

ΤΕΧΝΙΚΕΣ ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΟΥΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Αρχεία

Ρεύματα

- Τι είναι ένα **ρεύμα** (ροή)? Μια **αφαίρεση** που αναπαριστά μια **ροή δεδομένων**
 - Η ροή αυτή μπορεί να είναι **εισερχόμενη** προς το πρόγραμμα (μια **πηγή** δεδομένων) οπότε έχουμε **ρεύμα εισόδου**.
 - Παράδειγμα: το πληκτρολόγιο, ένα αρχείο που ανοίγουμε για διάβασμα
 - Ή μπορεί να είναι **εξερχόμενη** από το πρόγραμμα (ένας **προορισμός** για τα δεδομένα) οπότε έχουμε ένα **ρεύμα εξόδου**.
 - Παράδειγμα: Η οθόνη, ένα αρχείο που ανοίγουμε για γράψιμο.
- Όταν δημιουργούμε το ρεύμα το **συνδέουμε** με την ανάλογη πηγή, ή προορισμό.

Βασικά ρεύματα εισόδου/εξόδου

- Ένα **ρεύμα** είναι ένα **αντικείμενο**. Τα βασικά ρεύματα εισόδου/εξόδου είναι έτοιμα αντικείμενα τα οποία ορίζονται σαν πεδία (**στατικά**) της κλάσης **System**
- **System.out**: Το **βασικό ρεύμα εξόδου** που αναπαριστά την οθόνη.
 - Έχει στατικές μεθόδους με τις οποίες μπορούμε να τυπώσουμε στην οθόνη.
- **System.in**: Το **βασικό ρεύμα εισόδου** που αναπαριστά το πληκτρολόγιο.
 - Χρησιμοποιούμε την κλάση **Scanner** για να πάρουμε δεδομένα από το ρεύμα.
- Μια εντολή εισόδου/εξόδου έχει αποτέλεσμα το λειτουργικό να πάρει ή να στείλει δεδομένα από/προς την αντίστοιχη πηγή/προορισμό.
- Ένα επιπλέον ρεύμα: **System.err**: Ρεύμα για την εκτύπωση **λαθών** στην οθόνη
 - Μας επιτρέπει την ανακατεύθυνση της εξόδου.

Παράδειγμα

```
class SystemErrTest
{
    public static void main(String args[]) {
        System.err.println("Starting program");
        for (int i = 0; i < 10; i ++){
            System.out.println(i);
        }
        System.err.println("End of program");
    }
}
```

Και τα δύο τυπώνουν στην οθόνη αλλά αν κάνουμε ανακατεύθυνση μόνο το System.out ανακατευθύνεται

Αρχεία

- Ένα ρεύμα εξόδου ή εισόδου μπορεί να **συνδέεται** με ένα **αρχείο** στο οποίο γράφουμε ή από το οποίο διαβάζουμε.
 - Δύο τύποι αρχείων: **Αρχεία κειμένου** (ή αρχεία ASCII) και **δυναμικά (binary) αρχεία**
- Στα αρχεία κειμένου η πληροφορία είναι κωδικοποιημένη σε **χαρακτήρες ASCII**
 - Πλεονέκτημα: μπορεί να διαβαστεί και από ανθρώπους
- Στα binary αρχεία έχουμε διαφορετική **κωδικοποίηση**
 - Πλεονέκτημα: πιο γρήγορη η μεταφορά των δεδομένων.
- Εμείς θα ασχοληθούμε με αρχεία κειμένου

Ρεύμα εξόδου σε αρχεία

- Για να γράψουμε σε ένα αρχείο θα πρέπει καταρχάς να δημιουργήσουμε ένα **ρεύμα εξόδου** που θα **συνδέεται** με το αρχείο.
- Η Java μας παρέχει την κλάση **FileOutputStream** η οποία μας επιτρέπει να δημιουργήσουμε ένα τέτοιο ρεύμα.
- Δημιουργία του ρεύματος:

```
FileOutputStream outputStream =  
    new FileOutputStream(<ονομα αρχείου>);
```

Παράδειγμα

- `FileOutputStream outputStream =
new FileOutputStream("stuff.txt");`
- Δημιουργεί το αντικείμενο `outputStream` το οποίο είναι ένα **ρεύμα εξόδου** προς το αρχείο με το όνομα `stuff.txt`
 - Αν το αρχείο **δεν υπάρχει** τότε **θα δημιουργηθεί** ένα κενό αρχείο στο οποίο μπορούμε να γράψουμε
 - Αν **υπάρχει** ήδη τότε τα περιεχόμενα του θα **σβηστούν** και γράφουμε και πάλι σε ένα κενό αρχείο

FileNotFoundException

- Η δημιουργία του ρεύματος πετάει μια εξαίρεση **FileNotFoundException** την οποία πρέπει να πιάσουμε
 - Η δημιουργία του ρεύματος είναι πάντα μέσα σε ένα **try-catch block**

```
try
{
    FileOutputStream outputStream =
        new FileOutputStream("stuff.txt");
}
catch (FileNotFoundException e)
{
    System.out.println("Error opening the file stuff.txt.");
    System.exit(0);
}
```


FileNotFoundException

- Τι σημαίνει FileNotFoundException όταν δημιουργούμε ένα αρχείο?
 - Μπορεί να έχουμε δώσει λάθος path
 - Μπορεί να μην υπάρχει χώρος στο δίσκο
 - Μπορεί να μην έχουμε write access
 - κλπ

Εγγραφή σε αρχείο

- Με την προηγούμενη εντολή συνδέσαμε ένα **ρεύμα εξόδου** με ένα **αρχείο στο δίσκο**, στο οποίο θα γράψουμε
- Για να γίνει η εγγραφή πρέπει:
 - Να δημιουργήσουμε ένα **αντικείμενο** που μπορεί να **γράφει** στο αρχείο («**Ανοίγουμε το αρχείο**»)
 - Να καλέσουμε **μεθόδους** που γράφουν στο αρχείο («**Εγγραφή**»)
 - Όταν τελειώσουμε να **αποδεσμεύσουμε** το αντικείμενο από το ρεύμα («**Κλείνουμε το αρχείο**»)
- Μπορούμε να τα κάνουμε αυτά με την κλάση **PrintWriter**

PrintWriter

- **Constructor:**

- `PrintWriter (FileOutputStream o)`: Παίρνει σαν όρισμα ένα αντικείμενο τύπου `FileOutputStream`
- Όταν δημιουργούμε ένα αντικείμενο `PrintWriter` ανοίγουμε το αρχείο για γράψιμο.
- Παράδειγμα:
 - `PrintWriter outputWriter = new PrintWriter(outputStream);`

- **Μέθοδοι:**

- `print (String s)`: παρόμοια με την `print` που ξέρουμε αλλά γράφει πλέον στο αρχείο
- `println (String s)`: παρόμοια με την `println` που ξέρουμε αλλά γράφει πλέον στο αρχείο
- `close ()`: ολοκληρώνει την εγγραφή (γράφει ότι υπάρχει στο buffer) και κλείνει το αρχείο
- `flush ()`: γράφει ότι υπάρχει στο buffer

Ένα ολοκληρωμένο παράδειγμα

```
import java.io.PrintWriter;
import java.io.FileOutputStream;
import java.io.FileNotFoundException;

public class TextFileOutputDemol
{
    public static void main(String[] args)
    {
        FileOutputStream outputStream = null;
        try
        {
            outputStream = new FileOutputStream("stuff.txt");
        }
        catch (FileNotFoundException e)
        {
            System.out.println("Error opening the file stuff.txt.");
            System.exit(0);
        }

        PrintWriter outputWriter = new PrintWriter(outputStream);

        System.out.println("Writing to file.");

        outputWriter.println("The quick brown fox");
        outputWriter.println("jumped over the lazy dog.");

        outputWriter.close( );

        System.out.println("End of program.");
    }
}
```

```
import java.io.PrintWriter;  
import java.io.FileOutputStream;  
import java.io.FileNotFoundException;
```

Πιο συνοπτικός κώδικας

```
public class TextFileOutputDemo2  
{  
    public static void main(String[] args)  
    {  
        PrintWriter outputWriter = null;  
        try  
        {  
            outputWriter = new PrintWriter(new FileOutputStream("stuff.txt"));  
        }  
        catch (FileNotFoundException e)  
        {  
            System.out.println("Error opening the file stuff.txt.");  
            System.exit(0);  
        }  
  
        System.out.println("Writing to file.");  
  
        outputWriter.println("The quick brown fox");  
        outputWriter.println("jumped over the lazy dog.");  
  
        outputWriter.close( );  
  
        System.out.println("End of program.");  
    }  
}
```

Το αντικείμενο `FileOutputStream` έτσι κι αλλιώς δεν το χρησιμοποιούμε αλλού. Δημιουργούμε ένα **ανώνυμο αντικείμενο**.

Προσάρτηση σε αρχείο

- Τι γίνεται αν θέλουμε να προσθέσουμε (**append**) επιπλέον δεδομένα σε ένα **υπάρχον αρχείο**
 - Ο constructor της **FileOutputStream** που ξέρουμε θα σβήσει τα περιεχόμενα και θα το ξαναγράψουμε από την αρχή.
- Γι αυτό το σκοπό χρησιμοποιούμε ένα άλλο constructor

```
FileOutputStream outputStream =  
    new FileOutputStream("stuff.txt", true);
```

- Το όρισμα **true** υποδηλώνει ότι θέλουμε να προσθέσουμε (**append**) στο αρχείο

```
import java.io.PrintWriter;  
import java.io.FileOutputStream;  
import java.io.FileNotFoundException;
```

```
public class TextFileOutputDemo3  
{  
    public static void main(String[] args)  
    {  
        PrintWriter outputWriter = null;  
        try  
        {  
            outputWriter = new PrintWriter(new FileOutputStream("stuff.txt"), true);  
        }  
        catch(FileNotFoundException e)  
        {  
            System.out.println("Error opening the file stuff.txt.");  
            System.exit(0);  
        }  
  
        System.out.println("Writing to file.");  
  
        outputWriter.println("The quick brown fox");  
        outputWriter.println("jumped over the lazy dog.");  
  
        outputWriter.close( );  
  
        System.out.println("End of program.");  
    }  
}
```

Ανοίγει το αρχείο για να προσθέσει περιεχόμενο.

Διάβασμα από αρχείο κειμένου

- Η διαδικασία είναι παρόμοια και για διάβασμα
- Πρώτα δημιουργούμε ένα αντικείμενο τύπου **FileInputStream** το οποίο συνδέει ένα ρεύμα εισόδου με το όνομα του αρχείου

```
FileInputStream inputStream =  
    new FileInputStream(<όνομα αρχείου>);
```

- Μετά θα χρησιμοποιήσουμε την γνωστή μας κλάση **Scanner** για να:
 - Να ανοίξουμε το αρχείο
 - `Scanner inputReader = new Scanner(inputStream);`
 - Να διαβάσουμε από το αρχείο
 - `inputReader.nextLine();`
 - Να κλείσουμε το αρχείο
 - `inputReader.close();`

Το `System.in` που χρησιμοποιούσαμε μέχρι τώρα είναι ένα ρεύμα εισόδου

Ένα παράδειγμα

```
import java.util.Scanner;  
import java.io.FileInputStream;  
import java.io.FileNotFoundException;
```

```
public class TextFileScannerDemo  
{  
    public static void main(String[] args)  
    {  
        Scanner inputReader = null;  
  
        try  
        {  
            inputReader =  
                new Scanner(new FileInputStream("morestuff.txt"));  
        }  
        catch(FileNotFoundException e)  
        {  
            System.out.println("File morestuff.txt was not found");  
            System.out.println("or could not be opened.");  
            System.exit(0);  
        }  
  
        String line = inputReader.nextLine( );  
  
        System.out.println("The line read from the file is:");  
        System.out.println(line);  
  
        inputStream.close( );  
    }  
}
```

Η συνοπτική έκδοση του κώδικα

Scanner

- Η Scanner έχει διάφορες μεθόδους για να διαβάσουμε:
 - `nextLine()`: διαβάζει μέχρι το τέλος της γραμμής
 - `nextInt()`: διαβάζει ένα ακέραιο
 - `nextDouble()`: διαβάζει ένα πραγματικό
 - `next()`: διαβάζει το επόμενο λεκτικό στοιχείο (χωρισμένο με κενό)
- Έλεγχοι για τέλος εισόδου
 - `hasNextLine()`: επιστρέφει true αν υπάρχει κι άλλη γραμμή να διαβάσει
 - `hasNext()`: επιστρέφει true αν υπάρχει κι άλλο String να διαβάσει
 - `hasNextInt()`: επιστρέφει true αν υπάρχει κι άλλος ακέραιος

```
import java.util.Scanner;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.io.FileOutputStream;
```

```
public class ReadWriteDemo
{
    public static void main(String[] args){
        Scanner inputStream = null;
        PrintWriter outputStream = null;

        try
        {
            inputStream = new Scanner(new FileInputStream("original.txt"));
            outputStream = new PrintWriter(new FileOutputStream("numbered.txt"));
        }
        catch(FileNotFoundException e){
            System.out.println("Problem opening files."); System.exit(0);
        }

        int count = 0;
        while (inputStream.hasNextLine()){
            String line = inputStream.nextLine();
            count++;
            outputStream.println(count + " " + line);
        }
        inputStream.close();
        outputStream.close();
    }
}
```

Ένα παράδειγμα με διάβασμα και γράψιμο

Διαβάζουμε από ένα αρχείο και γράφουμε τις γραμμές του αριθμημένες σε ένα νέο αρχείο.

Η `hasNextLine` θα επιστρέψει `false` όταν φτάσουμε στο τέλος του αρχείου

Χρήση των εξαιρέσεων για έλεγχο

```
import java.util.Scanner;  
import java.io.FileInputStream;  
import java.io.FileNotFoundException;  
import java.io.PrintWriter;  
import java.io.FileOutputStream;
```

```
public class ReadWriteDemo{  
    public static void main(String[] args){  
        Scanner keyboard = new Scanner(System.in);  
        String inputFilename = keyboard.next();  
        String outputFilename = keyboard.next();  
  
        Scanner inputStream = null;  
        PrintWriter outputStream = null;  
  
        while (true){  
            try  
            {  
                inputStream = new Scanner(new FileInputStream(inputFilename));  
                outputStream = new PrintWriter(new FileOutputStream(outputFilename));  
                break;  
            }  
            catch(FileNotFoundException e){  
                System.out.println("Problem opening files. Enter names again:");  
                inputFilename = keyboard.next();  
                outputFilename = keyboard.next();  
            }  
        }  
        <υπόλοιπος κώδικας...>  
    }  
}
```

Η κλάση File

- Η κλάση File μας δίνει πληροφορίες για ένα αρχείο που θα μπορούσαμε να πάρουμε από το λειτουργικό σύστημα
- Constructor:
 - `File fileObject = new File(<όνομα>);`
 - Το όνομα συνήθως θα είναι ένα όνομα **αρχείου**, αλλά μπορεί να είναι και **directory**.
- Μέθοδοι:
 - `exists()`: επιστρέφει boolean αν υπάρχει ή όχι το αρχείο/path
 - `getName()`: επιστρέφει το όνομα του αρχείου από το full path name
 - `getPath()`: επιστρέφει το path μέχρι το αρχείο από το full path name
 - `isFile()`: boolean που μας λέει αν το όνομα είναι αρχείο η όχι
 - `isDirectory()`: boolean που μας λέει αν το όνομα είναι directory η όχι
 - `mkdir()`: δημιουργεί το directory στο path που δώσαμε ως όρισμα.