Αρχεία

Γιώργος Θάνος
gthanos@uth.gr
Γραφείο: Γ5/8, 3ος όροφος
Γκλαβάνη 37

Αρχεία

Οι κλάσεις που συνδέονται με διάβασμα και γράψιμο σε αρχείο βρίσκονται κατά κανόνα στον πακέτο java.io. Εξαίρεση αποτελεί η κλάση Scanner που βρίσκεται μέσα στο πακέτο java.util.

Η κλάση java.io.File

Η κλάση **`java.io.File`** απεικονίζει ένα αρχείο ή κατάλογο από το λειτουργικό σας σύστημα.

Μέσω της κλάσης **`File`** μπορείτε να έχετε πρόσβαση στα αρχεία του συστήματος σας, να εξετάσετε κατά πόσο ένα αρχείο είναι directory ή απλό αρχείο, να δείτε τα περιεχόμενα ενός directory, να εξετάσετε εάν ένα αρχείο είναι εκτελέσιμο ή όχι κ.α.

Προκειμένου να φτιάξετε ένα αντικείμενο της κλάσης **`File`** αρκεί να χρησιμοποιήσετε τον κατασκευαστή της κλάσης ο οποίος λαμβάνει ως όρισμα ένα **`String`** που περιέχει το μονοπάτι του αρχείου.

```
public File(String pathname);
```

Αν θέλετε να δημιουργήσετε ένα αντικείμενο τύπου **`File`** για τον τρέχον κατάλογο αρκεί να γράψετε:

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

Σημείωση1: Στο λειτουργικό σύστημα Windows τα paths διαχωρίζονται από backslash (‘\’). Εφόσον εισάγετε path με backslash θα πρέπει να τα κάνετε escape όπως παρακάτω:

```
File favoritSong = new
File("C:\\Users\\<YourUsername>\\Music\\myF
avoritSong.mp3");
```

Σημείωση2: Όταν δημιουργείται ένα αντικείμενο τύπου **`File`**, αυτό μπορεί να μην αντικατοπτρίζει ένα υπάρχον αρχείο στο *filesystem*. Ο έλεγχος γίνεται μέσω της μεθόδου **exists()**.

Μέθοδοι της κλάσης File (½)

`public boolean canExecute()`: Ελέγχει εάν το αρχείο είναι εκτελέσιμο ή όχι.

`public boolean canRead()`: Ελέγχει εάν μπορούμε να ανοίξουμε το αρχείο για διάβασμα.

`public boolean canWrite()`: Ελέγχει εάν μπορούμε να ανοίξουμε το αρχείο για γράψιμο.

`public boolean createNewFile()`: Δημιουργεί ένα νέο κενό αρχείο στο filesystem, εφόσον το αντικείμενο **`File`** δεν αντιστοιχεί σε υπαρκτό αρχείο.

`public boolean delete()`: Επιχειρεί να διαγράψει το αρχείο ή κατάλογο. Επιστρέφει true εάν επέτυχε η διαγραφή, διαφορετικά **false**.

`public boolean exists()`: Ελέγχει εάν το συγκεκριμένο αντικείμενο αντικατοπτρίζει ένα πραγματικό αρχείο ή κατάλογο στο filesystem.

`public String getAbsolutePath()` : Επιστρέφει το απόλυτο path για ένα υπαρκτό αρχείο ή κατάλογο. Η συγκεκριμένη μέθοδος έχει νόημα εάν έχετε δημιουργήσει το αντικείμενο χρησιμοποιώντας ένα σχετικό path αντί για απόλυτο.

`public String getName()`: Επιστρέφει μόνο το όνομα του αρχείου ή καταλόγου του.

Μέθοδοι της κλάσης File (2/2)

`public boolean isDirectory()`: Ελέγχει εάν το συγκεκριμένο αντικείμενο αντιπροσωπεύει ένα directory.

`public boolean isFile()`: Ελέγχει ένα το συγκεκριμένο αντικείμενο είναι απλό αρχείο (όχι directory, , όχι ειδικό αρχείο).

`public long length()`: Επιστρέφει το μήκος του αρχείου.

`public String[] list()`: Επιστρέφει ένα πίνακα από Strings που περιέχει τα ονόματα των περιεχόμενων αρχείων. Η συγκεκριμένη μέθοδος έχει νόημα μόνο εάν το αντικείμενο τύπου **`File`** αντιπροσωπεύει ένα directory. Διαφορετικά επιστρέφει **null**.

`public File[] listFiles()`: Επιστρέφει ένα πίνακα από Files που περιέχει τα αντικείμενα File για τα περιεχόμενα αρχεία. Η συγκεκριμένη μέθοδος έχει νόημα μόνο εάν το αντικείμενο File αντιπροσωπεύει ένα directory. Διαφορετικά επιστρέφει **null**.

`public boolean mkdir()`: Δημιουργεί ένα νέο κενό κατάλογο στο filesystem, εφόσον δεν υπάρχει. Επιστρέφει **true** εάν ο κατάλογος δημιουργηθεί επιτυχώς, διαφορετικά επιστρέφει **false**.

Παράδειγμα 1ο

Με χρήση της κλάσης **java.io.File** δημιουργήστε ένα πρόγραμμα που εκτυπώνει μία λίστα με τα περιεχόμενα των αρχείων του τρέχοντος καταλόγου

```
import java.io.File;


public class CurrentDirList {
  public static void main(String []args) {
    File currentDir = new File(".");
    File files[] = currentDir.listFiles();
    for(File f : files) {
      System.out.println(f.getName());
    }
  }
}
```

Παράδειγμα 2ο

Με χρήση της κλάσης `java.io.File` δημιουργήστε ένα πρόγραμμα που που λαμβάνει το όνομα ενός αρχείου **από την γραμμή εντολών** και εάν δεν υπάρχει το δημιουργεί στο τρέχοντα κατάλογο.

```
import java.io.File;
public class CreateEmptyFile {
  public static void main(String args[]) {
    if( args.length == 0 )
      return;
    File f = new File("./"+args[0]);
    if( !f.exists() ) {
      try {
        f.createNewFile();
      } catch( Exception ex ) {
        ex.printStackTrace();
      }
    }
}
```

Διάβασμα από αρχεία κειμένου

Διαβάζοντας γραμμή - γραμμή με χρήση της κλάσης BufferedReader

```
import java.io.*;
public class CopyLines {
  public static void main(String[] args) throws IOException {
    String filename = "xanadu.txt";
      if(args.length > 0)
        filename = args[0];
    try (BufferedReader inputStream = new BufferedReader(new FileReader(filename));
         PrintWriter outputStream = new PrintWriter(new FileWriter("__"+filename)) ) {
      String str;
      while ((str = inputStream.readLine()) != null) {
        outputStream.println(str+"\n");
      }
    }
  }
}
```

Μπορείτε να διαβάσετε γραμμή-γραμμή και με χρήση της κλάσης java.util.Scanner που θα δούμε παρακάτω.

Διαβάζοντας με χρήση της κλάσης java.util.Scanner

Κατασκευαστές της κλάσης java.util.Scanner

`public Scanner (File file)`

Δημιουργεί ένα νέο αντικείμενο της κλάσης Scanner προκειμένου να διαβάσουμε από το αρχείο file.

`public Scanner (InputStream source):`

Δημιουργεί ένα νέο αντικείμενο της κλάσης Scanner προκειμένου να διαβάσουμε από ένα java.io.InputStream source. Ο συγκεκριμένος κατασκευαστής είναι χρήσιμος εάν θέλουμε να διαβάσουμε από την κονσόλα ή από ένα ήδη ανοικτό **InputStream**.

Οι μέθοδοι της κλάσης java.util.Scanner

public void close(): Κλείνει το Scanner. Μετά την κλήση της close δεν μπορείτε να διαβάσετε τίποτα.

public boolean hasNext(): Επιστρέφει true εάν υπάρχει κάτι για διάβασμα.

public String next(): Επιστρέφει το επόμενο String για διάβασμα. Το επόμενο String οριοθετείται μέχρι να βρεθεί κενό ή χαρακτήρας αλλαγής γραμμής.

public String nextLine(): Επιστρέφει την επόμενη γραμμή. Το τέλος της γραμμής οριοθετείται από την ύπαρξη του χαρακτήρα αλλαγής γραμμής newline \n ή carriege return \r ή συνδυασμό και των δύο.

public boolean hasNextInt(): Επιστρέφει true εάν υπάρχει κάτι για διάβασμα το οποίο μπορεί να μεταφραστεί ως ακέραιος. Εάν επιστρέψει true, διαβάζουμε με την μέθοδο nextInt().

public int nextInt(): Διαβάζει και επιστρέφει έναν ακέραιο αριθμό.

public boolean hasNextDouble(): Επιστρέφει true εάν υπάρχει κάτι για διάβασμα το οποίο μπορεί να μεταφραστεί ως double. Εάν επιστρέψει true, διαβάζουμε με την μέθοδο nextDouble().

public int nextDouble(): Διαβάζει και επιστρέφει έναν αριθμό double.

public boolean hasNextBoolean(): Επιστρέφει true εάν υπάρχει κάτι για διάβασμα το οποίο μπορεί να μεταφραστεί ως boolean. Εάν επιστρέψει true, διαβάζουμε με την μέθοδο nextBoolean().

public int nextBoolean(): Διαβάζει και επιστρέφει μία τιμή boolean.

Παράδειγμα ανάγνωσης με χρήση της Scanner

Θέλετε να διαβάσετε το αρχείο **telephone-list.txt** που έχει την παρακάτω μορφή CSV:

```
John Smith, 6944854544
Robert Parson, 6978456123
Nick Carlson, 6982147852
Barbara Miller, 6978852456
Max Taylor, 6936785412
```

και να αποθηκεύσετε τα δεδομένα του σε ένα πίνακα του τύπου **TelephoneNumber**.

```
public class TelephoneNumber {
  private String name, number;
  public TelephoneNumber(String newName, String newNumber) {
    name = newName;   number = newNumber;
  }
  public String getName() { return name; }
  public String getNumber() { return number; }
  public void setName(String newName) { name = newName; }
  public void setNumber(String newNumber) { number = newNumber;
}
  public String toString() {
    return "Name: "+name+", Number: "+number;
  }
}
```

```
import java.util.Scanner;
import java.io.File;
public class ReadWithScanner {
  public static void main(String []args) {
    String firstname, lastname, number;  int i=0;  TelephoneNumber numbers[] = new
TelephoneNumber[5];
    try {
      Scanner sc = new Scanner(new File("telephone-list.txt"));
      while( sc.hasNext() ) {
        firstname = sc.next(); lastname = sc.next(); number = sc.next();
        if( lastname.charAt( lastname.length() -1 ) == ',' )
          lastname = lastname.substring(0, lastname.length() -1);
        numbers[i++] = new TelephoneNumber(firstname+" "+lastname, number);
      }
    } catch(IOException ex) {
      ex.printStackTrace();
    }
  }
}
```

Διάβασμα από δυαδικά αρχεία

Διαβάζοντας με χρήση της κλάσης java.io.FileInputStream

Κατασκευαστές

`FileInputStream(String filename)`: Δημιουργεί ένα αντικείμενο της κλάσης FileInputStream από το αρχείο με όνομα filename. Στη θέση του String filename θα πρέπει να βάλετε το path προς το αρχείο και όχι μόνο το όνομα του.

`FileInputStream(File file)`: Δημιουργεί ένα αντικείμενο της κλάσης FileInputStream από ένα αντικείμενο της κλάσης File.

Διαβάζοντας με χρήση της κλάσης java.io.FileInputStream

Μέθοδοι

`int read(byte[] b)`: Διαβάζετε ένα αρχείο μέσω της μεθόδου read. Σε κάθε διάβασμα αποθηκεύονται τα δεδομενα σας στον πίνακα b. Πριν την κλήση της read, θα πρέπει να έχετε αρχικοποιήσει τον πίνακα b σε ένα μέγεθος της επιλογής σας, προκειμένου να μπορούν να αποθηκευτούν τα δεδομένα σας σε αυτόν.

Η μέθοδος διαβάζει το πολύ **b.length** bytes.

Επιστρέφει τον αριθμό των bytes που διάβασε.

Διαβάζοντας με χρήση της κλάσης java.io.FileInputStream

```
public class ReadBinFile {
  public static void main(String [] args) {
    Scanner sc = new Scanner(System.in);
    System.out.print("Enter filename: ");
    String readFilename = sc.next();
    File readFile = new File(readFilename);
    try {
      byte []buffer = new byte[2048];
      FileInputStream in = new
              FileInputStream(readFile);
      int read_len;
      while( (read_len = in.read(buffer)) != -1 ) {
      }
      in.close();
    } catch( IOException ex ) {
      ex.printStackTrace();
    }
  }
}
```

Γράφοντας σε αρχεία κειμένου

Χρήση της κλάσης java.io.PrintWriter

Κατασκευαστές

`PrintWriter(String fileName)`: Κατασκευάζει ένα αντικείμενο της κλάσης PrintWriter από ένα αρχείο filename.

`PrintWriter(File file)`: Κατασκευάζει ένα αντικείμενο της κλάσης PrintWriter από ένα αντικείμενο της κλάσης File.

Μέθοδοι

`void print(String s)`: Εκτυπώνει το String s.

`void println(String s)`: Εκτυπώνει το String s ακολουθούμενο από χαρακτήρα αλλαγής γραμμής.

`PrintWriter printf(String format, Object… args)`: Η συγκεκριμένη μέθοδος είναι ανάλογη με την printf της γλώσσας C. Μπορείτε να διαμορφώσετε κατάλληλα μορφοποιημένη έξοδο.

Παράδειγμα χρήσης της μεθόδου PrintWriter printf(String format, Object… args)

```
import java.io.*;

public class TestPrintf {
  public static void main(String args[]) {
    double f = 754.541012;
    System.out.printf("The value of f is: %08.2f %n", f);
  }
}
```

Παράδειγμα εγγραφής σε αρχείο κειμένου

```
public class CopyTextFile {
  public static void main(String [] args) {
    Scanner sc = new Scanner(System.in);
    System.out.print("Enter src filename: ");
    String readFilename = sc.next();
    File readFile = new File(readFilename);

    System.out.print("Enter dst filename: ");
    String writeFilename = sc.next();
    File writeFile = new File(writeFilename);
```

```
    try {
      String input;
      BufferedReader in = new BufferedReader(new
FileReader(readFile));
      PrintWriter out = new
PrintWriter(writeFile);
      while( (input = in.readLine()) != null ) {
        out.println(input);
      }
      out.close();
      in.close();
    } catch( IOException ex ) {
      ex.printStackTrace();
    }
  }
}
```

Γράφοντας σε δυαδικά αρχεία

Γράφοντας με χρήση της κλάσης java.io.FileOutputStream

Κατασκευαστές

`FileOutputStream(String filename)`: Δημιουργεί ένα αντικείμενο της κλάσης FileOutputStream από το αρχείο με όνομα filename. Στη θέση του String filename θα πρέπει να βάλετε το path προς το αρχείο και όχι μόνο το όνομα του.

`FileOutputStream(File file)`: Δημιουργεί ένα αντικείμενο της κλάσης FileOutputStream από ένα αντικείμενο της κλάσης File.

Μέθοδοι

`public void write(byte[] b)` : Γράφει b.length bytes στο συγκεκριμένο outputStream.

`void write(byte[] b, int off, int len)` : Αντιγράφει στο outputStream len bytes από τον πίνακα b ξεκινώντας από την θέση off του πίνακα. Εάν θέσετε off=0 αντιγράφει από την αρχή του πίνακα.

Παράδειγμα χρήσης java.io.FileOutputStream

```
public class CopyBinFile {
  public static void main(String [] args) {
    Scanner sc = new Scanner(System.in);
    System.out.print("Enter src filename: ");
    String readFilename = sc.next();
    File readFile = new File(readFilename);

    System.out.print("Enter dst filename: ");
    String writeFilename = sc.next();
    File writeFile = new File(writeFilename);
    try {
      byte []buffer = new byte[2048];
      FileInputStream in = new FileInputStream(readFile);
      FileOutputStream out = new
FileOutputStream(writeFile);
      int read_len;
      while( (read_len = in.read(buffer)) != -1 ) {
        out.write(buffer, 0, read_len);
      }
      out.close();
      in.close();
    } catch( IOException ex ) {
      ex.printStackTrace();
    }
  }
}
```

Object Serialization & deserialization

Object Serialization & deserialization

Η Java παρέχει την δυνατότητα μετατροπής ενός αντικειμένου σε μία σειρά από bytes, προκειμένου αυτό στη συνέχεια να αποθηκευτεί σε ένα μέσω μόνιμης αποθήκευσης (π.χ. αρχείο στο filesystem) ή να μεταδοθεί μέσω δικτύου προκειμένου να δημιουργηθεί ένα αντίγραφο του σε απομακρυσμένο σημείο. Η διαδικασία μετατροπής των αντικειμένων σε bytes ονομάζεται object serialization.

Αφού ένα αντικείμενο μετατραπεί σε σειρά από bytes μπορούμε να ακολουθήσουμε την αντίστροφη διαδικασία προκειμένου να δημιουργήσουμε ένα αντίγραφο του αντικειμένου στο ίδιο ή σε άλλο JVM instance.

Η διαδικασία “σειριοποίησης” ενός αντικειμένου είναι ανεξάρτητη του JVM που χρησιμοποιούμε πράγμα που σημαίνει ότι ένα αντικείμενο που μεταδίδεται μέσω δικτύου μπορεί να αναπαραχθεί στη μνήμη σε μορφή αντικειμένου σε απομακρυσμένο JVM.

Object Serialization & deserialization

Οι κλάσεις **`ObjectInputStream`** και **`ObjectOutputStream`** αποτελούν κλάσεις τύπου **`InputStream`** και **`OutputStream`** υψηλού επιπέδου οι οποίες επιτελούν την διαδικασία της "σειριοποίησης" και "αποσειριοποίησης" των αντικειμένων.

Από τις μεθόδους της κλάσης **`ObjectOutputStream`** μας ενδιαφέρει η μέθοδος **`writeObject(Object obj)`** η οποία μετατρέπει σε σειρά από bytes το αντικείμενο **`obj`**.

Αντίστοιχα από τις μεθόδους της κλάσης **`ObjectInputStream`** μας ενδιαφέρει η μέθοδος **`readObject()`** η οποία μετατρέπει σε αντικείμενο τύπου **`Object`** μία σειρά από bytes από το stream.

Προϋποθέσεις εφαρμογής serialization /deserialization

Απαραίτητη προϋπόθεση για το serialization/deserialization των αντικειμένων είναι τα εξής:

- η κλάση του αντικειμένου να υλοποιεί το interface **`java.io.Serializable`**.
- όλα τα πεδία της κλάσης να υλοποιούν το interface **`Serializable`**. Εάν υπάρχουν πεδία που δεν το υλοποιούν τότε αυτά θα πρέπει να δηλωθούν ως *transient*. Η δήλωση *transient* μπροστά από ένα πεδίο υποδεικνύει ότι αυτό δεν θα συμπεριληφθεί στη διαδικασία του serialization/deserialization.

Παράδειγμα - Serialize Employee Class

```
import java.util.*;

public class Employee implements
 java.io.Serializable {

   public String name;

   public Employee next;

   public ArrayList<Employee> list;

   public String toString() {

      String str = "Name: " + name+"\n";

      str+="Address: " + address+"\n";

      if(next != null)

        str+="Next: " + next.name+"\n";
```

```
      if( list != null) {

        Iterator<Employee> it = list.iterator();

        if( it.hasNext() ) {

          str+="List: ";

        }

        while( it.hasNext() ) {

          str+= it.next().name+", ";

        }

        str+="\n";

      }

      return str;

   }

}
```

Object Serialization

```
import java.io.*;
import java.util.*;
public class SerializeDemo {
   public static void main(String [] args) {
      Employee e = new Employee();
      e.name = "Vana Doufexi";
      Employee e1 = new Employee();
      e1.name = "George Thanos";
      e.next = e1; e1.next = e;
      e.list = new ArrayList<>();
      e.list.add(e); e.list.add(e1);
      e1.list = new ArrayList<>();
      e1.list.add(e);  e1.list.add(e1);
```

```
      try (FileOutputStream fileOut = new
FileOutputStream("/tmp/employees.ser");){
         ObjectOutputStream out = new
ObjectOutputStream(fileOut);
         out.writeObject(e);
         out.writeObject(e1);
         out.close();
         fileOut.close();
         System.out.printf("Serialized data is saved
in /tmp/employee.ser");
      } catch(IOException ex) {
         ex.printStackTrace();
      }
   }
}
```

Object Deserialization

```
import java.io.*;
public class DeserializeDemo {
   public static void main(String [] args) {
      Employee e, e1;
      try {
         FileInputStream fileIn = new FileInputStream("employees.ser");
         ObjectInputStream in = new ObjectInputStream(fileIn);
         e = (Employee) in.readObject();
         e1 = (Employee) in.readObject();
         in.close();
         fileIn.close();
      } catch(IOException ex) {
         ex.printStackTrace();
         return;
      }
      catch(ClassNotFoundException ex) {
        System.out.println("Employee class"
                           +" not found");
        ex.printStackTrace();
        return;
      }
      System.out.println(e);
      System.out.println(e1);
   }
}
```