανάγνωση άλλων τύπων δεδομένων σε δυαδικό αρχείο

```
import java.io.*;
Import java.util.Random;

class BinaryFilesNotOnlyByte
{
        public static void main(String[] args)
        {
                Random generator=new Random();

                //Write a set of random numbers to disk
                try
                {
```

Εγγραφή και ανάγνωση άλλων τύπων δεδομένων σε δυαδικό αρχείο

```
try{
        FileOutputStream f=new FileOutputStream("myFile.dat");
        BufferedOutputStream b=new BufferedOutputStream(f);
        DataOutputStream d=new DataOutputStream(d);

        for (int i=0;i<100;i++)
        {
            double dataDouble=generator.nextDouble();
            d.writeDouble(dataDouble);
        }
        d.close();
    } catch(IOException e)
```

Εγγραφή και ανάγνωση άλλων τύπων δεδομένων σε δυαδικό αρχείο

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

Εγγραφή και ανάγνωση άλλων τύπων δεδομένων σε δυαδικό αρχείο

```
//Read the saved set of numbers from disk
try
{
        FileInputStream f=new FileInputStream("myFile.dat");
        BufferedInputStream b=new BufferedInputStream(f);
        DataInputStream d=new DataInputStream(d);

        //Nested try-catch detecting EOF
        try
        {
                int c=1;
                //Infinite loop
```

Εγγραφή και ανάγνωση άλλων τύπων δεδομένων σε δυαδικό αρχείο

```
        //Infinite loop
        for(;;)
        {
                double dataDouble;
                dataDouble=d.readDouble();
                System.out.println(c+" "+dataDouble);
                c++;
        }
} catch (EOFException eof)
{
        d.close();
}
```

Εγγραφή και ανάγνωση άλλων τύπων δεδομένων σε δυαδικό αρχείο

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

Κατασκευή εξαιρέσεων

- Μελετήθηκε η χρήση του μπλοκ **try-catch** για σύλληψη εξαιρέσεων (exceptions) κατά την εκτέλεση ενός προγράμματος.
- Η Java μας δίνει τη δυνατότητα να κατασκευάζουμε και δικές μας εξαιρέσεις.
- Αυτό γίνεται επεκτείνοντας την κλάση Exception.
- Ή κάποια κατάλληλη υποκλάση της, ανάλογα με το είδος της εξαίρεσης που επιθυμούμε. Πχ την IOException για είσοδο/έξοδο αρχείων.

Κατασκευή εξαιρέσεων

```
public class NegativeNumberException extends Exception
{
        public NegativeNumberException()
        {
                super("Negative Number Exception!");
        }

        public NegativeNumberException(String message)
        {
                super(message);
        }
}
```

Χειρισμός εξαιρέσεων

- Μπορούμε να προκαλέσουμε μια εξαίρεση σε οποιοδήποτε σημείο ενός προγράμματος που κατασκευάζουμε με την εντολή **throw** («πέταξε»).
- Αν η **throw** καλείται μέσα σε μια μέθοδο, τότε θα πρέπει στην επικεφαλίδα της συνάρτησης να δηλώνεται ότι η μέθοδος κάνει **throw**.
- Για να χρησιμοποιήσουμε μια μέθοδο που προκαλεί εξαιρέσεις, πρέπει οπωσδήποτε να την καλέσουμε μέσα από ένα μπλοκ **try-catch**.
- Το μπλοκ κάνει **δοκιμή (try)** αν ο κώδικας που περιλαμβάνει θα προκαλέσει εξαίρεση, και αν συμβεί θα τη συλλάβει (**catch**). Διαφορετικά δε μεταγλωττίζεται το πρόγραμμα.

Χειρισμός εξαιρέσεων

- 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.
- Στο παράδειγμα που ακολουθεί, αναλόγως με την τιμή που δίνεται ως όρισμα στη μέθοδο **exerciseMethod** εκτελείται και διαφορετική εξαίρεση.

Χειρισμός εξαιρέσεων

```
public class FinallyDemo
{
        public static void main(String[] args)
        {
                try
                {exerciseMethod(-2);}
                catch(Exception e)
                {System.out.println(“Caught in main.”);}
        }
```

Χειρισμός εξαιρέσεων

```
public static void exerciseMethod(int n) throws Exception
{
        try
        {
        if (n>0) throw Exception();
        else if (n<0) throw new NegativeNumberException();
        else System.out.println("Still in sampleMethod.");
        }
        catch(NegativeNumberException e)
        {System.out.println("Caught in sampleMethod.");
         System.out.println((e.getMessage());
        }
```

Χειρισμός εξαιρέσεων

```
        finally
        {
                System.out.println("In finally block");
        }
        System.out.println("After finally block");
    }
}
```

Πότε ένα αντικείμενο μπορεί να εγγραφεί σε αρχείο;

- Για να μπορεί ένα αντικείμενο να εγγραφεί σε αρχείο, πρέπει να υλοποιεί τη διεπαφή **Serializable**.
- Η διεπαφή αυτή υπάρχει στο πακέτο **java.io**.
- Στο παράδειγμα που ακολουθεί, έχουμε μια κλάση ημερομηνιών (Date) και επιτρέπεται να γράψουμε τις ημερομηνίες αυτές σε αρχείο.

Πότε ένα αντικείμενο μπορεί να εγγραφεί σε αρχείο;

```
import java.io.Serializable;

public class Date implements Serializable
{
    int day,month,year;

    public Date(int d,int m,int y)
    {day=d; month=m; year=y;}

    public String toString()
    {return day+”/”+month+”/”+year;}
}
```

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**, οι οποίες ανήκουν στο πακέτο **java.io**.

- Ακολουθεί παράδειγμα εγγραφής και αποθήκευσης ενός αντικειμένου τύπου Date.

Αρχεία αντικειμένων

```
import java.io.*

public class ObjectFileDemo
{
    public static void main(String[] args)
    {
            //File write
            try
            {
                    ObjectOutputStream oos=new ObjectOutputStream
                    (new FileOutputStream("TestObjectFile.dat"));
```

Αρχεία αντικειμένων

```
                Date d=new Date(3,2,2014);
                oos.writeObject(d);
                oos.close();
} catch(IOException e)
{System.out.println(e);}

//File read
try
{
        ObjectOutputStream oos=new ObjectOutputStream
        (new FileOutputStream("TestObjectFile.dat"));
        Date d;
```

Αρχεία αντικειμένων

```
            Date d;
            d=(Date)oo.readObject();
            System.out.println(d);
            oos.close();
        } catch(Exception e)
        {
            System.out.println(e);
        }
    }
}
```

Αρχεία αντικειμένων

- 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 έγινε στα δυαδικά αρχεία.