Αντικειμενοστρεφής Προγραμματισμός
Διάλεξη – 11 : ΑΡΧΕΙΑ ΚΑΙ ΧΕΙΡΙΣΜΟΣ ΣΦΑΛΜΑΤΩΝ

Κατηγορίες

- Διαχείριση αρχείων
- Η κλάση File για τη διαχείριση αρχείων
- Δυαδικά ρεύματα
- Εγγραφή και ανάγνωση δυαδικών αρχείων
- Αρχεία Χαρακτήρων
- Εγγραφή και ανάγνωση σε/από αρχεία χαρακτήρων
- Serializable αρχεία
- Εγγραφή και ανάγνωση σε/από Serializable αρχεία

Διαχείριση Αρχείων

- Το πακέτο **java.io** περιέχει πολλές κλάσεις που βοηθούν τον προγραμματιστή σε διεργασίες σχετικές με την είσοδο/έξοδο (I/O).
- Οι λειτουργίες (I/O) στη Java γίνονται με τη χρήση ρευμάτων (streams) έτσι ώστε κάθε ρεύμα να προσφέρει σειριακή πρόσβαση στα δεδομένα.
- Τα ρεύματα χωρίζονται σε δύο βασικές κατηγορίες, σε δυαδικά ρεύματα (binary streams) και σε ρεύματα χαρακτήρων (character streams). Επίσης, ανάλογα με την κατεύθυνσή του ένα ρεύμα μπορεί να είναι ρεύμα εισόδου (input stream) ή ρεύμα εξόδου (output stream). Το πρώτο μπορεί να 'διαβάσει' μία ακολουθία δεδομένων ενώ το δεύτερο να 'γράψει' μία ακολουθία δεδομένων.

Οντότητες που μπορούν να χρησιμοποιηθούν ως ρεύματα εισόδου ή ως ρεύματα εξόδου

- Ένα αρχείο
- Ένας πίνακας χαρακτήρων ή πίνακας τύπου **byte**
- Ένα pipe, το οποίο αποτελεί μηχανισμό επικοινωνίας δεδομένων μεταξύ προγραμμάτων
- Μία δικτυακή σύνδεση

Η κλάση File

- Μία από τις βασικότερες κλάσεις του πακέτου **java.io** είναι η **File** που παρέχει έναν μηχανισμό που αναπαριστά το υποκείμενο σύστημα αρχείων (file system). Άρα αντικείμενα της κλάσης File δουλεύουν και σε Windows και σε Linux
- Ένα αντικείμενο τύπου **File** αναπαριστά τη διαδρομή (path) ενός αρχείου ή φακέλου στο υποκείμενο σύστημα. Η συγκεκριμένη κλάση δηλαδή μπορεί να χρησιμοποιηθεί από μία εφαρμογή για την διαχείριση των φυσικών αρχείων, όχι όμως των περιεχομένων τους.
- Η κλάση **File** παρέχει σταθερές και μεθόδους που βοηθούν τη διαχείριση αρχείων σε διαφορετικές πλατφόρμες, δίνοντας έναν κοινό μηχανισμό διαχείρισης για όλες τις πλατφόρμες. Τα διαφορετικά σύμβολα που χρησιμοποιούνται για την αναπαράσταση διαδρομών στο file system (π.χ. :, \, /) κάθε ενός εκ των δημοφιλέστερων λειτουργικών συστημάτων αποθηκεύεται στην κλάση με τη μορφή μιας σταθεράς.

Μέθοδοι της κλάσης File

- Πληροφορίες για αρχεία/φακέλους

getName()	getPath()
getAbsolutePath()	getCanonicalPath()
getParent()	isAbsolute()
list()	listFiles()
lastModified()	length()
isFile()	isDirectory()

- Διαχείριση δικαιωμάτων
  - **setReadable(boolean), setWritable(boolean), setExecutable(boolean)**
  - **SecurityException() αν το δικαίωμα δε μπορεί να αλλάξει**
  - **canWrite(), canRead(), canExecute():**
- Δημιουργία μετονομασία και διαγραφή αρχείων

createNewFile()	mkdir()
renameTo(File)	delete()

Παράδειγμα χρήσης της κλάσης File (1/2)

```
import java.io.File;
import java.io.IOException;
public class DirectoryListing {
        public static void main(String[] args) {
                if(args.length == 0)
                        System.err.println("Please specify a directory name");
                else {
                        File entry = new File(args[0]);
                        listDirectory(entry);
                }
        }
}
```

Παράδειγμα χρήσης της κλάσης File (2/2)

```
public static void listDirectory(File dir){
    try {
            if(!dir.exists() || dir.isFile())
                    System.out.println("Directory " + dir.getName() + " not found");
            else {
                    String[] structure = dir.list();
                    File curFile = null;
                    for (String name : structure){
                            curFile = new File(dir.getPath(), name);
                            System.out.println(curFile.getCanonicalPath());
                    }
            }
    }
    catch(IOException e){
            e.printStackTrace();
    }
}
```

Δυαδικά Ρεύματα (Byte Streams) (1/4)

- Οι αφηρημένες κλάσεις **InputStream** και **OutputStream** είναι οι βασικές που χειρίζονται την ανάγνωση και την εγγραφή bytes σε δυαδικά ρεύματα (binary ή byte streams), ορίζοντας τη βασική συμπεριφορά όλων των κλάσεων αυτών.
- Η κλάση **InputStream** αναπαριστά ένα δυαδικό ρεύμα εισόδου και βασική της μέθοδος είναι η **read()**, που υπάρχει σε 3 εκδόσεις.
  1. **int read() throws IOException**
  2. **int read(byte[] b) throws IOException**
  3. **int read(byte[] b, int off, int len) throws IOException**
- Η μέθοδος αυτή διαβάζει bytes από ένα ρεύμα και τα επιστρέφει με τη μορφή int, με την τιμή του byte που διαβάστηκε να βρίσκεται στα 8 λιγότερο σημαντικά bits του int που επιστρέφεται. Όταν η read() φτάσει στο τέλος του ρεύματος θα επιστρέψει την τιμή -1.

Δυαδικά Ρεύματα (Byte Streams) (2/4)

- Η κλάση **OutputStream** με τη σειρά της αναπαριστά ένα δυαδικό ρεύμα εξόδου και βασική της μέθοδος είναι η **write()** που επίσης διατίθεται σε τρεις εκδόσεις.
  1. **void write(int b) throws IOException**
  2. **void write(byte[] b) throws IOException**
  3. **void write(byte[] b, int off, int len) throws IOException**
- Η πρώτη έκδοση λαμβάνει ως παράμετρο ένα **int** του οποίου όμως διαβάζει τα 8 λιγότερο σημαντικά bits τα οποία και γράφει ως byte.
- Ένα ρεύμα θα πρέπει να κλείσει όταν δεν το χρειαζόμαστε άλλο και έτσι και οι δύο κλάσεις (**InputStream, OutputStream**)υποστηρίζουν τις μεθόδους **close()** και **flush()**. Η μεν πρώτη κλείνει το ρεύμα, ενώ η **flush()** αδειάζει όλα τα περιεχόμενά του. Κλείνοντας ένα ρεύμα με την **close()** ταυτόχρονα αδειάζει και το ρεύμα.

Δυαδικά Ρεύματα (Byte Streams) (3/4)

- Οι συμπαγείς κλάσεις που κληρονομούν από την **InputStream** είναι οι ακόλουθες:
  - **FileInputStream**: Διαβάζει δεδομένα με τη μορφή bytes από ένα αρχείο.
  - **FilterInputStream**: Προσφέρει εξειδικευμένες λειτουργίες για την ανάγνωση δεδομένων από ένα ρεύμα, συνήθως μέσω της υποκλάσης της **BufferedInputStream**.
  - **DataInputStream**: Διαβάζει δυαδικές αναπαραστάσεις βασικών τύπων δεδομένων από ένα ρεύμα εισόδου.
  - **ObjectInputStream**: Διαβάζει δυαδικές αναπαραστάσεις αντικειμένων Java και βασικών τύπων από ένα δυαδικό ρεύμα.

Δυαδικά Ρεύματα (Byte Streams) (4/4)

- Οι συμπαγείς κλάσεις που κληρονομούν από την **OutputStream** είναι οι εξής:
  - **FileOutputStream**: Γράφει δεδομένα με τη μορφή bytes σε ένα αρχείο.
  - **FilterOutputStream**: Προσφέρει εξειδικευμένες λειτουργίες για την εγγραφή δεδομένων από ένα ρεύμα, συνήθως μέσω της υποκλάσης της **BufferedOutputStream**.
  - **DataOutputStream**: Γράφει δυαδικές αναπαραστάσεις βασικών τύπων δεδομένων σε ένα ρεύμα εξόδου.
  - **ObjectOutputStream**: Γράφει δυαδικές αναπαραστάσεις αντικειμένων Java και βασικών τύπων σε ένα δυαδικό ρεύμα.

Διαδικασία εγγραφής σε binary αρχείο

- Για να γράψουμε δεδομένα βασικών τύπων σαν bytes σε ένα binary αρχείο εκτελούμε τη διαδικασία που αποτελείται από τα εξής βήματα:
  - Δημιουργούμε ένα αντικείμενο τύπου **FileOutputStream**, π.χ. FileOutputStream outFile = **new** FileOutputStream("binary.dat");
  - Δημιουργούμε ένα αντικείμενο τύπου **DataOutputStream** και το συνδέουμε με το **FileOutputStream** του πρώτου βήματος: DataOutputStream outputStream = **new** DataOutputStream(outFile);
  - Καλούμε τις κατάλληλες **write____(__)** μεθόδους της κλάσης **DataOutputStream**, ανάλογα με τον τύπο που θέλουμε να γράψουμε, π.χ. outputStream.writeInt(3523);
  - Κλείνουμε το ρεύμα όταν δεν το χρειαζόμαστε άλλο: **outputStream.close();**

Διαδικασία ανάγνωσης από binary αρχείο

- Για την ανάγνωση δεδομένων βασικών τύπων από ένα binary αρχείο χρησιμοποιούμε τα ίδια ακριβώς βήματα με τη διαφορά πως χρησιμοποιούμε τις αντίστοιχες κλάσεις ρευμάτων εισόδου, δηλαδή την
  - κλάση **FileInputStream** αντί της **FileOuputStream** και την **DataInputStream** αντί της **DataOutputStream** και φυσικά χρησιμοποιούμε τις αντίστοιχες μεθόδους ανάγνωσης π.χ. **readInt()**, **readBoolean()** κλπ.
- Θα δούμε παραδείγματα κώδικα εγγραφής και ανάγνωσης σε/από binary αρχεία

Ανάγνωση και εγγραφή από/σε binary αρχείο- Παράδειγμα (1/2)

```
import java.io.*;
class ByteBinaryFiles
   public static void main(String[] args) {
         // Write a byte to disk
         try {
             FileOutputStrearn f =new FileOutputStream
                                        ("myTextFile.dat");
             BufferedOutputStream b = new
                                        BufferedOutputStream(f);
              int data = 20;
              b.write(data);
              b.close();
         }catch(IOException e) {
```

Συνεχίζεται.....

Ανάγνωση και εγγραφή από/σε binary αρχείο- Παράδειγμα (2/2)

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

Δυαδικά Αρχεία που δεν περιέχουν ΜΟΝΟ bytes δεδομένων

- Σε δυαδικά αρχεία μπορούμε να αποθηκεύσουμε διάφορους τύπους δεδομένων (π.χ. **int, float, double, char**) εκτός από **bytes**.
- Η διαδικασία είναι παρόμοια με την προηγούμενη μόνο που χρειάζεται να χρησιμοποιήσουμε επιπλέον τα αντικείμενα **DataOutputStream** και **DataInputStream** διότι έχουν τις κατάλληλες μεθόδους (π.χ. **writeInt, readInt, writeDouble, readDouble**)
- Το τέλος του αρχείου μπορούμε να το εντοπίσουμε με τη χρήση της εξαίρεσης **EOFException**
- Στο παρακάτω παράδειγμα η ανάγνωση του δυαδικού αρχείου γίνεται σε **loop** και δεν σταματά μέχρι να φθάσουμε στο τέλος οπότε γίνεται **throw** το **EOFException**

Δυαδικά Αρχεία που δεν περιέχουν ΜΟΝΟ bytes δεδομένων- Παράδειγμα (1/2)

```
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 {
               FileOutputStream f = new FileOutputStream("myFile.dat");
               BufferedOutputStream b =new BufferedOutputStream(f);
               DataOutputStream d =new DataOutputStream(b);
               for(int i=0; i<100; i++) {
                        double dataDouble =generator.nextDouble ();
                        d.writeDouble(dataDouble);
               }
               d.close();
            }catch(IOException e){
                System.out.println(e.getMessage());
            }
```

Δυαδικά Αρχεία που δεν περιέχουν MONO bytes δεδομένων- Παράδειγμα (2/2)

```
// Read the saved set of numbers from disk
try {
    FileinputStream f = new FileinputStream("myFil.e.dat");
    BufferedinputStream b = new BufferedinputStream(f);
    DatainputStream d = new DatainputStream(b);
    // Nested try-catch detecting end of file
    try {
        int c = 1;
        for (;;) {    // Infinite loop
            double dataDouble;
            dataDoubl.e =  d.readDouble();
            System.out.println(c + " " + dataDouble);
            c++;
    }catch (EOFException eof) {
        d.cl.ose();
    }
} catch(IOException e) {
        System.out.println (e.getMessage());
}
}}
```

Ρεύματα Χαρακτήρων (Character Streams)

- Για τα ρεύματα χαρακτήρων (character streams) οι αφηρημένες κλάσεις **Reader** και **Writer** ορίζουν τη βασική συμπεριφορά για την ανάγνωση και εγγραφή χαρακτήρων συγκεκριμένης κωδικοποίησης (encoding) από και προς ένα ρεύμα εισόδου ή εξόδου αντίστοιχα.
- Η εξ' ορισμού κωδικοποίηση που χρησιμοποιείται είναι η Unicode.
- Παρέχουν αντίστοιχες μεθόδους με αυτές των κλάσεων που χειρίζονται τα binary streams, δηλαδή την **read()**, την **write()**, την **flush()** και την **close()**, ενώ η **Reader** υποστηρίζει μία επιπλέον μέθοδο, τη **skip(long)** που παραλείπει τόσους χαρακτήρες όσους ορίζει η παράμετρος τύπου **long** που λαμβάνει.

Ρεύματα Χαρακτήρων (Character Streams) Κληρονομικότητα της Reader

- Οι συμπαγείς κλάσεις που κληρονομούν από την **Reader** είναι οι εξής:
  - **BufferedReader**: Κλάση που μπορεί να χρησιμοποιηθεί για την ανάγνωση χαρακτήρων από τον υποκείμενο reader (π.χ. **InputStreamReader**) κάνοντας χρήση buffer. Στο παράδειγμα που θα δείτε παρακάτω γίνεται χρήση ενός buffered reader.
  - **InputStreamReader**: Διαβάζει χαρακτήρες από ένα δυαδικό ρεύμα εισόδου χρησιμοποιώντας την **default** κωδικοποίηση.
  - **FileReader**: Διαβάζει χαρακτήρες από ένα αρχείο χρησιμοποιώντας την default κωδικοποίηση.

Ρεύματα Χαρακτήρων (Character Streams) Κληρονομικότητα της Writer

- Οι συμπαγείς κλάσεις που κληρονομούν από την **Writer** είναι οι:
  - **BufferedWriter**: Κλάση που εκτελεί εγγραφή χαρακτήρων (π.χ. **OutputStreamWriter**) μέσω buffer.
  - **OutputStreamWriter:** Γράφει χαρακτήρες σε ένα binary stream με default κωδικοποίηση.
  - **FileWriter**: Γράφει χαρακτήρες σε ένα αρχείο χρησιμοποιώντας την default κωδικοποίηση.
  - **PrintWriter**: Μία κλάση που επιτρέπει την εγγραφή βασικών τύπων αλλά και αντικειμένων υπό μορφή κειμένου σε ένα υποκείμενο ρεύμα εξόδου ή έναν writer.
- Η PrintWriter είναι χρήσιμη: Παρέχει 2 υπερφορτωμένες μεθόδους, τις **print(X)** και **println(X)** οι οποίες γράφουν με τη μορφή κειμένου στην έξοδο την παράμετρο X

Ρεύματα Χαρακτήρων (Character Streams) Παράδειγμα Writing

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

Ρεύματα Χαρακτήρων (Character Streams) Παράδειγμα Reading

```
import java.io.*;
class ReadTextFile {
        public static void main(String[ ] args)
        {
            try {
                    FileReader 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());
                }
        }
}
```

Μπορούμε να γράψουμε αντικείμενα σε αρχεία?

- Για να μπορεί ένα αντικείμενο μίας κλάσης να γράφεται σε αρχείο, πρέπει αυτή η κλάση να υλοποιεί το Interface Serializable
- Η διεπαφή αυτή υπάρχει στο πακέτο java.io και ο ρόλος της είναι απλά για να υποδεικνύει ότι ένα αντικείμενο που υλοποιεί τη διεπαφή αυτή μπορεί να εγγραφεί σε αρχείο.
- ΠΑΡΑΔΕΙΓΜΑ

```
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;
    }
}
```

Serialization (1/4)

- Με τον όρο αυτόν αναφερόμαστε στη διαδικασία μετατροπής ενός αντικειμένου σε μία ακολουθία από bytes ώστε να μπορεί να γραφτεί σε ένα αρχείο ή μία δικτυακή σύνδεση.
- Η αντίστροφη διαδικασία, δηλαδή η επανασύνθεση του αντικειμένου ονομάζεται deserialization. Έτσι λοιπόν, στην περίπτωση που έχουμε ένα αρχείο, το deserialization περιλαμβάνει την επανασύνθεση του αντικειμένου από την ανάγνωση των δεδομένων του αρχείου, ενώ στην περίπτωση που έχουμε μία δικτυακή σύνδεση το deserialization θα αφορούσε στην επανασύνθεση του αντικειμένου διαβάζοντας τα δεδομένα που καταφθάνουν με τη μορφή bytes στο άλλο άκρο της σύνδεσης.

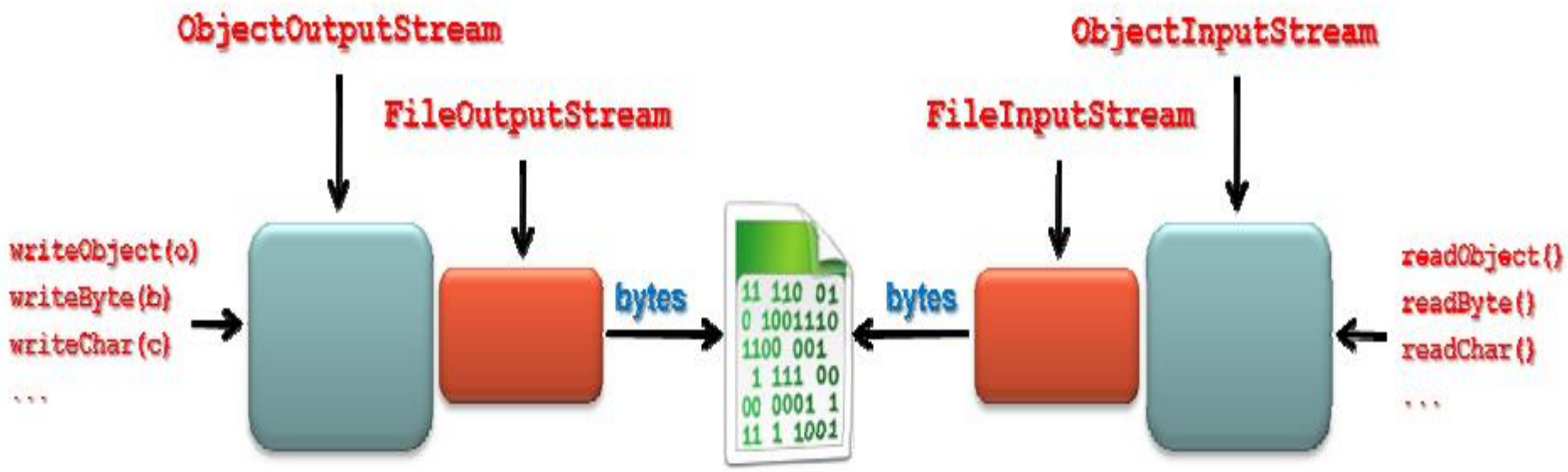
Serialization (2/4)

- Το serialization χρησιμοποιείται από όλα σχεδόν τα προγράμματα κατά την αποθήκευση δεδομένων στον σκληρό δίσκο. Η κατάσταση που βρίσκεται το πρόγραμμα μία δεδομένη χρονική στιγμή μπορεί να αποθηκευτεί και σε μεταγενέστερο χρόνο διαβάζοντας τα συγκεκριμένα δεδομένα από το αποθηκευτικό μέσο και επανασυνθέτοντας τα ίδια ακριβώς αντικείμενα να επανέλθει στην ίδια κατάσταση που βρισκόταν κατά τη στιγμή της αποθήκευσης.
- Οι κλάσεις **ObjectOutputStream ΚΑΙ ObjectInputStream** (τις είδαμε όταν μιλούσαμε για τα δυαδικά ρεύματα) παρέχουν τις απαραίτητες διευκολύνσεις στους προγραμματιστές που επιθυμούν να χρησιμοποιήσουν serialization στον κώδικά τους

Serialization (3/4)

- 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 **writeX(X)**, π.χ. **writeBoolean(boolean), writeChar(char)** κλπ και για τα αντικείμενα τη μέθοδο **writeObject(Object).**
- Η **ObjectInputStream** παρέχει τις αντίστοιχες μεθόδους για 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 **readX()**, π.χ. **readInt()**, **readByte()** κλπ ενώ για τα αντικείμενα υπάρχει η μέθοδος **readObject().**

Serialization (4/4)



Διαδικασία εγγραφής δεδομένων σε αρχείο με Serialization

- Για την αποθήκευση δεδομένων σε ένα binary αρχείο συνεργάζονται οι κλάσεις **FileOutputStream** και **ObjectOutputStream.** Δημιουργούμε τα αντίστοιχα αντικείμενα και τα συνδέουμε μεταξύ τους σύμφωνα με τον παρακάτω κώδικα:

```
FileOutputStream outFile = new FileOutputStream("data.dat");
ObjectOutputStream os = new ObjectOutputStream(outFile);
```

Στη συνέχεια θα μπορούσαμε να καλέσουμε τις μεθόδους της **ObjectOutputStream** μέσω του **os** για να αποθηκεύσουμε δεδομένα στο αρχείο, είτε βασικούς τύπους είτε αντικείμενα, π.χ.

```
os.writeBoolean(true);
os.writeObject(myObject);
```

Διαδικασία ανάγνωσης δεδομένων από αρχείο με Serialization

- Η διαδικασία ανάγνωσης δεδομένων είναι πανομοιότυπη, με τη διαφορά πως εδώ χρησιμοποιούνται οι κλάσεις **FileInputStream** και **ObjectInputStream** και οι μέθοδοι **readX()**, π.χ.:

```
FileInputStream inFile =new FileInputStream("data.dat");
ObjectInputStream is =new ObjectInputStream(inFile);
boolean b = is.readBoolean();
```

- Για να μπορεί μια οποιαδήποτε δική μας κλάση να γίνει serialized, βασική προϋπόθεση είναι να υλοποιεί το interface **Serializable**. Έχοντας μία κλάση που υλοποιεί το **Serializable** interface, μπορεί να γίνει εύκολα serialized κάνοντας χρήση της μεθόδου **writeObject()** της **ObjectOutputStream**.

Παράδειγμα εγγραφής/ανάγνωσης σε αρχείο με Serialization

```
import java.io.*;
public class ObjectFileDemo {
        public static void main(String[ ] args) {
        try {
            ObjectOutputStream oos = new ObjectOutputStream (new
                    FileOutputStream("TestObjectFile.dat"));
            Date d = new Date(3,2,2014);
            oos.writeObject(d);
            oos.close();
        } catch (IOException e) {
                    System.out.println (e);
        }
        try {
            ObjectinputStream oos = new ObjectinputStream (
                    new FileinputStream("TestObjectFile.dat"));
            Date d;
            d = (Date)oos.readObject();
            System.out.println (d);
           oos.close();
        } catch (Exception e) {
                    System.out.println(e);
        }
}
```