Οντοκεντρικός Προγραμματισμός

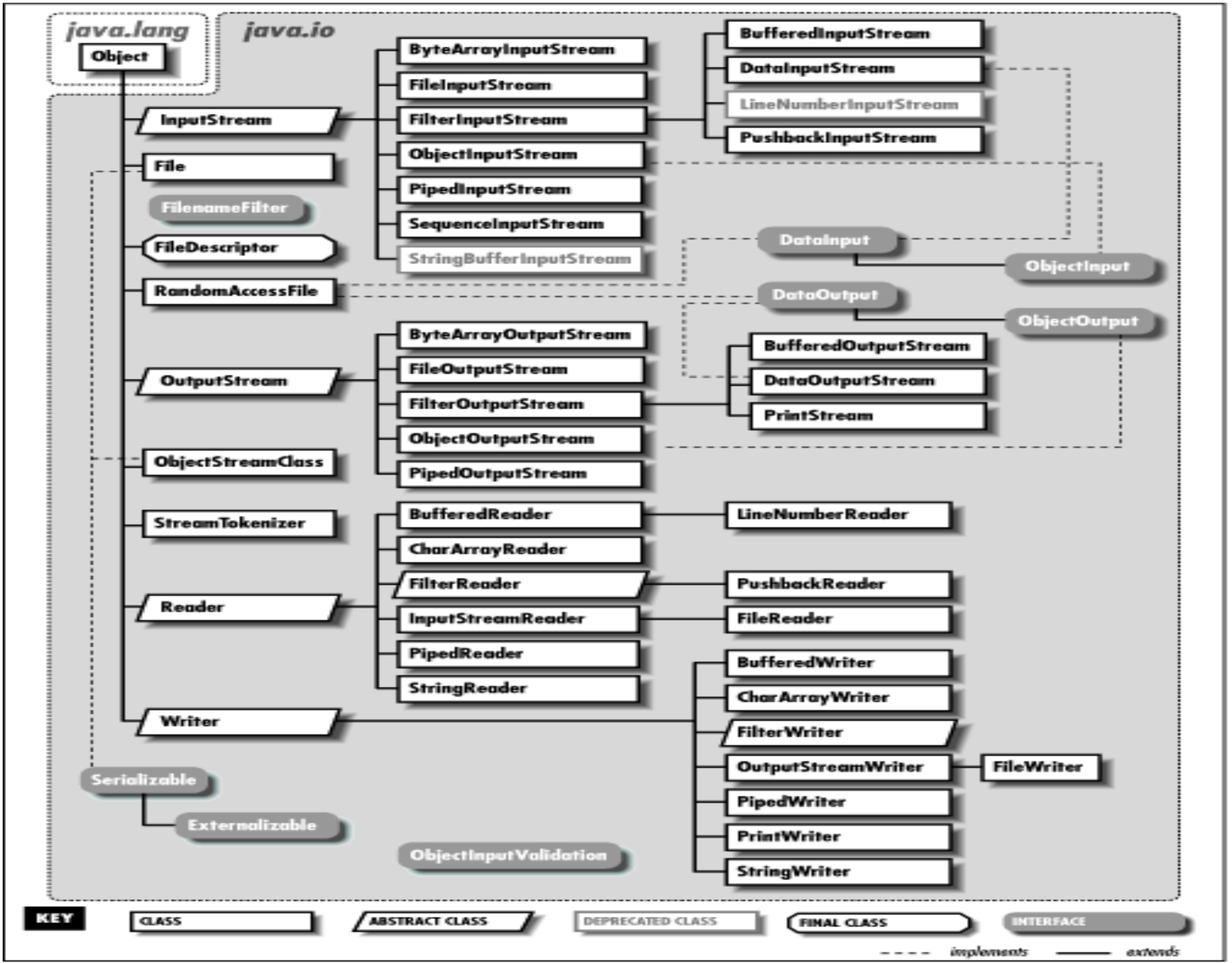
Φροντιστήριο JAVA

Αρχεία

Αρχεία δεδομένων

- Το πακέτο java.io περιλαμβάνει περισσότερες από 60 κλάσεις και διασυνδέσεις για το χειρισμό αρχείων δεδομένων.

- Αρκετές από τις κλάσεις του πακέτου java.io εντάσσονται σε μία από τις δύο κύριες κατηγορίες, οι οποίες διαχειρίζονται
  1. αρχεία κειμένου
  2. δυαδικά αρχεία


java.lang
java.io
Object
InputStream
File
FilenameFilter
FileDescriptor
RandomAccessFile
OutputStream
ObjectStreamClass
StreamTokenizer
Reader
Writer
Serializable
Externalizable
ObjectInputValidation
ByteArrayInputStream
FileInputStream
FilterInputStream
ObjectInputStream
PipedInputStream
SequenceInputStream
StringBufferInputStream
BufferedInputStream
DataInputStream
LineNumberInputStream
PushbackInputStream
DataInput
ObjectInput
DataOutput
ObjectOutput
ByteArrayOutputStream
FileOutputStream
FilterOutputStream
ObjectOutputStream
PipedOutputStream
BufferedOutputStream
DataOutputStream
PrintStream
BufferedReader
CharArrayReader
FilterReader
InputStreamReader
PipedReader
StringReader
LineNumberReader
PushbackReader
FileReader
BufferedWriter
CharArrayWriter
FilterWriter
OutputStreamWriter
PipedWriter
PrintWriter
StringWriter
FileWriter
KEY
CLASS
ABSTRACT CLASS
DEPRECATED CLASS
FINAL CLASS
INTERFACE
implements
extends

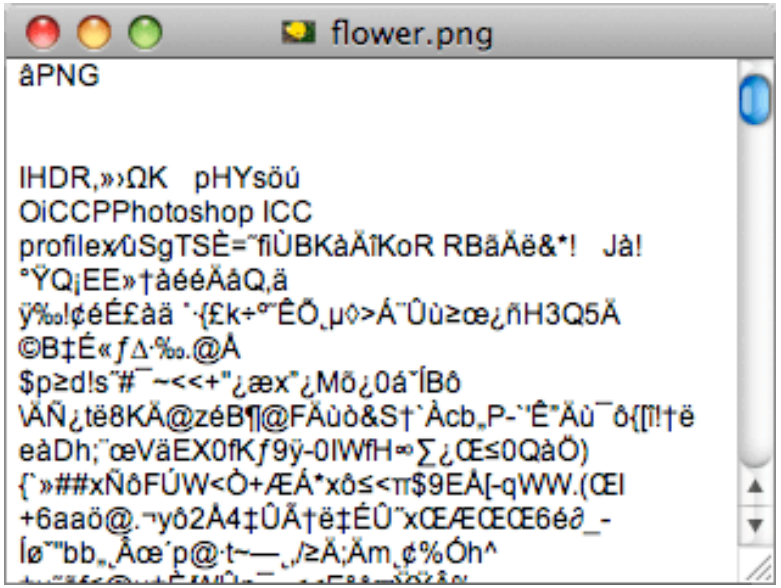
Αρχεία κειμένου

- Όταν μεταφέρουμε δεδομένα από ένα πρόγραμμα σε ένα αρχείο κειμένου, τότε όλα τα δεδομένα μετατρέπονται σε κείμενο πριν από την αποθήκευση.
  - Στην ουσία, τα αρχεία κειμένου περιλαμβάνουν δεδομένα σε μορφή παρόμοια με τον τύπο **char** της Java — συνήθως απλές, οργανωμένες σε γραμμές, αλφαριθμητικές πληροφορίες τις οποίες μπορεί να διαβάσει ο χρήστης χρησιμοποιώντας ένα συντάκτη κειμένου.
- Τα αρχεία κειμένου μπορούν να διαβαστούν και από προγράμματα που έχουν γραφεί σε άλλες γλώσσες προγραμματισμού, εκτός από την Java.

Δυαδικά αρχεία

- Όταν μεταφέρουμε δεδομένα από ένα πρόγραμμα σε ένα δυαδικό αρχείο, τότε τα δεδομένα δεν υφίστανται καμία μετατροπή πριν από την αποθήκευση.
- Τα δυαδικά αρχεία ποικίλουν περισσότερο: συνηθισμένο παράδειγμα είναι τα αρχεία εικόνων, και τα εκτελέσιμα προγράμματα.
- Σε ένα δυαδικό αρχείο μπορούμε να αποθηκεύσουμε με μία λειτουργία εγγραφής ακόμα και ένα ολόκληρο αντικείμενο ή συλλογή αντικειμένων, διαδικασία γνωστή ως σειριοποίηση.
- Ένα δυαδικό αρχείο μπορεί να διαβαστεί μόνο από προγράμματα όπου τα δεδομένα αναπαρίστανται εσωτερικά με τον ίδιο τρόπο όπως στο πρόγραμμα που δημιούργησε το συγκεκριμένο αρχείο.

Δυαδικά αρχεία

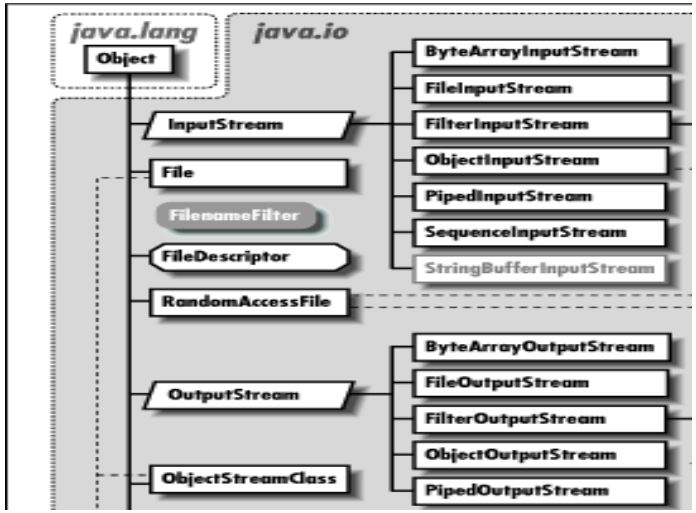


Αρχεία δεδομένων

- Οι κλάσεις που αναφέρονται στη διαχείριση αρχείων κειμένου ονομάζονται αναγνώστες (readers) και γραφείς (writers).
- Οι κλάσεις που αναφέρονται στη διαχείριση δυαδικών αρχείων είναι γνωστές ως χειριστές ρευμάτων (stream).
- Σε αρκετές περιπτώσεις, ωστόσο, χρησιμοποιείται ο όρος **ρεύμα δεδομένων** είτε αναφερόμαστε **σε είσοδο/έξοδο από/σε δυαδικά αρχεία είτε σε αρχεία κειμένου**. Στη δεύτερη περίπτωση αναφερόμαστε **σε ρεύματα κειμένου**.

Αντικείμενα αρχείων – η κλάση File

- Για τη δημιουργία αντικειμένων αρχείων χρησιμοποιούμε την κλάση File.
- Τα αντικείμενα File χρησιμοποιούνται για να πάρουμε πληροφορίες για φακέλους ή αρχεία.
- Τα αντικείμενα File δεν χρησιμοποιούνται για την ανάγνωση ή εγγραφή σε φακέλους ή αρχεία. Γι' αυτό το λόγο χρησιμοποιούνται τα ρεύματα εισόδου/εξόδου.



Αντικείμενα αρχείων – η κλάση File

Μέθοδος	Περιγραφή
`exists()`	Επιστρέφει **`true`** αν υπάρχει το αντικείμενο αρχείου ή φακέλου, και **`false`** διαφορετικά
`isDirectory()`	Επιστρέφει **`true`** αν το αντικείμενο αναφέρεται σε φάκελο, και **`false`** διαφορετικά
`isFile()`	Επιστρέφει **`true`** αν το αντικείμενο αναφέρεται σε αρχείο, και **`false`** διαφορετικά
`isAbsolute()`	Επιστρέφει **`true`** αν το αντικείμενο αναφέρεται σε μια πλήρη διαδρομή, και **`false`** διαφορετικά
`canRead()`	Επιστρέφει **`true`** αν μπορούμε να διαβάσουμε από το αρχείο, και **`false`** διαφορετικά
`canWrite()`	Επιστρέφει **`true`** αν μπορούμε να γράψουμε στο αρχείο, και **`false`** διαφορετικά
`getName()`	Επιστρέφει ένα αλφαριθμητικό, το οποίο αντιπροσωπεύει το όνομα του αρχείου ή το όνομα του φακέλου ανάλογα, χωρίς να αναφέρεται η διαδρομή
`getPath()`	Επιστρέφει ένα αλφαριθμητικό με το όνομα της διαδρομής
`length()`	Επιστρέφει έναν ακέραιο τύπου long που εκφράζει το μήκος του αρχείου σε **`bytes`**

Η κλάση File

Για τη δημιουργία ενός αντικειμένου File:

- Δημιουργούμε πρώτα ένα αντικείμενο που αντιστοιχεί στη διαδρομή του αρχείου (path):

```
File myDir = new File("C:\\projects\\files");
```

Το "\\" είναι ο χαρακτήρας διαφυγής "\" μαζί με το διαχωριστικό διαδρομής "\"

- Στη συνέχεια, δημιουργούμε ένα αντικείμενο αρχείου που αντιστοιχεί στο ίδιο το αρχείο data.txt:

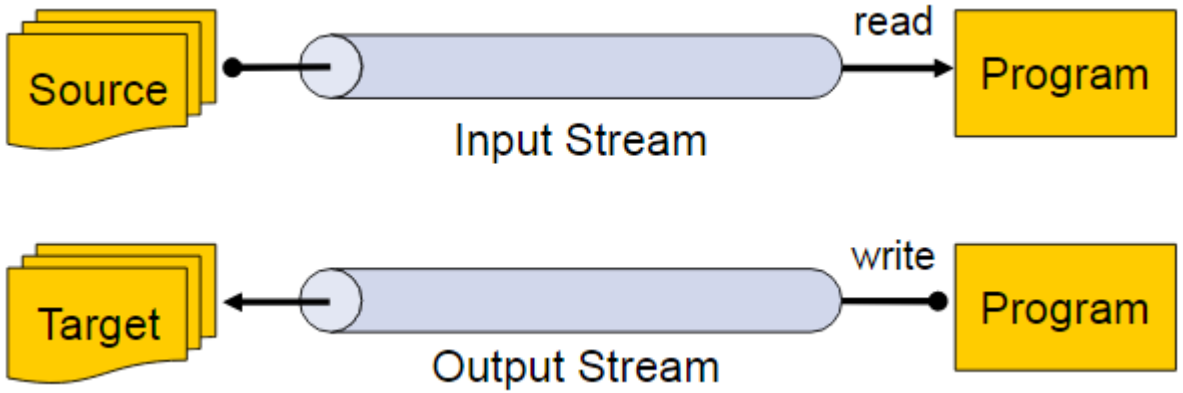
```
File myFile = new File(myDir, "data.txt");
```

- Μπορούμε να συνδυάσουμε τις δύο εργασίες ως εξής:

```
File myFile = new File("C:\\projects\\files", "data.txt");
```

Ρεύματα δεδομένων 1

- Τα δεδομένα ανταλλάσσονται μεταξύ του προγράμματος και ενός αρχείου ως ένα ρεύμα.
- Τα δεδομένα ρέουν από μια προέλευση (source) προς ένα προορισμό (destination).
- Αρχικά, «ανοίγουμε» το ρεύμα.
- Αν το ρεύμα ρέει από αρχείο προς το πρόγραμμα, τότε το πρόγραμμα «διαβάζει» δεδομένα.
- Αν το ρεύμα ρέει από το πρόγραμμα προς το αρχείο, τότε «γράφουμε» στο ρεύμα.
- Τέλος, «κλείνουμε» το ρεύμα.



Ρεύματα δεδομένων $_2$

Χρησιμοποιώντας την καθιερωμένη ορολογία, διακρίνουμε τα εξής βήματα:

- **Άνοιγμα αρχείου**: αντιστοιχίζουμε ένα **αντικείμενο ρεύματος στο αρχείο** από το οποίο θέλουμε να διαβάσουμε ή στο οποίο θέλουμε να γράψουμε.
- **Εγγραφή/ανάγνωση:** πραγματοποιείται η επικοινωνία με το φυσικό αρχείο δεδομένων διαβιβάζοντας μηνύματα εγγραφής και ανάγνωσης στο αντικείμενο ρεύματος που έχουμε δημιουργήσει. Στην ουσία, το φυσικό αρχείο δεδομένων «κρύβεται» μέσα στο αντικείμενο ρεύματος.
- **Κλείσιμο αρχείου**: κλείσιμο του αντικειμένου ρεύματος και τερματισμός της σύνδεσης με το φυσικό αρχείο δεδομένων. Ταυτόχρονα, απελευθερώνονται οι εσωτερικοί πόροι που είχαν δοθεί στο αντικείμενο ρεύματος. Αν ξεχάσουμε να κλείσουμε ένα αρχείο στο οποίο γράφει το πρόγραμμα, τότε υπάρχει κίνδυνος απώλειας δεδομένων.

Ρεύματα δεδομένων 3

- Όπως ήδη αναφέρθηκε, υπάρχουν πολλές κλάσεις στο πακέτο java.io για τη δημιουργία αντικειμένων ρεύματος.

- Η δημιουργία ενός αντικειμένου ρεύματος γίνεται σε αρκετές περιπτώσεις με έναν ιδιαίτερο τρόπο: ένα στιγμιότυπο μιας κλάσης ρεύματος μπορεί να χρησιμοποιηθεί ως όρισμα στον κατασκευαστή μιας άλλης κλάσης ρεύματος. Η διαδικασία αυτή μπορεί να επαναληφθεί αρκετές φορές μέχρι να καταλήξουμε σε ένα ρεύμα που θα μας ικανοποιεί.

Εγγραφή σε αρχείο κειμένου: η κλάση FileWriter (1)

- Η αποθήκευση δεδομένων σε αρχείο γίνεται, όπως ήδη αναφέρθηκε, σε τρία βήματα:

  1. Άνοιγμα του αρχείου

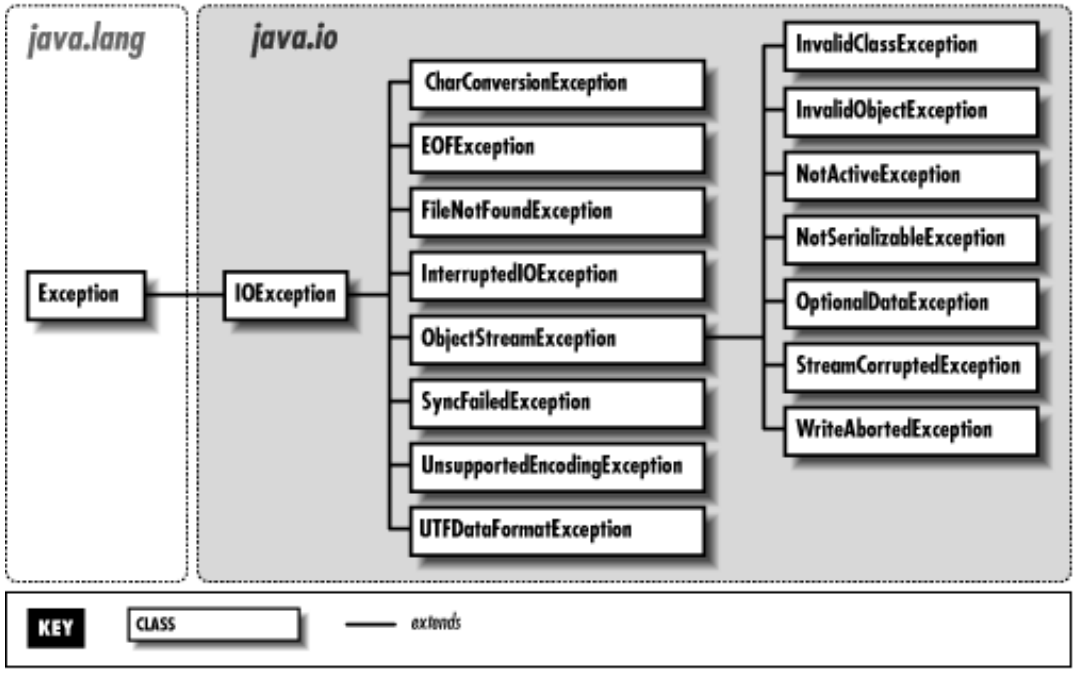
  2. Εγγραφή των δεδομένων

  3. Κλείσιμο του αρχείου

- Οποιοδήποτε από αυτά τα βήματα μπορεί να αποτύχει για διαφορετικούς λόγους, πολλοί από τους οποίους είναι πέρα από τον έλεγχο του προγραμματιστή της εφαρμογής. Για παράδειγμα, μπορεί να υπάρχει αδυναμία ανοίγματος του αρχείου ή εγγραφής στο σκληρό δίσκο κτλ.

Εγγραφή σε αρχείο κειμένου: η κλάση FileWriter (1)

- Συνεπώς, πρέπει να αναμένουμε την παραγωγή εξαιρέσεων σε οποιοδήποτε από τα παραπάνω στάδια.



Εγγραφή σε αρχείο κειμένου: η κλάση FileWriter (2)

Βήμα 1ο: άνοιγμα αρχείου

- Δημιουργούμε ένα αντικείμενο **`FileWriter`**, του οποίου ο κατασκευαστής δέχεται το όνομα του αρχείου.

```
FileWriter w = new FileWriter("…όνομα αρχείου…");
```

- Αν το αρχείο δεν υπάρχει ήδη, δημιουργείται.
- Αν το αρχείο υπάρχει και θέλουμε να γίνει εγγραφή μετά από όσα υπάρχουν ήδη στο αρχείο, τότε χρησιμοποιούμε τον παρακάτω κατασκευαστή

```
public FileWriter(String filename, boolean append)
```

με δεύτερο όρισμα την τιμή true.

- Το όνομα του αρχείου μπορεί να έχει τη μορφή αλφαριθμητικού ή ενός αντικειμένου **`File`**:

```
File f = new File("…όνομα αρχείου…");
```

Εγγραφή σε αρχείο κειμένου: η κλάση FileWriter (3)

Βήμα 1ο: άνοιγμα αρχείου

- η δημιουργία ενός αντικειμένου **`FileWriter`** έχει ως αποτέλεσμα το άνοιγμα του εξωτερικού αρχείου και την προετοιμασία του για τη λήψη της εξόδου (=εγγραφή).
- Αν το άνοιγμα του αρχείου αποτύχει για οποιονδήποτε λόγο, τότε ο κατασκευαστής θα παραγάγει μια εξαίρεση **`IOException`**.

Εγγραφή σε αρχείο κειμένου: η κλάση FileWriter (4)

Βήμα 2ο: εγγραφή δεδομένων

- Όταν ένα αρχείο ανοίξει επιτυχώς, τότε μπορεί να χρησιμοποιηθεί η μέθοδος **`write`** του εγγραφέα για την αποθήκευση χαρακτήρων — συχνά με μορφή αλφαριθμητικών — στο αρχείο:

```
w.write("…τμήμα κειμένου…");
```

- Ακόμα και αν το αρχείο ανοίξει επιτυχώς, υπάρχει πιθανότητα να αποτύχει οποιαδήποτε προσπάθεια εγγραφής σε αυτό.

Εγγραφή σε αρχείο κειμένου: η κλάση FileWriter (5)

Βήμα 3ο: κλείσιμο αρχείου

- Όταν τελειώσει η εγγραφή της εξόδου, πρέπει να κλείσουμε κανονικά το αρχείο:

```
w.close();
```

- Αυτό εξασφαλίζει ότι όλα τα δεδομένα έχουν πράγματι γραφεί στο εξωτερικό σύστημα αρχείων, και έχει συχνά ως αποτέλεσμα την αποδέσμευση κάποιων εσωτερικών ή εξωτερικών πόρων.
- Πάλι, υπάρχει πιθανότητα (αν και μικρή) να αποτύχει η προσπάθεια να κλείσουμε το αρχείο.

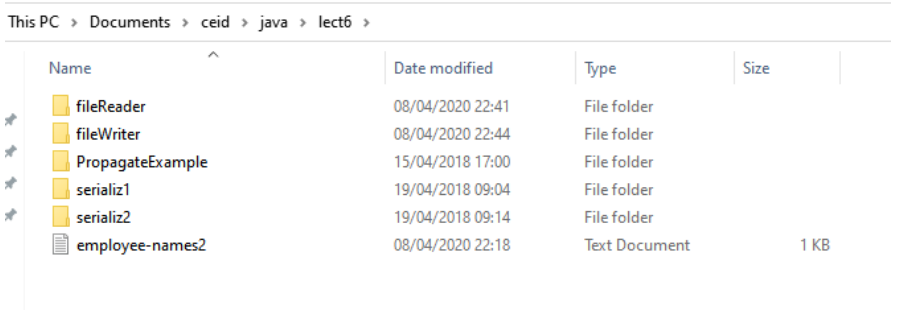
Εγγραφή σε αρχείο κειμένου: η κλάση FileWriter

```
File f = new File("... όνομα αρχείου ...");
FileWriter w = new FileWriter(f);
while(υπάρχει επιπλέον κείμενο για εγγραφή)
{
    ...
    w.write(επόμενο τμήμα κειμένου);
    ...
}
w.close();
```

Εγγραφή σε αρχείο κειμένου: η κλάση FileWriter

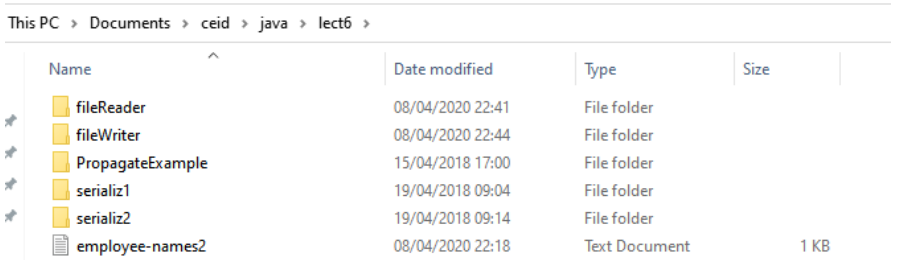
```
try
{
    File f = new File("... όνομα αρχείου ...");
    FileWriter w = new FileWriter(f);
    while(υπάρχει επιπλέον κείμενο για εγγραφή)
    {
        ...
        w.write("επόμενο τμήμα κειμένου");
        ...
    }
    w.close();
}
catch(IOException e)
{
    προέκυψε κάποιο λάθος στην προσπέλαση του αρχείου
}
```

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;


public class Main {

        public static void main(String[] args) {

                ArrayList<String> names = new ArrayList<String>();
                names.add("John");
                names.add("Mary");
                names.add("Helen");
                names.add("Nick");
                names.add("Jack");

                try
                {
                        File f = new File("..\\newFile.txt");
                        FileWriter writer = new FileWriter(f);

                        for(String name: names) {
                          writer.write(name);
                          //writer.write(System.lineSeparator());
                          writer.write("\r\n");
                        }

                        writer.close();
                }
                catch(IOException e) {
                        e.printStackTrace();
                }


        }

}
```

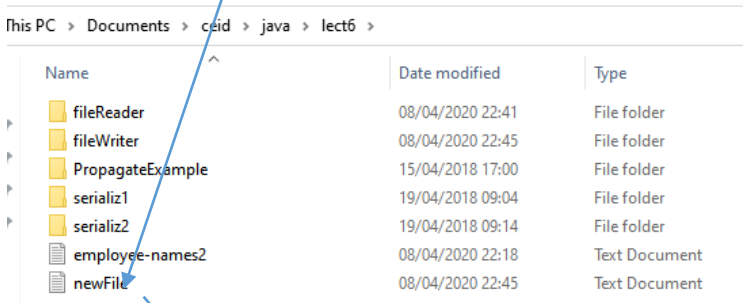


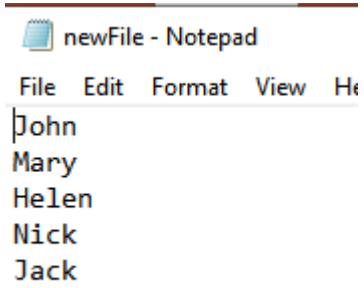
Πού γράφτηκε το αρχείο;

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;


public class Main {

        public static void main(String[] args) {

                ArrayList<String> names = new ArrayList<String>();
                names.add("John");
                names.add("Mary");
                names.add("Helen");
                names.add("Nick");
                names.add("Jack");

                try
                {
                        File f = new File("..\\newFile.txt");
                        FileWriter writer = new FileWriter(f);

                        for(String name: names) {
                          writer.write(name);
                          //writer.write(System.lineSeparator());
                          writer.write("\r\n");
                        }

                        writer.close();
                }
                catch(IOException e) {
                        e.printStackTrace();
                }


        }

}
```

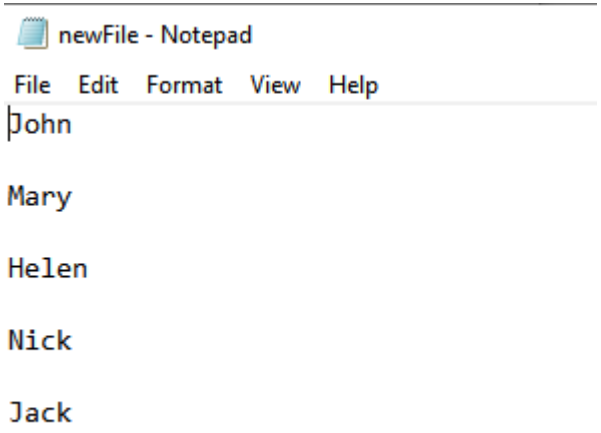


Πού γράφτηκε το αρχείο;



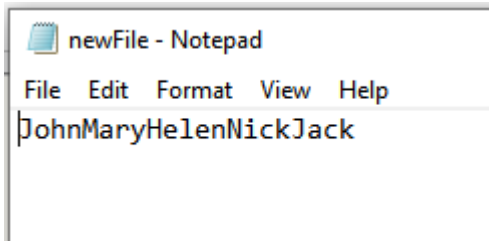


```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;


public class Main {

	public static void main(String[] args) {

		ArrayList<String> names = new ArrayList<String>();
		names.add("John");
		names.add("Mary");
		names.add("Helen");
		names.add("Nick");
		names.add("Jack");

		try
		{
			File f = new File("..\\newFile.txt");
			FileWriter writer = new FileWriter(f);

			for(String name: names) {
			  writer.write(name);
			 writer.write(System.lineSeparator());
			  writer.write("\r\n");
			}

			writer.close();
		}
		catch(IOException e) {
			e.printStackTrace();
		}


	}

}
```

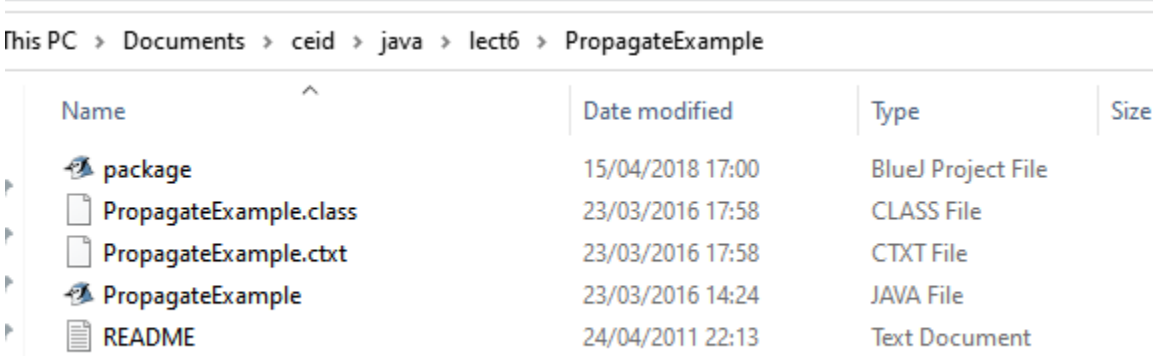


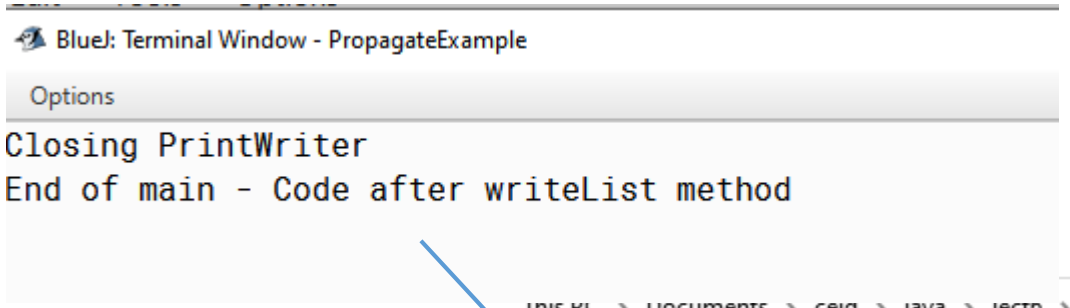
Strings have no newlines. We can form them into two lines by concatenating a newline string. Use System lineSeparator to get a platform-dependent newline string., https://www.tutorialspoint.com/line-separator-in-java

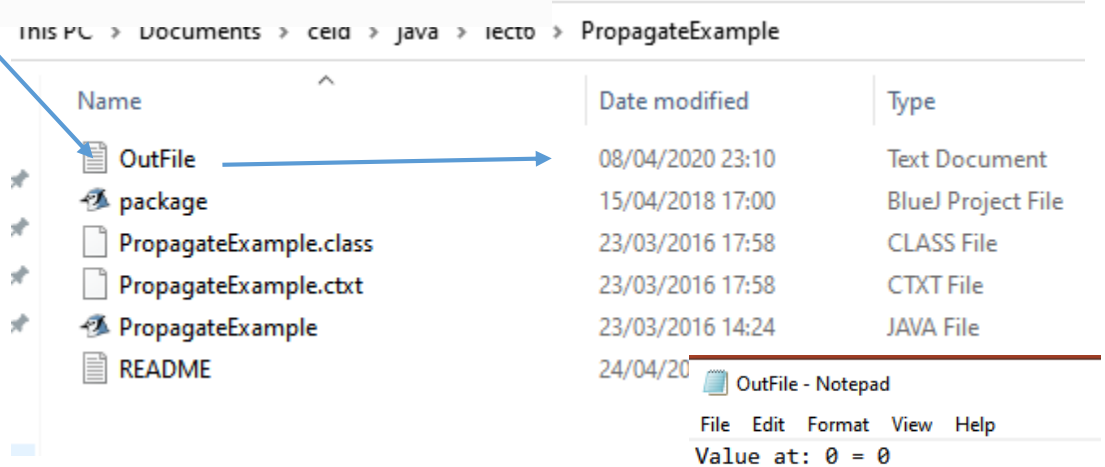
```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;


public class Main {

	public static void main(String[] args) {

		ArrayList<String> names = new ArrayList<String>();
		names.add("John");
		names.add("Mary");
		names.add("Helen");
		names.add("Nick");
		names.add("Jack");

		try
		{
			File f = new File("..\\newFile.txt");
			FileWriter writer = new FileWriter(f);

			for(String name: names) {
			  writer.write(name);
			  //writer.write(System.lineSeparator());
			  //writer.write("\r\n");
			}

			writer.close();
		}
		catch(IOException e) {
			e.printStackTrace();
		}


	}

}
```

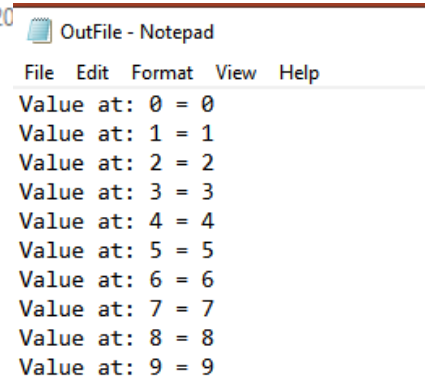


```
import java.io.*;
import java.util.List;
import java.util.ArrayList;

public class PropagateExample {

  private static final int SIZE = 10;

  public static void main(String args[]) {
    try{
       writeList();
    }catch (FileNotFoundException e){ // necessary to be caught, as it is being propaged from the writeList method
       System.out.println("Problem with this file!");
    }
    System.out.println("End of main - Code after writeList method");
  }

 public static void writeList() throws FileNotFoundException {
// throws necessary : FileNotFoundException is propagated

   List<Integer> list = new ArrayList<Integer>(SIZE);

    for(int index = 0; index < SIZE; index++) {
      list.add(new Integer(index));     }

    PrintWriter out = null;

    try{
       FileWriter fw =  new FileWriter("OutFile.txt");
       out = new PrintWriter(fw);

       for (int i = 0; i < SIZE; i++) {
          out.println("Value at: " + i + " = " + list.get(i)); }
    }
    catch (FileNotFoundException e) {
       System.err.println("FileNotFoundException: " + e.getMessage());
       throw e;
    }
    catch (IOException e)
    {  System.err.println("Caught IOException: " + e.getMessage());     }
    finally
    {   if (out != null) {
          System.out.println("Closing PrintWriter");
          out.close();
       } else {
          System.out.println("PrintWriter not open");
    }     }  }}
```


This PC › Documents › ceid › java › lect6 › PropagateExample
Name
Date modified
Type
Size
package
15/04/2018 17:00
BlueJ Project File
PropagateExample.class
23/03/2016 17:58
CLASS File
PropagateExample.ctxt
23/03/2016 17:58
CTXT File
PropagateExample
23/03/2016 14:24
JAVA File
README
24/04/2011 22:13
Text Document


BlueJ: Terminal Window - PropagateExample
Options
Closing PrintWriter
End of main - Code after writeList method


This PC › Documents › ceid › java › lect6 › PropagateExample
Name
Date modified
Type
OutFile
08/04/2020 23:10
Text Document
package
15/04/2018 17:00
BlueJ Project File
PropagateExample.class
23/03/2016 17:58
CLASS File
PropagateExample.ctxt
23/03/2016 17:58
CTXT File
PropagateExample
23/03/2016 14:24
JAVA File
README


OutFile - Notepad
File Edit Format View Help
Value at: 0 = 0
Value at: 1 = 1
Value at: 2 = 2
Value at: 3 = 3
Value at: 4 = 4
Value at: 5 = 5
Value at: 6 = 6
Value at: 7 = 7
Value at: 8 = 8
Value at: 9 = 9

Ανάγνωση από αρχείο κειμένου: η κλάση FileReader (1)

Το διάβασμα δεδομένων από αρχείο γίνεται, όπως ήδη αναφέρθηκε, σε τρία βήματα:

1. Άνοιγμα του αρχείου
2. Ανάγνωση των δεδομένων
3. Κλείσιμο του αρχείου

Ανάγνωση από αρχείο κειμένου: η κλάση FileReader (2)

- Ενώ **οι φυσικές μονάδες για την εγγραφή κειμένου είναι οι χαρακτήρες και τα αλφαριθμητικά**, οι αντίστοιχες **για την ανάγνωση** του κειμένου είναι οι **χαρακτήρες και οι γραμμές**.
- Ωστόσο, παρόλο που η κλάση **`FileReader`** περιλαμβάνει μια μέθοδο για την ανάγνωση ενός χαρακτήρα, δεν περιλαμβάνει κάποια μέθοδο για την ανάγνωση μίας γραμμής.
- Το πρόβλημα με την ανάγνωση γραμμών από αρχείο είναι ότι **δεν υπάρχει κάποιο προκαθορισμένο όριο για το μήκος γραμμής**.
- Αυτό σημαίνει ότι οποιαδήποτε μέθοδος χρησιμοποιηθεί για την επιστροφή της επόμενης ολόκληρης γραμμής από ένα αρχείο πρέπει να μπορεί να διαβάσει έναν οσοδήποτε μεγάλο αριθμό χαρακτήρων.

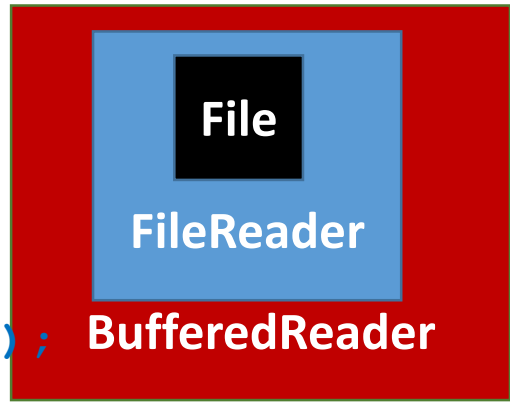
Ανάγνωση από αρχείο κειμένου: η κλάση FileReader (3)

- Γι' αυτό, συνήθως, το αντικείμενο **FileReader** περιτυλίγεται σε ένα αντικείμενο **BufferedReader**:

```
FileReader freader = new FileReader("…όνομα αρχείου…");
BufferedReader reader = new BufferedReader(freader);
```

ή συνοπτικά

```
BufferedReader reader = new BufferedReader(
                    new FileReader("…όνομα αρχείου…"));
```



- Ένα αντικείμενο **BufferedReader** διαθέτει τη μέθοδο **readLine** για το διάβασμα μιας γραμμής.

- Ο χαρακτήρας τερματισμού γραμμής αφαιρείται πάντα από το αλφαριθμητικό που επιστρέφεται
- Το τέλος ενός αρχείου υποδηλώνεται με μια τιμή **null**.

Ανάγνωση από αρχείο κειμένου: η κλάση FileReader (4)

- Η κλάση **`BufferedReader`**, εκτός από το γεγονός ότι διαθέτει τη μέθοδο **`readLine`**, παρουσιάζει και ένα άλλο σημαντικό πλεονέκτημα:
- Η κλάση **`BufferedReader`** χρησιμοποιεί μια **`περιοχή προσωρινής αποθήκευσης`** (buffer), δηλαδή μια περιοχή στη μνήμη που λειτουργεί ως προσωρινός αποθηκευτικός χώρος.
- Το τυπικό μέγεθος μιας περιοχής προσωρινής αποθήκευσης που χρησιμοποιείται για την ενδιάμεση αποθήκευση δεδομένων από ένα αρχείο σε ένα πρόγραμμα Java είναι 8192 χαρακτήρες.
- Με τη χρήση του buffer η προσπέλαση του δίσκου είναι λιγότερο συχνή, κάτι που αυξάνει την ταχύτητα του προγράμματος.

Default buffer size is 8192 characters capacity. Line size is considered as 80 chars capacity. 8192 buffer size is sufficient for smaller file sizes., Stack Overflowhttps://stackoverflow.com

Ανάγνωση από αρχείο κειμένου: η κλάση FileReader (5)

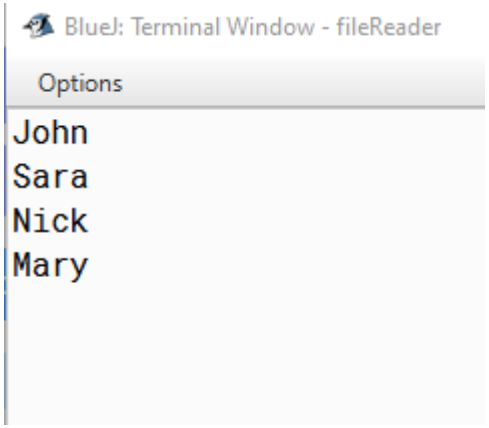
```
File f = new File("... όνομα αρχείου ...");
BufferedReader reader = new BufferedReader(new FileReader(f));
String line = reader.readLine();
while(line != null)
{
    κάποια ενέργεια με τη γραμμή
    line = reader.readLine();
}
reader.close();
```

Είσοδος κειμένου: η κλάση FileReader (5)

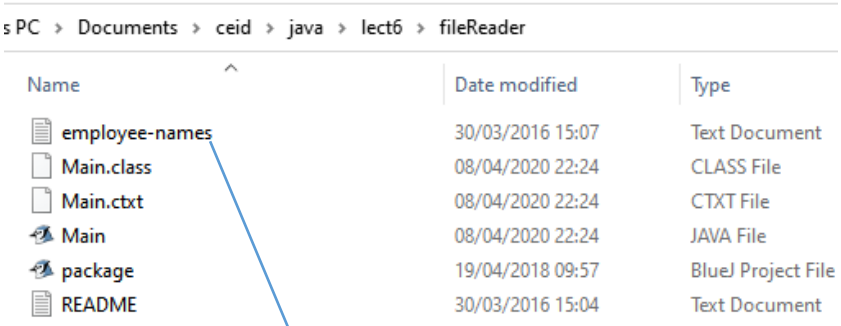
```
try
{
  File f = new File("... όνομα αρχείου ...");
  BufferedReader reader = new BufferedReader(new FileReader(f));
  String line = reader.readLine();
  while(line != null)
  {
      κάποια ενέργεια με τη γραμμή
      line = reader.readLine();
  }
  reader.close();
}
catch(FileNotFoundException e)
{
   //δεν εντοπίστηκε το συγκεκριμένο αρχείο
}
catch(IOException e)
{
   //κάποιο πρόβλημα προέκυψε στην ανάγνωση ή στο κλείσιμο
}
```

```
import java.io.*;
import java.util.ArrayList;
public class Main {

   public static void main(String[] args) {

      ArrayList<String> names = new ArrayList<String>();

      try
      {
         FileReader fReader = new FileReader( "employee-names.txt");
         BufferedReader reader = new BufferedReader(fReader);
         String line = reader.readLine();

         while (line != null){
                  names.add(line);
                  line=reader.readLine();
          }
         reader.close();
         fReader.close();
      }

      catch(IOException e) {
         e.printStackTrace();
      }
       for (String name:names){
      System.out.println(name);
   }
}
}
```

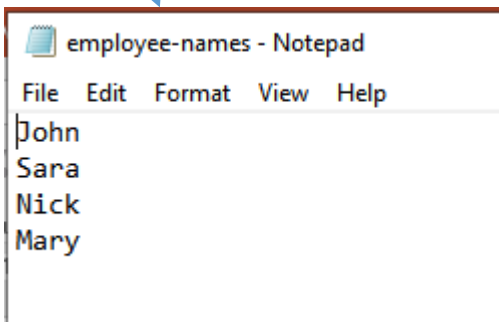
Πού βρήκε τα ονόματα;

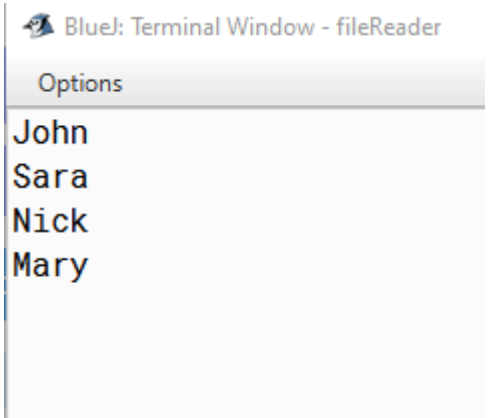


```
import java.io.*;
import java.util.ArrayList;
public class Main {

   public static void main(String[] args) {

      ArrayList<String> names = new ArrayList<String>();

      try
      {
         FileReader fReader = new FileReader( "employee-names.txt");
         BufferedReader reader = new BufferedReader(fReader);
         String line = reader.readLine();

         while (line != null){
                names.add(line);
                line=reader.readLine();
          }
         reader.close();
         fReader.close();
      }

      catch(IOException e) {
         e.printStackTrace();
      }
       for (String name:names){
      System.out.println(name);
   }
}
}
```

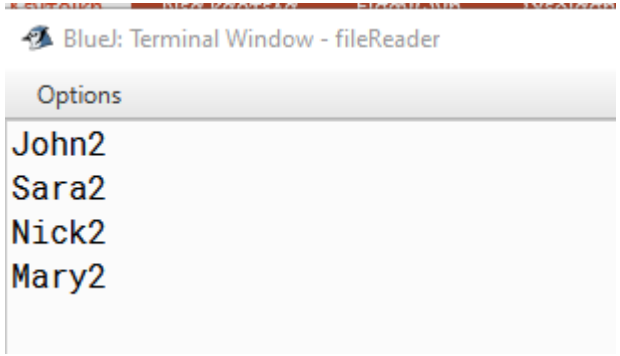


Πού βρήκε τα ονόματα;



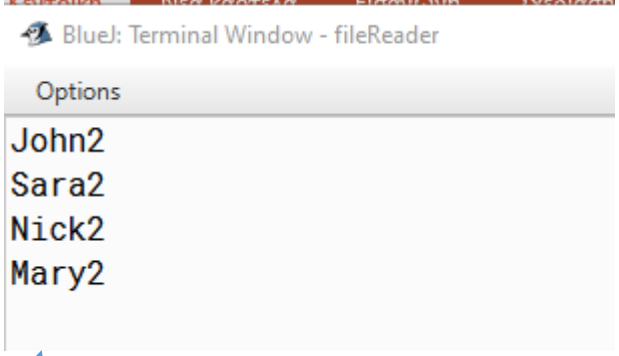


```
import java.io.*;
import java.util.ArrayList;


public class Main {

   public static void main(String[] args) {

      ArrayList<String> names = new ArrayList<String>();

      try
      {FileReader fReader = new FileReader( new File("C:\\Users\\evoyiatzaki\\Documents\\ceid\\java\\lect6", "employee-names2.txt"));
        BufferedReader reader = new BufferedReader(fReader);
          String line = reader.readLine();

          while (line != null){
                names.add(line);
                line=reader.readLine();
           }

          reader.close();
          fReader.close();
      }

      catch(IOException e) {
          e.printStackTrace();
      }
       for (String name:names){
      System.out.println(name);
   }
}
}
```

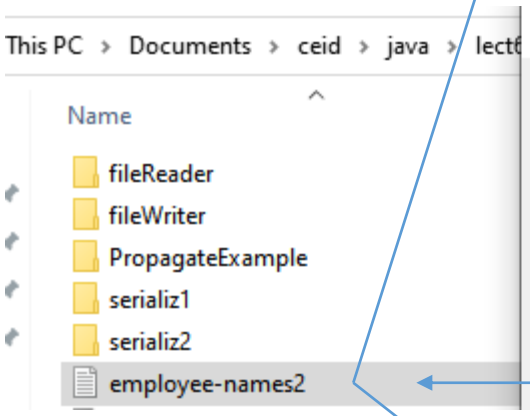


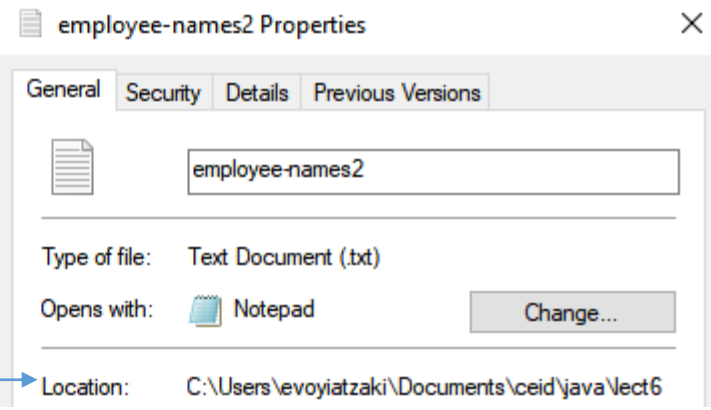
Πού βρήκε τα ονόματα;

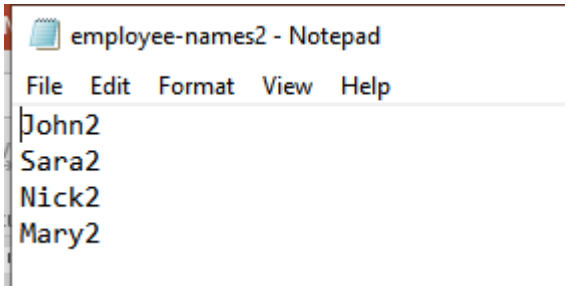
```
import java.io.*;
import java.util.ArrayList;


public class Main {

   public static void main(String[] args) {

      ArrayList<String> names = new ArrayList<String>();

      try
      {FileReader fReader = new FileReader( new File("C:\\Users\\evoyiatzaki\\Documents\\ceid\\java\\lect6", "employee-names2.txt"));
        BufferedReader reader = new BufferedReader(fReader);
          String line = reader.readLine();

          while (line != null){
                names.add(line);
                line=reader.readLine();
           }

          reader.close();
          fReader.close();
      }

      catch(IOException e) {
          e.printStackTrace();
      }
       for (String name:names){
      System.out.println(name);
   }
}
}
```



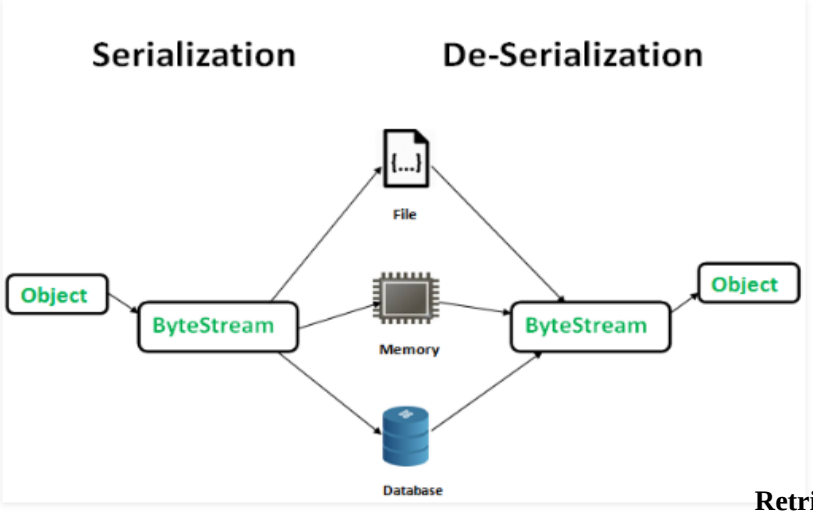
Πού βρήκε τα ονόματα;







Σειριακοποίηση ή σειριοποίηση αντικειμένων (1)



Retrieved from https://www.geeksforgeeks.org/serialization-in-java/

- Η **σειριοποίηση** επιτρέπει
  - την εγγραφή ενός ολόκληρου αντικειμένου σε ένα εξωτερικό αρχείο με μία λειτουργία
  - την ανάγνωση ενός ολόκληρου αντικειμένου από ένα εξωτερικό αρχείο με μία λειτουργία
- Η σειριακοποίηση επιτρέπει επίσης να γραφούν και να διαβαστούν αντικείμενα όχι μόνο σε ένα σύστημα αρχείων, αλλά και σε ένα δίκτυο.
- Η σειριακοποίηση λειτουργεί τόσο με απλά αντικείμενα όσο και με αντικείμενα πολλών στοιχείων, όπως οι συλλογές.
- Με την σειριοποίηση αποφεύγεται η ανάγνωση και η εγγραφή αντικειμένων πεδίο προς πεδίο.

Σειριακοποίηση αντικειμένων (2)

- Προκειμένου να είναι κατάλληλη για σειριακοποίηση, μια κλάση πρέπει να υλοποιεί τη διασύνδεση **`Serializable`** η οποία ορίζεται στο πακέτο **`java.io`**.
- Η διασύνδεση **`Serializable`** δεν ορίζει μεθόδους (marking interface).
- Αυτό σημαίνει ότι γίνεται αυτόματη διαχείριση της διαδικασίας σειριακοποίησης από το σύστημα χρόνου εκτέλεσης, και από το χρήστη απαιτείται απλά να γράψει λίγες γραμμές κώδικα.

- Αν κάποιο πεδίο της κλάσης που έχουμε δηλώσει ότι υλοποιεί το serializable interface δεν θέλουμε να γίνει serialize (δηλ. να αποθηκεύεται) τότε το δηλώνουμε σαν transient.

Εγγραφή σε δυαδικό αρχείο (1)

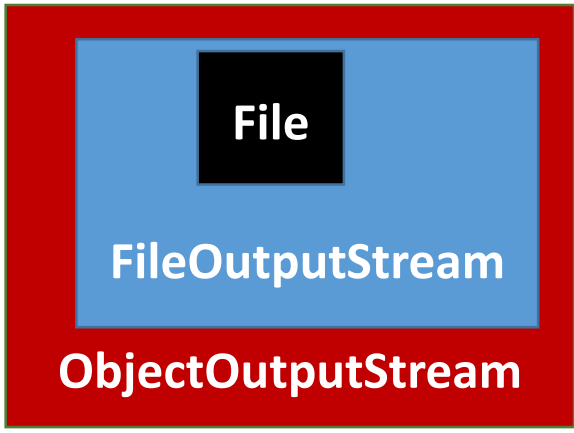
- Η αποθήκευση αντικειμένων σε αρχείο γίνεται σε τρία βήματα:

  1. Άνοιγμα του αρχείου
  2. Εγγραφή των δεδομένων
  3. Κλείσιμο του αρχείου

- Κάθε αντικείμενο που εμπλέκεται πρέπει να προέρχεται από μια κλάση που υλοποιεί τη διασύνδεση Serializable.

- Για να γράψουμε αντικείμενα σε ένα αρχείο πρέπει να δημιουργήσουμε ένα αντικείμενο ObjectOutputStream, ακολουθώντας τα παρακάτω βήματα:

Εγγραφή σε δυαδικό αρχείο (2)



Βήμα 1ο: άνοιγμα αρχείου για εγγραφή αντικειμένων

1. Δημιουργούμε ένα αντικείμενο αρχείου **File**:

```
File f = new File("....."); 
```

2. Δημιουργούμε ένα αντικείμενο **FileOutputStream** χρησιμοποιώντας ως όρισμα το αντικείμενο **File** που δημιουργήσαμε παραπάνω:

```
FileOutputStream fouts = new FileOutputStream(f);
```

3. Δημιουργούμε ένα αντικείμενο **ObjectOutputStream** χρησιμοποιώντας ως όρισμα το αντικείμενο **fouts**

```
ObjectOutputStream douts = new ObjectOutputStream(fouts);
```

Οι δύο τελευταίες εντολές μπορούν να συνδυαστούν σε μία:

```
ObjectOutputStream douts = new ObjectOutputStream(
                                   new FileOutputStream(f));
```

Έξοδος σε δυαδικό αρχείο (3)

Βήμα 2ο: εγγραφή αντικειμένων

- Για την εγγραφή των αντικειμένων χρησιμοποιείται η μέθοδος `writeObject()`:

```
douts.writeObject(αντικείμενο);
```

Βήμα 3ο: κλείσιμο αρχείου

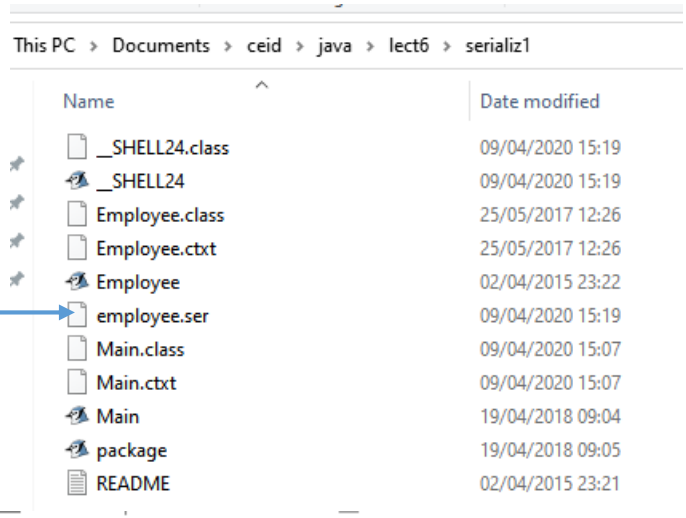
```
douts.close();
fouts.close();
```

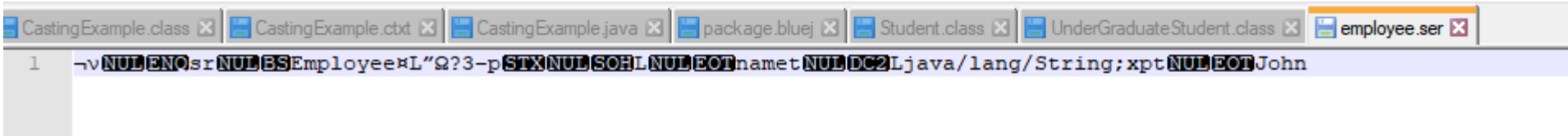
Παράδειγμα: **Serializ1**

```
import java.io.*;
public class Main {
        public static void main(String[] args) {

        Employee e = new Employee("John");

        try {
                FileOutputStream fileOut = new
FileOutputStream("employee.ser");
                ObjectOutputStream out = new
ObjectOutputStream(fileOut);
                out.writeObject(e);
                out.close();
                fileOut.close();
                }
        catch(IOException i) {
                i.printStackTrace();
                }

        System.out.println("Serialized Object...");
        System.out.println("Name: " + e.getName());
        }
}
```

breakpoint

```
import java.io.Serializable;

public class Employee implements Serializable
{

        private String name;

        public Employee(String text) {
                name = text;
        }

        public String getName() {
                return name;
        }

}
```





Ανάγνωση από δυαδικό αρχείο (1)

- Η ανάγνωση αντικειμένων από αρχείο γίνεται σε τρία βήματα:

  1. Άνοιγμα του αρχείου
  2. Ανάγνωση των αντικειμένων
  3. Κλείσιμο του αρχείου

- Κάθε αντικείμενο που εμπλέκεται πρέπει να προέρχεται από μια κλάση που υλοποιεί τη διασύνδεση **`Serializable`**.

- Για να διαβάσουμε αντικείμενα από ένα αρχείο πρέπει να δημιουργήσουμε ένα αντικείμενο **`ObjectInputStream`**, ακολουθώντας τα παρακάτω βήματα:

Ανάγνωση από δυαδικό αρχείο (2)

Βήμα 1ο: άνοιγμα αρχείου για διάβασμα αντικειμένων

1. Δημιουργούμε ένα αντικείμενο αρχείου **`File`**:

```
File f = new File("...");
```

2. Δημιουργούμε ένα αντικείμενο **`FileInputStream`** χρησιμοποιώντας ως όρισμα το αντικείμενο **`File`** που δημιουργήσαμε παραπάνω:

```
FileInputStream fins = new FileInputStream(f);
```

3. Δημιουργούμε ένα αντικείμενο **`ObjectInputStream`** χρησιμοποιώντας ως όρισμα το αντικείμενο **`fins`**

```
ObjectInputStream dins = new ObjectInputStream(fins);
```

Οι δύο τελευταίες εντολές μπορούν να συνδυαστούν σε μία:

```
ObjectInputStream dins =

new ObjectIputStream(new FileInputStream(f));
```

Ανάγνωση από δυαδικό αρχείο (3)

Βήμα 2ο: διάβασμα αντικειμένων

- Για την ανάγνωση/διάβασμα των αντικειμένων χρησιμοποιείται η μέθοδος **`readObject`** της κλάσης **`ObjectInputStream`**.
- Η μέθοδος **`readObject`** επιστρέφει ένα αντικείμενο τύπου **`Object`**, οπότε χρειάζεται να γίνει casting για την προσαρμογή του επιστρεφόμενου αντικειμένου στον κατάλληλο τύπο:

```
(τύπος αντικειμένου) dins.readObject();
```

- ***Βήμα 3ο: κλείσιμο αρχείου***

```
dins.close();
fins.close();
```

Παράδειγμα: **Serializ2**

```
import java.io.Serializable;

public class Employee implements Serializable {

		private String name;
		private Company company;

		public Employee(String text, Company c) {
			name = text;
			company = c;
		}

		public void printInfo() {
			System.out.println("Name: " +
name + " works for :" + company.getName());
		}

}
```

```
import java.io.Serializable;

public class Company implements Serializable {

		private String name;

		public Company(String text) {
			name = text;
		}

		public String getName() {
			return name;
		}

}
```

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

public class Main {

		public static void main(String[] args) {

			SerializationDemo demo = new
SerializationDemo();


		}
```

```
public class SerializationDemo {

        private ArrayList<Employee> employees = new ArrayList<Employee>();

        public SerializationDemo() {
                Company C1 = new Company("Oracle");
                Company C2 = new Company("Google");
                employees.add(new Employee("John", C1));
                employees.add(new Employee("Bob", C2));
                employees.add(new Employee("Nikol", C1));

                serializing();

                ArrayList<Employee> employeesFromFile = deserializing();

                for(Employee e: employeesFromFile) {
                        e.printInfo();
                }

        }
                public void serializing() {
                try {
                        FileOutputStream fileOut = new FileOutputStream("employee.ser");
                        ObjectOutputStream out = new ObjectOutputStream(fileOut);
                        out.writeObject(employees);
                        out.close();
                        fileOut.close();
                }
                catch(IOException i) {
                        i.printStackTrace();
                }
                finally {
                        System.out.println("Serialization Attempted...");
                }
        }
```

SerializationDemo()(part1)

SerializationDemo()(part2)

```
public ArrayList<Employee> deserializing() {
                try {
                        FileInputStream fileIn = new FileInputStream("employee.ser");
                        ObjectInputStream in = new ObjectInputStream(fileIn);
                        ArrayList<Employee> employees = (ArrayList<Employee>) in.readObject();
                                                        // ATTENTION
                        in.close();
                        fileIn.close();
                }
                catch(IOException i) {
                        i.printStackTrace();
                }
                catch(ClassNotFoundException c) {
                        c.printStackTrace();
                }
                finally {
                        System.out.println("De-Serialization Attempted...");
                        return employees;
                }

        }


}
```

```
public void serializing() {
		try {
				FileOutputStream fileOut = new
FileOutputStream("employee.jjj");
				ObjectOutputStream out = new
ObjectOutputStream(fileOut);
				out.writeObject(employees);
				out.close();
				fileOut.close();
				}
		catch(IOException i) {
				i.printStackTrace();
				}
		finally {
				System.out.println("Serialization Attempted...");
				}

		}

		public ArrayList<Employee> deserializing() {
		try {
				FileInputStream fileIn = new
FileInputStream("employee.jjj");
				ObjectInputStream in = new
ObjectInputStream(fileIn);
				ArrayList<Employee> employees =
(ArrayList<Employee>) in.readObject();// ATTENTION
				in.close();
```