# Κατανεμημένα Συστήματα
## 1ο Φυλλάδιο εργαστηρίου 02/03/2017

## Ροές Δεδομένων στην Java

Όπως και κάθε σύγχρονη γλώσσα προγραμματισμού, η Java προσφέρει ένα πλούσιο σύνολο κλάσεων και μεθόδων για διαχείριση αρχείων. Με τη Java μπορείτε να ανοίξετε αρχεία, να διαβάσετε και να γράψετε δεδομένα στα αρχεία και μάλιστα διατίθενται κλάσεις για να αυτοματοποιηθούν ορισμένες διαδικασίες, π.χ. αυτόματη αποθήκευση ενός πίνακα από String σε ένα αρχείο ή φιλτράρισμα των δεδομένων που διαβάζονται από ένα αρχείο.

Η είσοδος/έξοδος δεδομένων πραγματοποιείται μέσω αντικειμένων τα οποία ονομάζονται **ροές** (**streams**) τα οποία περιγράφουν ακριβώς ακολουθίες δεδομένων, bytes ή χαρακτήρων. Τονίζεται ότι μια ροή μπορεί να αναφέρεται στην ανάγνωση ή εγγραφή από ένα αρχείο, καθώς και από άλλους πόρους εισόδου/εξόδου όπως συνδέσεις δικτύου, εκτυπωτές, κ.λπ. Η Java παρέχει ένα πλήρες σύνολο κλάσεων για την προσπέλαση των δεδομένων ενός αρχείου και μάλιστα σε δύο επίπεδα, ένα σε *επίπεδο byte (byte streams)* και ένα σε *επίπεδο χαρακτήρων (character streams)*. Όπως μαρτυρά και το όνομα τους, τα byte streams χρησιμοποιούνται για την επεξεργασία των δεδομένων των αρχείων, για ανάγνωση και εγγραφή και ειδικά όταν τα δεδομένα δε μπορούν να απεικονιστούν ως κείμενο (π.χ. εικόνα, ήχος). Για την περίπτωση αρχείων κειμένου, τα character streams προσφέρουν επιπλέον λειτουργίες, όπως απευθείας κωδικοποίηση σε κάποιο σύνολο χαρακτήρων (π.χ. ISO-8859-7, UTF-8). Σε μερικές περιπτώσεις μάλιστα και ειδικά για αρχεία κειμένου, τα character streams αποδίδουν καλύτερα από τα byte streams.
Για να δημιουργήσουμε μια ροή δεδομένων, αρχικά θα χρησιμοποιήσουμε την κατάλληλη κλάση που εξαρτάται από την συσκευή που βρίσκεται στην άλλη μεριά της ροής μας και τον τύπο των δεδομένων που θέλουμε να χειριστούμε. Παραδείγματος χάριν, για την περίπτωση των δυαδικών αρχείων οι κατάλληλες κλάσεις είναι οι *FileInputStream* και *FileOutputStream*.

Η κύρια λειτουργία των βασικών ροών εισόδου/εξόδου είναι να μεταφέρουν τα δεδομένα όπως ακριβώς δίνονται στις μεθόδους τους ένα-ένα byte κάθε φορά χωρίς κάποια επεξεργασία. Για να αυξήσουμε λοιπόν την λειτουργικότητα μίας βασικής ροής "τυλίγουμε" (wrap) γύρω της μια ανωτέρου επιπέδου ροή. Αυτή η κατηγορία των κλάσεων αυξάνει την αποδοτικότητα του προγράμματός μας καθώς μπορεί να κάνει buffering των δεδομένων ή να τα επεξεργαστεί κατάλληλα. Παράδειγμα μιας τέτοιας ροής είναι οι κλάσεις *BufferedInputStream* και *BufferedOutputStream*, που χρησιμοποιούν ενδιάμεση μνήμη (buffer) για να αυξήσουν την απόδοση της εισόδου/εξόδου δεδομένων. Το "τύλιγμα" της βασικής ροής επιτυγχάνεται με το να την περάσουμε σαν παράμετρο στον κατασκευαστή (constructor) της ανώτερης ροής.

Ακολουθούν παραδείγματα για ανάγνωση και εγγραφή χαρακτήρων σε αρχείο:

### Ροές χαρακτήρων

#### 1ο Παράδειγμα - Ανάγνωση από αρχείο χαρακτήρων

```
import java.io.*;

public class Main  {
    public static void main( String args[] ) {
```

```
            BufferedReader in;
            try {
            in = new BufferedReader(new FileReader("InData.txt"));

             // Διάβασμα αρχείου ανά χαρακτήρα
            // while (in.ready()) // Έλεγχος αν μπορεί να γίνει ανάγνωση από stream
            // {
            // char ch = (char) in.read();
            // System.out.print( ch );
            // }

           String str ;
           // Διάβασμα αρχείου ανά γραμμή
           while( (str=in.readLine()) !=null){
                System.out.println( str );
                  }
           in.close();
            }
            catch ( FileNotFoundException e ) {
             System.out.println("File not found");
            }
            catch( IOException e ) {
                System.out.println("Read error");
             }
          }
}
```

## 2ο Παράδειγμα - Εγγραφή σε αρχείο χαρακτήρων

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

public class Main {
public static void main( String args[] ) {
        BufferedWriter out;
        try {
           out = new BufferedWriter(new FileWriter("OutData.txt"));
           System.out.println("Enter text: ");
           BufferedReader in = new BufferedReader(new
InputStreamReader(System.in));

           String str = in.readLine();
           //Scanner in = new Scanner (System.in);
           //String str = in.nextLine();

            out.write(str, 0, str.length());
           //out.write(str);
            out.close();
    }

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

## Ροές bytes

Οι κλάσεις οι οποίες αναπαριστούν ροές από bytes είναι υποκλάσεις των κλάσεων **InputStream** και **OutputStream**, αντίστοιχα. Υποκλάσεις των παραπάνω οι οποίες αναφέρονται σε ροές αρχείων είναι οι **FileInputStream** και **FileOutputStream**, αντίστοιχα.
Το άνοιγμα μιας τέτοιας ροής γίνεται με χρήση του αντίστοιχου κατασκευαστή. Έτσι, το άνοιγμα ενός αρχείου για ανάγνωση μέσω μιας ροής από bytes γίνεται με την εντολή:

**FileInputStream input = new FileInputStream("data.txt");**

όπου data.txt είναι το όνομα του αρχείου το οποίο θέλουμε να ανοίξουμε για ανάγνωση. Το άνοιγμα ενός αρχείου για εγγραφή μέσω μιας ροής από bytes γίνεται με την εντολή:

**FileOutputStream output = new FileOutputStream("data.txt");**

όπου data.dat είναι το όνομα του αρχείου το οποίο θέλουμε να ανοίξουμε για εγγραφή. Η ανάγνωση ενός byte από το αρχείο γίνεται με χρήση της μεθόδου ***byte read()*** της κλάσης InputStream (άρα και της FileInputStream) η οποία επιστρέφει το *επόμενο* byte στη ροή, ξεκινώντας από το πρώτο (το επιστρέφει με τη μορφή int). Όταν η ανάγνωση φτάσει στο τέλος της ροής (αρχείου στην προκειμένη περίπτωση) η read() επιστρέφει την τιμή -1.
Η *εγγραφή* ενός byte σε ένα αρχείο γίνεται με χρήση της μεθόδου ***write(byte b)*** της κλάσης OutputStream (άρα και της FileOutputStream) η οποία εγγράφει το byte b του ορίσματός της στη ροή.
Μετά την ολοκλήρωση της προσπέλασης σε μια ροή είναι απαραίτητη η αποδέσμευση του σχετικού πόρου. Αυτό γίνεται με τη μέθοδο close() των κλάσεων InputStream και OutputStream.
Όλες οι παραπάνω μέθοδοι και κατασκευαστές εγείρουν, σε περίπτωση σφάλματος, μια εξαίρεση του τύπου IOException.

## 3ο Παράδειγμα

Υλοποιήστε πρόγραμμα που αντιγράφει τα περιεχόμενα ενός αρχείου (File1.txt) σε ένα άλλο αρχείο (File2.txt). Η ανάγνωση και η εγγραφή θα γίνει ανά byte.

```
import java.io.*;

class CopyFile {

public static void main(String args[]) {
int i;
FileInputStream fin;
FileOutputStream fout;

try {
	fin = new FileInputStream("File1.txt");
}
catch (FileNotFoundException exc) {
	System.out.println("Input file not found: ");
	return;
}

try {
	fout = new FileOutputStream("File2.txt");
```

```
} catch (FileNotFoundException exc) {
       System.out.println("Error opening output file: ");
       return;
}

try {
       do {
              i = fin.read();
              if (i != -1) // δηλαδή αν δεν έχουμε φτάσει
                     fout.write(i); // στο τέλος του αρχείου fin
       } while (i != -1);

       fin.close();
       fout.close();
}

catch (IOException exc) {
       System.out.println("File error");
}

       }
}
```

## 4ο Παράδειγμα

Το ακόλουθο πρόγραμμα δημιουργεί αρχείο *data.txt* στο οποίο γράφει μια σειρά από αριθμούς σε δυαδική μορφή. Οι ίδιοι αριθμοί εμφανίζονται στην οθόνη μετά την ανάγνωση τους από το αρχείο.

```
import java.io.*;
public class FileStreamExample {

public static void main(String args[]) {

      try {
         byte bArray [] = {10, 2, 30, 40, 5};
         OutputStream outs = new FileOutputStream("data.txt");

         for(int i = 0; i < bArray.length ; i++) {
            outs.write( bArray[i] );
         }
         outs.close();

         InputStream inputs = new FileInputStream("data.txt");
         int size = inputs.available();

         for(int i = 0; i < size; i++) {
            System.out.print( inputs.read() + "  ");
         }
         inputs.close();

      }catch(IOException e) {
         System.out.print("Exception");
      }
   }
}
```

## 1η Εργαστηριακή Άσκηση

Δημιουργείστε ένα αρχείο στο οποίο ο χρήστης να μπορεί να καταχωρεί τις επαφές του. Ζητήστε από τον χρήστη επαναληπτικά το όνομα, το επίθετο, το σταθερό και το κινητό τηλέφωνο κάθε επαφής. Στο τέλος κάθε επανάληψης ο χρήστης πρέπει να ρωτάτε εάν θέλει να συνεχίσει με την εισαγωγή επαφών.

### Απάντηση

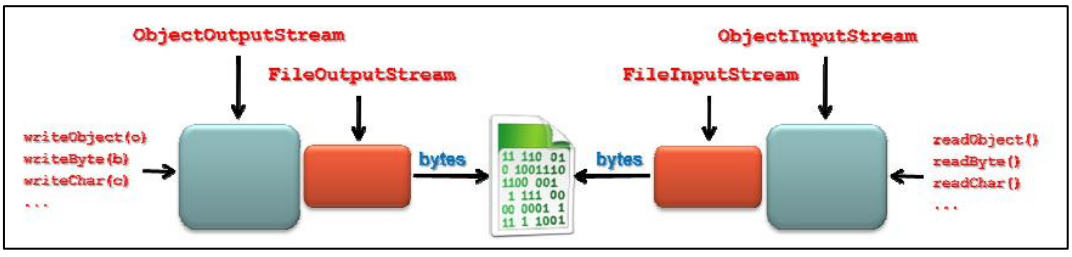
```
import java.io.*;

public class Buffer
{
public static void main (String [] args)
{
boolean prostheseEpafi=true;
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
BufferedWriter out;

try {
out = new BufferedWriter(new FileWriter("contacts.txt"));

while(prostheseEpafi)
{
System.out.print("Enter name: ");
String name=br.readLine();

System.out.print("Enter surname: ");
String lastname=br.readLine();

System.out.print("Enter mobile phone: ");
String mobile=br.readLine();

System.out.print("Enter land line phone: ");
String landline=br.readLine();

out.write(name+" "+lastname+" "+mobile+" "+landline);
out.newLine();

System.out.print("If you want to enter another contact enter yes.\nIf you don't,
enter no: ");
String answer=br.readLine();
if(!answer.equals("yes"))
{
prostheseEpafi=false;
}
}

out.close();
}

catch (IOException e) {
    e.printStackTrace();
    System.out.println("Output error");
       }
    }
}
```

## Εγγραφή/Ανάγνωση αντικειμένου σε μια ροή (Serialization)

Επιπλέον των χαρακτήρων ή άλλων πρωταρχικών (int, double, float, byte, …) δεδομένων που μπορούμε να γράψουμε ή να διαβάσουμε σε/από αρχεία ή οποιοδήποτε άλλου τύπου stream, η Java επιτρέπει την εγγραφή και ανάγνωση αντικειμένων. Για να το επιτύχουμε αυτό θα πρέπει να χρησιμοποιήσουμε την κλάση *ObjectOutputStream* για να εγγράψουμε ένα αντικείμενο της Java σε ένα *OutputStream*. Ομοίως θα χρησιμοποιήσουμε την κλάση *ObjectInputStream* για να διαβάσουμε ένα αντικείμενο από κάποιο *InputStream*, που έχει γραφεί προηγουμένως με την *ObjectOutputStream*.

Η κλάση *ObjectOutputStream* παρέχει μεθόδους για την εγγραφή τόσο βασικών τύπων όσο και αντικειμένων με τη μορφή δυαδικής αναπαράστασης σε ένα ρεύμα εξόδου. Διαθέτει για κάθε βασικό τύπο μία μέθοδο της μορφής write*X*(*X*), δηλαδή writeInt(int), writeBoolean(boolean). Στη κλάση αυτή ορίζεται και η μέθοδος ***void writeObject (Object obj)***, η οποία παίρνει ως παράμετρο το αντικείμενο που θέλουμε να γράψουμε στο stream της εξόδου.
Αντίστοιχα η κλάση *ObjectInputStream* παρέχει τις αντίστοιχες μεθόδους για την ανάγνωση τόσο βασικών τύπων όσο και αντικειμένων με τη μορφή δυαδικής αναπαράστασης από ένα ρεύμα εισόδου. Οι μέθοδοι αυτές έχουν για τους βασικούς τύπους τη μορφή read*X*(*X*), δηλαδή readInt(), readByte(). Επίσης ορίζεται η μέθοδος ***Object readObject()***, η οποία επιστρέφει το αντικείμενο που είχαμε γράψει προηγουμένως στο stream. Όπως βλέπουμε η μέθοδος επιστρέφει αντικείμενο τύπου Object και επομένως το αντικείμενο αυτό θα πρέπει να το κάνουμε cast στον πραγματικό τύπο του αντικειμένου.
Οποιοδήποτε αντικείμενο μπορεί να περαστεί ως όρισμα στην μέθοδο writeObject() αρκεί η κλάση να υλοποιεί το interface java.io.Serializable. Το συγκεκριμένο interface δεν περιέχει κάποια μέθοδο, αλλά λειτουργεί ως ένδειξη (marker) ότι τα αντικείμενα κλάσεων που το υλοποιούν μπορούν να εγγραφούν/αναγνωστούν σε streams. Επίσης μπορούν να χρησιμοποιηθούν πίνακες ή String, που υλοποιούν το συγκεκριμένο interface. Μια Serializable κλάση πρέπει να περιέχει μόνο πρωταρχικούς τύπους δεδομένων ή αναφορές σε αντικείμενα κλάσεων που υλοποιούν το συγκεκριμένο interface. Τα μόνα πεδία που εξαιρούνται από το serialization είναι τα static και τα transient.

Στο ακόλουθο σχήμα απεικονίζεται η διαδικασία εγγραφής και ανάγνωσης αντικειμένων αλλά και πρωταρχικών τύπων προς/από ένα δυαδικό αρχείο. Είναι φανερό ποιες είναι οι κλάσεις που θα πρέπει να συνεργαστούν για αυτή τη λειτουργία:




Σε ένα stream μπορούμε να εγγράψουμε πολλαπλά αντικείμενα, κατά την ανάγνωση όμως θα πρέπει να τα διαβάσουμε με τη σειρά που τα γράψαμε και να τα κάνουμε κατάλληλα casting.

### Παράδειγμα

```
import java.io.*;
public class MainOut { // εγγραφή αντικειμένων σε αρχείο
```

```
public static void main(String[] args) {
     ObjectOutputStream out;
     int array[] = {1,4,5,8};
      try{
       out=new ObjectOutputStream(new FileOutputStream("object.txt"));

       out.writeObject(array);
       out.writeObject("abc");
       out.writeObject(new PhoneBook("Nikos", "2273012345"));
       out.writeObject(new PhoneBook("Mixalis", "2273054321"));
       out.flush();
       out.close();

       System.out.println("Object written to file");
     }
     catch (FileNotFoundException ex) {
         System.out.println("Error with specified file") ;
         ex.printStackTrace();
        }
     catch (IOException ex) {
            System.out.println("Error with I/O processes") ;
            ex.printStackTrace();
        }
    }
}
```

---

```
import java.io.Serializable;

public class PhoneBook implements Serializable{ //serializable κλάση
  private String name;
  private String phone;

  public PhoneBook(String n, String p){
    this.name = n;
    this.phone = p;
  }

  public String toString (){
    return "the subscriber "+ this.name+ " has the number :"+this.phone;

  }
}
```

## 2η εργαστηριακή άσκηση – Αρχεία αντικειμένων

Υλοποιήστε main κλάση που θα διαβάζει τα αντικείμενα από το αρχείο που δημιουργήσαμε στο προηγούμενο παράδειγμα και θα εμφανίζει τα περιεχόμενα του στην οθόνη.

## Απάντηση

```
import java.io.*;
public class MainIn { // ανάγνωση αντικειμένων από αρχείο

public static void main(String[] args) {
     ObjectInputStream in=null;
     int array1[]=null;
     String str=null;
```

```
      PhoneBook book;

       try{
        in = new ObjectInputStream(new FileInputStream("object.txt"));

         array1 = (int []) in.readObject();
         str = (String) in.readObject();

         while( (book = (PhoneBook) in.readObject()) != null){
              System.out.println(book);
         }
      }

      catch (EOFException ex) {
             System.out.println("End of file reached.");
      }
      catch (ClassNotFoundException ex) {
         System.out.println("Error casting") ;
          ex.printStackTrace();
         }

      catch (FileNotFoundException ex) {
          System.out.println("Error specified file does not exist") ;
          ex.printStackTrace();
         }

      catch (IOException ex) {
             System.out.println("Error with I/O processes") ;
             ex.printStackTrace();
         }

      finally {
          try{
             in.close();
          }
          catch(IOException ex){
              System.out.println("Another IOException during the closing");
              ex.printStackTrace();
          }
      }

 System.out.println("Array contains :");
 for(int i=0;i<array1.length; i++){
    System.out.print(array1[i]+"  ");
  }
 System.out.println("\nString contains :" +str);
}
}
```