Είσοδος – Έξοδος (Input – Output) – Εξαιρέσεις (Exceptions)

Οι εφαρμογές της Java είναι ως επί το πλείστον γραφικές παραθυρικές εφαρμογές και βασίζονται επάνω στο εργαλείο **Java's Abstract Window Toolkit (AWT)**. Η είσοδος των δεδομένων επομένως στα προγράμματα γίνεται μέσα από όμορφο και φιλικό προς τον χρήστη γραφικό περιβάλλον και όχι μέσα από το τερματικό του υπολογιστή. Παρόλα αυτά θα μάθουμε να χρησιμοποιούμε τις μεθόδους εισόδου και εξόδου από το τερματικό.

Η είσοδος και έξοδος δεδομένων στη Java

Οι εργασίες εισόδου - εξόδου γίνονται μέσα από ρεύματα (ή δέσμες) συνεχούς ροής τα **streams** που συνδέονται με τις φυσικές μονάδες του υπολογιστή. Ένα ρεύμα (ή δέσμη συνεχούς ροής - stream) είναι ένα αντικείμενο το οποίο επιτρέπει τη ροή δεδομένων μεταξύ ενός προγράμματος και κάποιας συσκευής εισόδου/εξόδου ή αρχείου. Όταν τα δεδομένα εισάγονται στο πρόγραμμα, τότε το ρεύμα λέγεται ρεύμα εισόδου (input stream), ενώ όταν τα δεδομένα εξάγονται από το πρόγραμμα, τότε το ρεύμα λέγεται ρεύμα εξόδου (output stream). Με τον τρόπο αυτό οι ίδιες κλάσεις και μέθοδοι εφαρμόζονται για οποιαδήποτε συσκευή. Έτσι για παράδειγμα μια δέσμη εισόδου μπορεί να χειριστεί οποιαδήποτε είσοδο, είτε είσοδο από το τερματικό είτε είσοδο από αρχείο. Η java χειρίζεται τα streams μέσα από κλάσεις και μεθόδους του πακέτου **java.io**.

Δύο τύποι δέσμης συνεχούς ροής: byte streams και streams χαρακτήρων

Η Java ορίζει δύο τύπους δέσμης συνεχούς ροής: τον **byte** και **character stream**. Οι δέσμες byte streams χειρίζονται εισόδους και εξόδους των bytes. Τέτοιες εισόδους και εξόδους έχουμε για παράδειγμα όταν διαβάζουμε ή γράφουμε δυαδικά δεδομένα από και προς τα αρχεία. Οι δέσμες character streams χειρίζονται εισόδους και εξόδους χαρακτήρων. Χρησιμοποιούν την δέσμη χαρακτήρων Unicode και για τον λόγο αυτό εφαρμόζονται σε διεθνείς εφαρμογές. Οι δέσμες αυτές είναι και πιο αποτελεσματικές από ότι οι δέσμες byte. Οι δέσμες Character προστέθηκαν στη Java με την έκδοση 1.1. Πάντως οι χαμηλού επιπέδου επεξεργασίες I/O εξακολουθούν να γίνονται με δέσμες byte.

Οι κλάσεις - byte stream

Η Java ορίζει για την byte stream δύο ιεραρχικές κλάσεις τις: **InputStream** και **OutputStream**. Κάτω από αυτές τις κλάσεις υπάρχουν πολλές υποκλάσεις που χειρίζονται τις διαφορετικές συσκευές εισόδου και εξόδου. Μέσα στις δύο αυτές κύριες κλάσεις μπορούμε να ορίσουμε πολλές μεθόδους για την επεξεργασία των bytes αλλά οι δύο κυριότερες από αυτές είναι οι **read()** και **write()** για το διάβασμα και γράψιμο των bytes.

Οι κλάσεις - character stream

Η Java ορίζει για την character stream δύο ιεραρχικές κλάσεις τις: **Reader** και **Writer**. Οι δύο αυτές κλάσεις χειρίζονται την δέσμη Unicode που επιτρέπει την διεθνοποίηση των μηνυμάτων των εφαρμογών μας. Μέσα στις δύο αυτές κύριες κλάσεις μπορούμε να ορίσουμε πολλές μεθόδους για την επεξεργασία των χαρακτήρων. Οι δύο πιο κύριες από τις μεθόδους αυτές είναι η **read()** για το διάβασμα των χαρακτήρων και η **write()** για το γράψιμο των χαρακτήρων.

Προκαθορισμένα streams – Είσοδος χαρακτήρων από το τερματικό

Όλα τα προγράμματα της Java κάνουν χρήση του πακέτου java.lang. Στο πακέτο αυτό ορίζεται μία σημαντική κλάση η **System** που πέρα από τις άλλες σπουδαίες εργασίες που εκτελεί ορίζει και τις προκαθορισμένες μεταβλητές δέσμης: **in, out** και **err**.

- Ήδη έχουμε χρησιμοποιήσει την **System.out** (ρεύμα εξόδου) για την εμφάνιση μηνυμάτων και αποτελεσμάτων στην κονσόλα, που εξ ορισμού είναι η οθόνη.
- Η **System.in** (ρεύμα εισόδου), χρησιμοποιείται για την αρχική είσοδο που είναι το πληκτρολόγιο.
- Η **System.err** χρησιμοποιείται για την αρχική δέσμη λαθών που είναι επίσης εξ ορισμού η οθόνη.

Για να εισάγουμε χαρακτήρες από το τερματικό (πληκτρολόγιο) ενεργοποιούμε την **System.in**, συνδέουμε το πληκτρολόγιο με ένα buffer – χαρακτήρων τον **BufferedReader**. Ο ορισμός του buffer αυτού γίνεται με την εντολή:

```
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
```

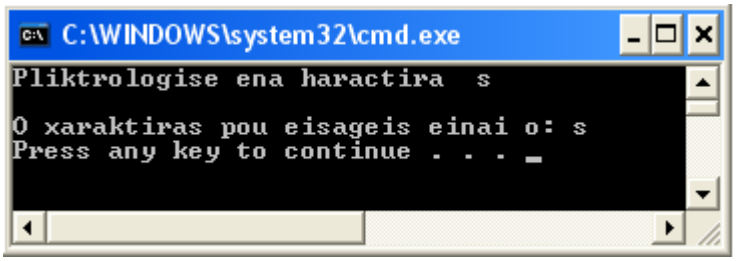
Σε αυτόν τον ορισμό δημιουργείται ένα αντικείμενο (το br – τυχαίο όνομα) του τύπου BufferedReader που θα χειριστεί τη δέσμη εισόδου. Πρώτα γίνεται κλήση της υποκλάσης InputStreamReader που μετατρέπει τα bytes σε χαρακτήρες. Για να επιτευχθεί αυτή η μετατροπή πρέπει να διασυνδεθεί η δέσμη με την System.in.

Είσοδος χαρακτήρα(ων) από το τερματικό (input)

Για να διαβάσουμε ένα χαρακτήρα από τον BufferedReader καλούμε την read(). Η έκδοση της read() είναι του τύπου: **int read() throws IOException.** Για τις εξαιρέσεις (IOException) θα αναφερθούμε παρακάτω.

```
import java.io.*;

class ReadAChar {
  public static void main(String args[]) throws IOException   {
    char c;

    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

    System.out.print("Pliktrologise ena haractira  ");

     c = (char)br.read();
     System.out.println();
     System.out.println("O xaraktiras pou eisageis einai o: " + c);
  }
}
```

Το αποτέλεσμα:



Προσέξτε:

1) την εντολή **import java.io.*;** που εισάγει τις κλάσεις και μεθόδους του πακέτου **io**.
2) την σύνταξη **throws IOException** μετά την main() που θα συντάσσουμε πάντα όταν χειριζόμαστε είσοδο – έξοδο από το τερματικό. Όλες οι εκδοχές της μεθόδου read() είναι του τύπου throws IOException. Δηλαδή:
   α) **int read() throws IOException**

β) **int read(char data[]) throws IOException,** και

γ) **int read(char data[], int start, int max) throws IOException**

Θα χειριστούμε προς το παρόν την πρώτη (α) παραλλαγή – όπως στο ανωτέρω παράδειγμα. Όλες αυτές οι εντολές read() προκαλούν μία εξαίρεση (IOException) σε περίπτωση λάθους. Θα δούμε παρακάτω πως θα χειριστούμε αυτές τις εξαιρέσεις.

3) Για την ανάγνωση πολλών χαρακτήρων θα χρησιμοποιήσουμε εντολές επανάληψης (loops) που θα δούμε σε επόμενο κεφάλαιο.

Είσοδος συμβολοσειρών (Strings) από το τερματικό

Για την είσοδο ενός String από το πληκτρολόγιο χρησιμοποιούμε την **readLine()** που είναι μέλος της κλάσης BufferedReader. Η γενική της μορφή είναι :

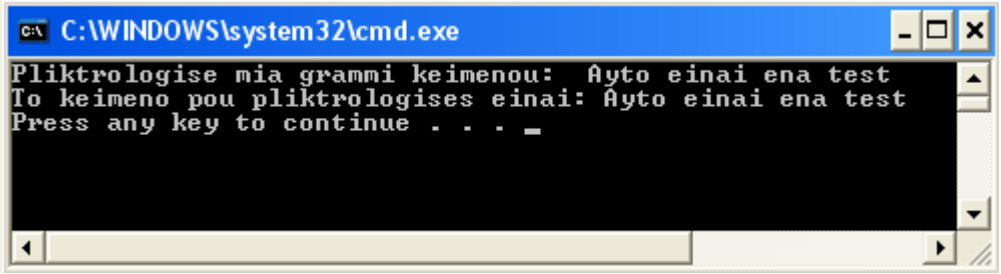
String readLine() throws IOException

Η είσοδος αυτού του τύπου τερματίζεται με το πάτημα του πλήκτρου Enter. Με αυτή την είσοδο θα επιστραφεί ένα αντικείμενο τύπου String που θα περιέχει τους χαρακτήρες που πληκτρολογήσαμε.

Παράδειγμα:

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

   // create a BufferedReader using System.in
   BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
   String str;
   System.out.println("Pliktrologise mia grammi keimenou:  ");
   str = br.readLine();
   System.out.println("H grammi keimenou einai: "+str);
   }
}
```

Το αποτέλεσμα:



Για την ανάγνωση πολλών συμβολοσειρών (γραμμών Strings) θα χρησιμοποιήσουμε εντολές επανάληψης (loops) που θα δούμε επίσης σε επόμενο κεφάλαιο.

Έξοδος χαρακτήρα(ων) στο τερματικό (output)

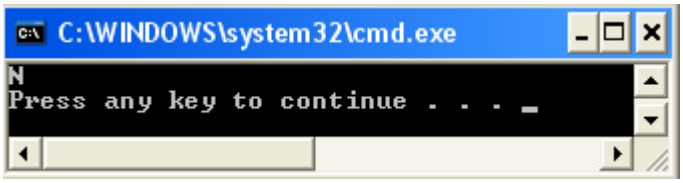
Οι μέθοδοι **print()** και **println()** που έχουμε ήδη χρησιμοποιήσει, ορίζονται από την κλάση **PrintStream** της **System.out** που κάνει έξοδο της μορφής byte – ροής. Επειδή η PrintStream είναι δέσμη εξόδου που προέρχεται από την **OutputStream** περιέχει και την μέθοδο **write()** που χρησιμοποιείται για την έξοδο στο τερματικό (οθόνη). Η απλή της μορφή είναι:

void write(int byteval) throws IOException

Στο παρακάτω παράδειγμα εμφανίζεται ένας χαρακτήρας που εισάγεται μέσα στο πρόγραμμα (ο χαρακτήρας N) και αφού αλλάξει γραμμή τελειώνει:

```
class WriteDemo {

  public static void main(String args[]) {
   char b;
   b = 'N';
   System.out.write(b);
   System.out.write('\n');
  }
}
```

Το αποτέλεσμα:



Έξοδος συμβολοσειρών (Strings) στην κονσόλα

Εκτός από την εμφάνιση αποτελεσμάτων στην κονσόλα με την System.out μπορούμε να εμφανίσουμε κείμενα και αποτελέσματα χρησιμοποιώντας την κλάση **PrintWriter**. Η PrintWriter χρησιμοποιείται περισσότερο στις πραγματικές εφαρμογές από ότι η System.out.

Υποστηρίζει τις μεθόδους **print()** και **println()** για έξοδο όλων των τύπων ακόμη και αντικειμένων (με την κλήση της μεθόδου **toString()**). Για να γράψουμε στην οθόνη χρησιμοποιώντας την PrintWriter ορίζουμε πρώτα την System.out για τη δέσμη εξόδου. Η PrintWriter θα οδηγήσει τη δέσμη στην οθόνη μετά από κάθε νέα γραμμή. Για να γράψουμε στην οθόνη με την PrintWriter πρέπει πρώτα να ορίσουμε ένα αντικείμενο της, χρησιμοποιώντας έναν από

τους δομητές που μας προσφέρει. Εμείς θα χρησιμοποιήσουμε την πιο απλή μορφή δομητή που είναι: **PrintWriter(OutputStream outputstream, Boolean emptyTheOutputStream)**
όπου ορίζουμε το stream εξόδου και ορίζουμε την τιμή true (ή false) αν θέλουμε να αδειάζει το stream όταν τελειώνει η εμφάνιση στην κονσόλα. Για παράδειγμα:

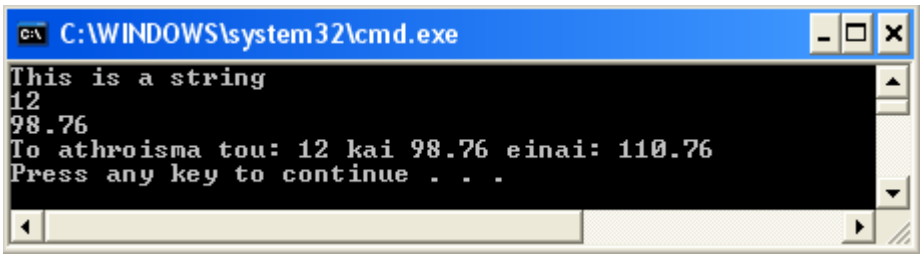
PrintWriter pw = new PrintWriter(System.out, true);

Δηλαδή ορίζουμε ένα αντικείμενο της PrintWriter που προσδιορίζει την System.out σαν stream εξόδου και την τιμή true για το άδειασμα του stream μετά την εμφάνιση των αποτελεσμάτων.

Το παρακάτω παράδειγμα δείχνει την χρήση της PrintWriter και της μεθόδου println() για έξοδο στην οθόνη.

```
import java.io.*;

public class PrintWriterDemo {
  public static void main(String args[]) {
    PrintWriter pw = new PrintWriter(System.out, true);

    int a=12;
    double b=98.76;

    pw.println("This is a string");
    pw.println(a);
    pw.println(b);
    pw.println("To athroisma tou: "+a+" kai "+b+" einai: " +(a+b));
  }
}
```

Το αποτέλεσμα:



Είσοδος αριθμών – Κλάσεις και μέθοδοι τροποποίησης αριθμητικών String

Ακέραιοι ή δεκαδικής υποδιαστολής αριθμοί εισάγονται υπό μορφή String χωρίς αυτόματη μετατροπή στην αντίστοιχη αριθμητική αξία. Η μετατροπή τους από αριθμητικά Strings σε αριθμούς γίνεται με την χρήση των κλάσεων **Integer, Double, Float, Long, Short, Byte.** Για τους

άλλους βασικούς τύπους χρησιμοποιούνται η **Character** και η **Boolean**. Οι αριθμητικές κλάσεις περιέχουν ένα πλήθος από μεθόδους μετατροπής που είναι οι παρακάτω:

Κλάση	Μέθοδοι Μετατροπής
Integer	static int **parseInt**(String str) throws NumberFormatException
Double	static double **parseDouble**(String str) throws NumberFormatException
Float	static float **parseFloat**(String str) throws NumberFormatException
Long	static long **parseLong**(String str) throws NumberFormatException
Short	static short **parseShort**(String str) throws NumberFormatException
Byte	static byte **parseByte**(String str) throws NumberFormatException

Για παράδειγμα αν ένας ακέραιος εισάγεται από το String str, τότε μπορεί να μετατραπεί σε ακέραιο ως εξής:

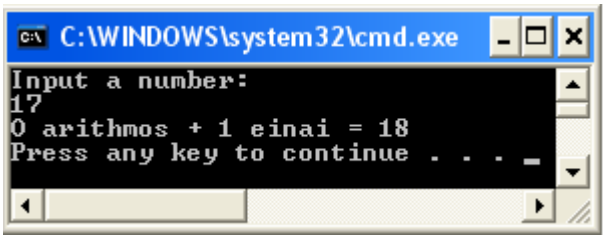
String str=br.readLine();

int x=Integer.parseInt(str);

Παράδειγμα – 1ο:

```
import java.io.*;

public class ANumberInput  {

   public static void main(String args[]) throws IOException {
      int a;
      String str;
      BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

      System.out.println("Input a number: ");
      str=br.readLine();
      a = Integer.parseInt(str);
      System.out.println("O arithmos + 1 einai = " + (a+1));
   }
}
```

Το αποτέλεσμα:

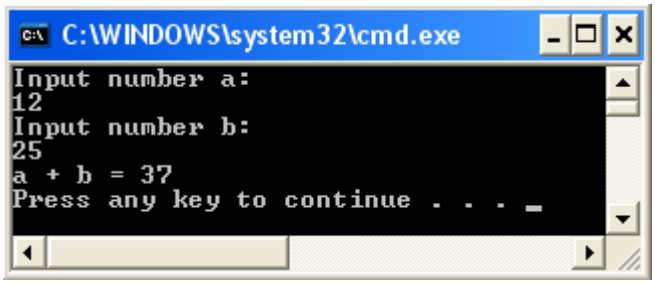


Παράδειγμα – 2ο:

Στο παρακάτω παράδειγμα εισάγουμε δύο ακέραιους, τους προσθέτουμε και εμφανίζουμε το αποτέλεσμα:

```
import java.io.*;

public class NumInput  {

   public static void main(String args[]) throws IOException {
      int a, b, c;
      BufferedReader din = new BufferedReader(new InputStreamReader(System.in));

      System.out.println("Input number a: ");
      a = Integer.parseInt(din.readLine());

      System.out.println("Input number b: ");
      b = Integer.parseInt(din.readLine());

      c = a + b;
      System.out.println("a + b = "+c);
   }
}
```

Το αποτέλεσμα:



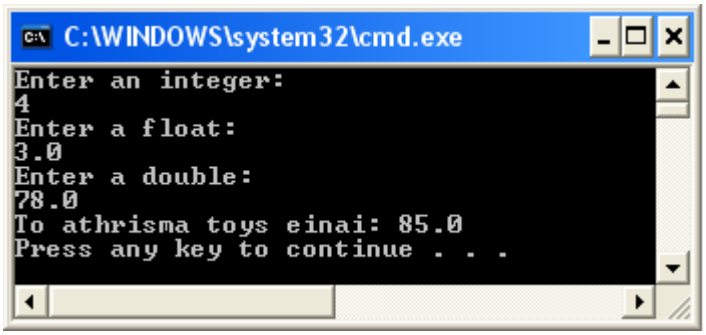
Παράδειγμα – 3ο:

Στο παρακάτω παράδειγμα θα εισάγουμε τρία αριθμητικά Strings των τριών αριθμητικών τύπων (int, float, και double), θα τους μετατρέψουμε σε αντίστοιχους αριθμούς, θα εκτελέσουμε την πρόσθεσή τους και θα εμφανίσουμε το αποτέλεσμα.

```
import java.io.*;

class NumbersInput{
 public static void main(String[] args) throws IOException {

  BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

  int a;
  float f;
  double d;
```

```
  System.out.println("Enter an integer: ");
  a=Integer.parseInt(br.readLine());

  System.out.println("Enter a float: ");
  f=Float.parseFloat(br.readLine());

  System.out.println("Enter a double: ");
  d=Double.parseDouble(br.readLine());

  System.out.println("To athrisma toys einai: " + (a+f+d));
 }
}
```

Το αποτέλεσμα:



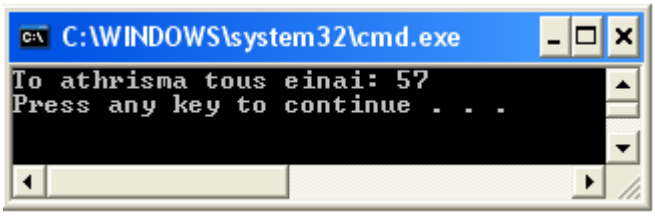
Η μέθοδος valueOf() της κλάσης Integer

Μετατροπή αριθμητικού String σε ακέραιο μπορεί να γίνει και με τη χρήση της μεθόδου **valueOF()** της κλάσης **Integer**.

Παράδειγμα:

```
import java.io.*;

class Number_Input{
 public static void main(String[] args) throws IOException {

  BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

        String numberOne = "12";
        String numberTwo = "45";

        Integer myint1 = Integer.valueOf(numberOne);
        Integer myint2 = Integer.valueOf(numberTwo);

        int athrisma = myint1.intValue() + myint2.intValue();

        System.out.println("To athrisma tous einai: "+athrisma);
}
}
```

Το αποτέλεσμα:



Είσοδος από το τερματικό με την κλάση Scanner

Από την έκδοση 5.0 και μετά η Java περιέλαβε την κλάση **Scanner** που κάνει πιο απλή την είσοδο από το πληκτρολόγιο. Για να χρησιμοποιήσουμε την κλάση και τις μεθόδους της πρώτα εισάγουμε την εντολή: **import java.util.Scanner;** που κάνει διαθέσιμη την κλάση Scanner και τις μεθόδους της από την βιβλιοθήκη κλάσεων (package) της java.util. Μετά δημιουργούμε ένα αντικείμενο της κλάσης Scanner με την εντολή: **Scanner input = new Scanner(System.in);** Το όνομα του αντικειμένου μπορεί να είναι οποιοδήποτε θέλουμε εμείς (εδώ input). Μετά την δημιουργία του αντικειμένου μπορούμε να χρησιμοποιήσουμε τις μεθόδους της Scanner για είσοδο από το πληκτρολόγιο. Τέτοιες μέθοδοι είναι οι **nextInt,** που διαβάζει μια τιμή int από το πληκτρολόγιο και την αναθέτει σε μια μεταβλητή. π.χ. **int i = input.nextInt();**, η **nextDouble**, που διαβάζει μια τιμή double από το πληκτρολόγιο και την αναθέτει σε μια μεταβλητή. Προσοχή στην είσοδο του αριθμού (θέλει κόμμα κατά την είσοδο). Π.χ. **double d = input.nextDouble();**, κ.λ.π.

Πολλαπλές είσοδοι διαχωρίζονται με το κενό διάστημα ή τα tabs και τα line breaks και πρέπει να διαβάζονται με πολλαπλές εισόδους.

Η μέθοδος **next()** χρησιμοποιείται για την είσοδο ενός String. Πολλαπλά Strings μπορούν να εισαχθούν με το κενό διάστημα. Για παράδειγμα σε μια είσοδο του τύπου:

String s1= input.next();

String s2= input.next();

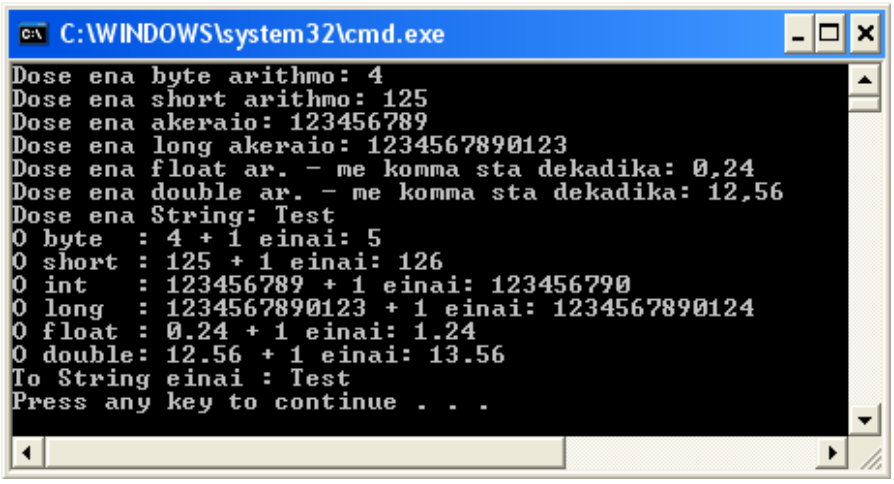
και της γραμμής εισόδου: Nikas Nikos, τότε το s1 θα πάρει την τιμή Nikas, ενώ το s2 την τιμή Nikos.

Η μέθοδος **nextLine()** εισάγει μια ολόκληρη γραμμή κειμένου. Το τέλος της γραμμής σηματοδοτείται με το πάτημα του πλήκτρου Enter (χαρακτήρας διαφυγής '\n').

Παράδειγμα:

```
import java.util.Scanner;

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

    byte b;
    short s;
    int i;
    long l;
    float f;
    double d;
    String str;

    Scanner input = new Scanner(System.in);

    System.out.print("Dose ena byte arithmo: ");
    b = input.nextByte();
    System.out.print("Dose ena short arithmo: ");
    s = input.nextShort();
    System.out.print("Dose ena akeraio: ");
    i = input.nextInt();
    System.out.print("Dose ena long akeraio: ");
    l = input.nextLong();
    System.out.print("Dose ena float ar. - me komma sta dekadika: ");
    f = input.nextFloat();
    System.out.print("Dose ena double ar. - me komma sta dekadika: ");
    d = input.nextDouble();
    System.out.print("Dose ena String: ");
    str = input.next();
    System.out.println("O byte  : " + b + " + 1 einai: " + (b+1));
    System.out.println("O short : " + s + " + 1 einai: " + (s+1));
    System.out.println("O int   : " + i + " + 1 einai: " + (i+1));
    System.out.println("O long  : " + l + " + 1 einai: " + (l+1));
    System.out.println("O float : " + f + " + 1 einai: " + (f+1));
    System.out.println("O double: " + d + " + 1 einai: " + (d+1));
    System.out.println("To String einai : " + str);
  }
}
```

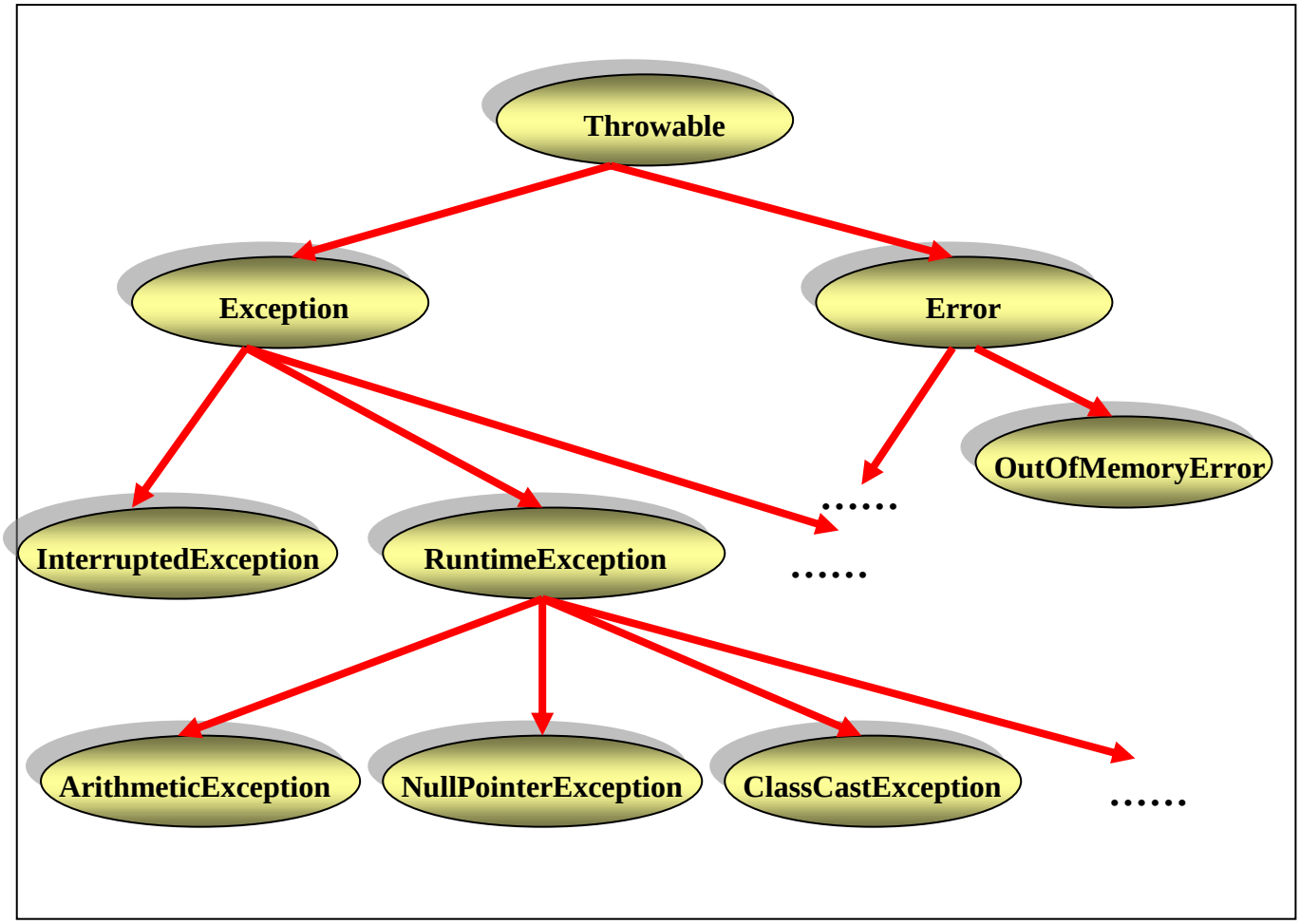
Το αποτέλεσμα:



Η κλάση Scanner χρησιμοποιείται και για είσοδο δεδομένων από αρχεία, όμως ο χειρισμός των αρχείων δεν συμπεριλαμβάνεται στην ύλη του εξαμήνου.

Εξαιρέσεις στη Java

Η java παρέχει ένα μηχανισμό, τις **Εξαιρέσεις** (exceptions), που βοηθά το πρόγραμμα να βρει και να χειριστεί τα λάθη που προκύπτουν κατά τις εργασίες I/O. Εξαίρεση λέγεται ένα συμβάν, συνήθως λάθος, που διακόπτει την ροή εκτέλεσης ενός προγράμματος. Οι εξαιρέσεις είναι αντικείμενα που προέρχονται άμεσα ή έμμεσα από την κλάση **Throwable.** Η κλάση αυτή περιέχει δύο υποκλάσεις τις **Error** και **Exception.** Μία υποκλάση της Exception είναι η **RuntimeException.**



Οι κλάσεις που παράγονται από την Error χρησιμοποιούνται για σημαντικά λάθη του συστήματος, όπως για παράδειγμα το λάθος OutOfMemoryError, ενώ οι κλάσεις που παράγονται από την Exception και την RuntimeException, όπως π.χ. το ArithmeticException, χρησιμοποιούνται για

συνηθισμένα λάθη. Όταν συμβεί ένα λάθος στο πρόγραμμα, π.χ. διαίρεση δια του μηδέν, τότε η Java δημιουργεί αυτόματα ένα αντικείμενο αυτών των κλάσεων και το **"ρίχνει" (throws)** στο πρόγραμμα. Αν το πρόγραμμα δεν το **"πιάσει" (catch),** τότε διακόπτεται η ροή του προγράμματος με την εμφάνιση ενός μηνύματος λάθους. Η κλάση δείχνει τον τύπο του λάθους, ενώ το αντικείμενο περιέχει ένα λεπτομερές μήνυμα για το λάθος, όπως για παράδειγμα το που συνέβη.

Για να δηλώσουμε ένα πιθανόν λάθος και να το διορθώσουμε πρέπει πρώτα να το στείλουμε στον κατάλληλο κώδικα και αυτό γίνεται με την εντολή - **throw**. Η γενική μορφή της εντολής είναι:

throw <exception>; (δες αναλυτικά παρακάτω)

Η throw διακόπτει την φυσική ροή του προγράμματος και προσπαθεί να βρει ένα **χειριστή εξαιρέσεων (exception handler)** για τον τύπο της εξαίρεσης που συνέβη. Ο χειριστής εξαιρέσεων είναι ένα κομμάτι κώδικα που χειρίζεται το ειδικό λάθος που προέκυψε. Ο κώδικας αυτός είτε ανακαλύπτει το λάθος είτε προκαλεί έξοδο από το πρόγραμμα αν αυτό δεν μπορεί να ανακαλυφθεί. Τρεις εντολές χειρίζονται τις εξαιρέσεις οι : **try, catch** και **finally.**

```
try {
   εντολές
   :      :
} catch (όνομα του τύπου της Εξαίρεσης) {
   εντολές
   :      :
} catch (όνομα του τύπου της Εξαίρεσης) {
   εντολές
   :      :
} finally {
   εντολές
   :      :
}
```

Η εντολή – **try** περιέχει ένα κομμάτι κώδικα στο οποίο στέλνεται μία εξαίρεση. Αν συμβεί η εξαίρεση που έχει προβλέψει ο κώδικας της try, τότε μία εντολή – **catch** με τον κατάλληλο κώδικα θα χειριστεί τον τύπο της εξαίρεσης. Εάν δεν συμβεί εξαίρεση στο τμήμα του κώδικα της try, τότε το κομμάτι του κώδικα της catch δεν εκτελείται. Η εντολή – **finally,** που συνοδεύει πάντα την try, περιέχει ένα κομμάτι κώδικα που εκτελείται είτε συμβεί λάθος είτε όχι μέσα στην try.

Παράδειγμα:

Ο παρακάτω κώδικας προκαλεί λάθος (διαίρεση δια του μηδενός) :

```
class Lathos1 {
        public static void main(String args[]) {
                int b = 0;
                int a = 20 / b;
        }
}
```

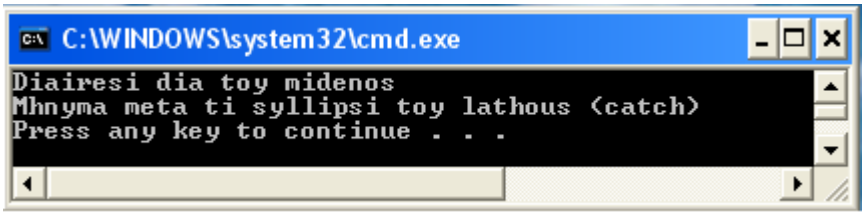
Επειδή συμβαίνει λάθος στην διαίρεση δημιουργείται εξαίρεση και το πρόγραμμα τερματίζει. Επειδή δεν έχουμε προγραμματίσει κώδικα χειρισμού της εξαίρεσης αναλαμβάνει δράση ο χειριστής εξαιρέσεων της java. Αυτός με την σειρά του εμφανίζει το μήνυμα της εξαίρεσης καθώς και ένα αντίγραφο της μνήμης στο σημείο που συνέβη, ενώ παράλληλα τερματίζει και την εκτέλεση του προγράμματος.

```
Exception in thread "main" java.lang.ArithmeticException: /by zero at Lathos1.main(Lathos1.java:4)
```

Για να αποφύγουμε μία τέτοια κατάσταση λάθους θα χρησιμοποιήσουμε μία **try – catch** εντολή. Με τον τρόπο αυτό θα αποφύγουμε το λάθος και τον απρόσμενο τερματισμό του προγράμματος. Στην try γράφουμε τον κώδικα που υποθέτουμε ότι θα προκαλέσει το λάθος και στην catch τον κώδικα που 'πιάσει' το λάθος δηλαδή, τον τύπο της εξαίρεσης που περιμένουμε.

```
class CatchLathos {
  public static void main(String args[]) {
    int a, b;

    try {          // ο κώδικας όπου περιμένουμε το λάθος
      b = 0;
      a = 20 / b;
      System.out.println("Ayto to mhnyma den tha emfanistei");

} catch (ArithmeticException e) {        // 'σύλληψη' του λάθους
      System.out.println("Diairesi dia toy midenos");
  }
    System.out.println("Mhnyma meta ti syllipsi toy lathous (catch)");
 }
}
```

Αν εκτελέσουμε το πρόγραμμα θα πάρουμε το παρακάτω αποτέλεσμα:



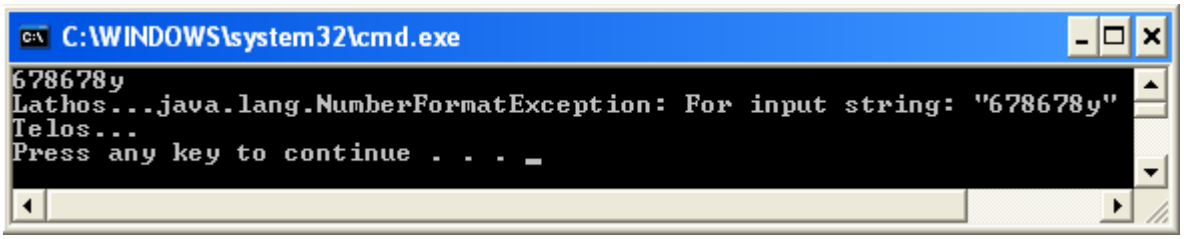
Μπορούμε να εμφανίσουμε τον τύπο της εξαίρεσης τροποποιώντας την εντολή catch:

```
catch (ArithmeticException e) {
  System.out.println("Exception: " + e);
}
```

Ο υψηλότερη στην ιεραρχία των κλάσεων είναι η κλάση της εξαίρεσης που συντάσσουμε στην catch τόσο περισσότερα λάθη πιάνουμε, όπως στο παρακάτω παράδειγμα**:**

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

BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

Try {
    int inputLine = Integer.parseInt(reader.readLine());
    //String inputLine = reader.readLine();
}
catch(Exception exc) {
    System.out.println("Lathos..." + exc);
}
System.out.println("Telos...");
}
}
```

Αν κατά την εκτέλεση του προγράμματος πληκτρολογήσουμε στον εισαγόμενο αριθμό κάποιο χαρακτήρα τότε θα πάρουμε το παρακάτω αποτέλεσμα:



Δηλαδή μπορούμε να πιάσουμε περισσότερα είδη εξαιρέσεων αν δηλώσουμε στην catch κάποια πιο γενική υπερκλάση όπως π.χ. την RuntimeException ή την Exception. Ο τύπος Throwable πιάνει όλες τις εξαιρέσεις. Συνήθως όμως χρησιμοποιούμε πολλές catch εντολές. Επίσης μπορούμε να χρησιμοποιήσουμε εστιασμένες try και catch – εντολές.

"Ρίψη" εξαίρεσης από κληθείσα μέθοδο

Η εντολή try μπορεί να πιάσει εξαιρέσεις οι οποίες εγείρονται μέσα σε μεθόδους που καλούνται στο σώμα της. Π.χ.

```
void myMethod1()
try
{
  myMethod2()
}
```

```
catch (ArithmeticException a)
{
  …..// kapoies entoles
}
}
```

Αν η μέθοδος myMethod2() ρίξει μια ArithmeticException και δεν πιαστεί η εξαίρεση θα προωθηθεί στην myMethod1().

Η Exception και η Throwable εμπεριέχουν τις ακόλουθες μεθόδους:

- String **getMessage()** //εμφανίζει το μήνυμα λάθους
- String **toString()** // εμφανίζει το όνομα της εξαίρεσης
- void **printStackTrace()** // εμφανίζουν το σωρό των
- void **printStackTrace(PrintStream)** // μεθόδων που κλήθηκαν

Throw

'Ρίψη' εξαίρεσης μπορούμε να κάνουμε και εμείς, όπως και η java κατά την εκτέλεση του προγράμματος, χρησιμοποιώντας την εντολή **throw.** Τότε έχουμε ένα νέο 'ριπτόμενο' αντικείμενο εξαίρεσης, π.χ. **throw new RuntimeException**. Η εντολή throw προκαλεί την προσωρινή διακοπή της εκτέλεσης του προγράμματος. Στην συνέχεια ελέγχονται οι **try** και **catch** διαδοχικά μέσα στο πρόγραμμα εάν η εξαίρεση ταιριάζει και έχει προβλεφθεί. Αν η εξαίρεση ταιριάξει με κάποια εντολή, τότε έχουμε διακλάδωση σε αυτήν. Διαφορετικά αν δεν γίνει διακλάδωση ο έλεγχος μεταβιβάζεται στον αρχικό χειριστή ο οποίος προκαλεί την διακοπή της εκτέλεσης του προγράμματος και εμφανίζει ένα αντίγραφο της μνήμης για το λάθος.

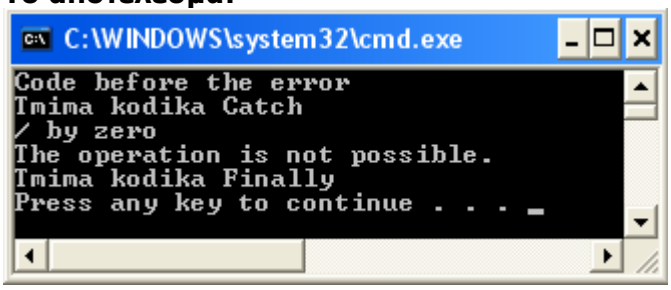
Finally

Κάθε try εντολή συνοδεύεται απαραίτητα από μία catch ή finally εντολή. Ο κώδικας της finally εκτελείται πάντα μετά από κάθε try εντολή. Η μόνη περίπτωση που δεν εκτελείται είναι να έχει προηγηθεί η εντολή **System.exit(0)** που προκαλεί την έξοδο από το πρόγραμμα.

```
public class TestExceptions{
  public static void main(String args[]){
        try {
          System.out.println("Code before the error");
          System.out.println(10/0);
          System.out.println("Code after the error");
         }
        catch(java.lang.ArithmeticException e){
          System.out.println("Tmima kodika Catch");
```

```
            System.out.println(e.getMessage());
            System.out.println("The operation is not possible.");
        }
        finally{
            System.out.println("Tmima kodika Finally");
        }
    }
}
```

Το αποτέλεσμα:



Τις εντολές try – catch – finally θα τις δούμε και σε επόμενα κεφάλαια όπως στον χειρισμό των πινάκων (arrays), κλπ.

Μορφοποίηση εξόδου με την printf

Μετά την έκδοση 5.0, η Java περιλαμβάνει μια μέθοδο την **printf** (παρόμοιας με την print), που μπορεί να αναπαραστήσει την έξοδο με συγκεκριμένο τρόπο (format). Όπως και στην print, η ενφάνιση γίνεται στην τρέχουσα γραμμή. Η System.out.printf μπορεί να περιέχει οποιοδήποτε αριθμό ορισμάτων. Το πρώτο όρισμα είναι πάντοτε ένα String μορφοποίησης (format string) που εμπεριέχει ένα ή περισσότερους προσδιοριστές μορφοποίησης (format specifiers) για τα υπόλοιπα ορίσματα. Τα υπόλοιπα ορίσματα (εκτός από το πρώτο) είναι τιμές που θα εμφανιστούν στην οθόνη.
Δύο σημαντικοί προσδιοριστές μορφοποίησης είναι ο **"% <αριθμός ψηφίων>.2f"** π.χ. **"%6.2f"** και ο **"%.2f"**. Οι δύο αυτοί προσδιοριστές χρησιμοποιούνται για την απεικόνιση των αριθμών με δεκαδικά ψηφία.
Ο προσδιοριστής "%n.2f" σημαίνει:

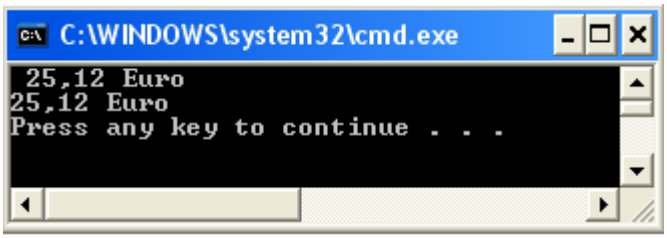
1) έναρξη του προσδιοριστή με το σύμβολο %
2) εμφάνισε το πολύ n χαρακτήρες, δεξιά στοιχισμένους (αν χρειαστεί γέμισε με κενά διαστήματα από αριστερά)
3) εμφάνισε 2 ψηφία μετά την υποδιαστολή (.2) και τερμάτισε την εμφάνιση του αριθμού τύπου float

Ο προσδιοριστής "%.2f" σημαίνει ότι θα εμφανιστούν 2 δεκαδικά ψηφία μετά την υποδιαστολή και θα χρησιμοποιηθεί το μικρότερο δυνατό εύρος ψηφίων.

Στο παρακάτω παράδειγμα γίνεται χρήση αυτών των προσδιοριστών:

```
class NumberFormat {
 public static void main(String[] args){
        double numb = 25.123456789;
        System.out.printf("%6.2f", numb);
        System.out.println(" Euro");
        System.out.printf("%.2f", numb);
        System.out.println(" Euro");
 }
}
```

Αν εκτελέσουμε το πρόγραμμα θα πάρουμε το παρακάτω αποτέλεσμα:



Άλλοι προσδιοριστές είναι ο **"%d"** για εμφάνιση ακεραίων, ο **"%e"** για εμφάνιση επιστημονικής μορφής (E-notation), ο **"%g"** για επιστημονική ή μη μορφή (η java αποφασίζει) και τέλος ο **"%c"** για εμφάνιση χαρακτήρων.

Μορφοποίηση με την NumberFormat και την DecimalFormat

Με την κλάση NumberFormat μπορούμε να τυπώσουμε χρηματικά ποσά στην κατάλληλη μορφή. Πρώτα εισάγουμε την κλάση NumberFormat με την εντολή import:

import java.text.NumberFormat και μετά δημιουργούμε ένα αντικείμενο του τύπου NumberFormat που χρησιμοποιούμε με την μέθοδο **getCurrencyInstance().** Τέλος με τη μέθοδο **format,** που δέχεται ως όρισμα έναν αριθμό τύπου float, εμφανίζουμε τον αριθμό ως String στο τοπικό νόμισμα.

Παράδειγμα:

```
import java.text.NumberFormat;

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

        NumberFormat moneyFormater = NumberFormat.getCurrencyInstance();
        System.out.println(moneyFormater.format(89.9));
        System.out.println(moneyFormater.format(89.99999));
        System.out.println(moneyFormater.format(89.999));
        System.out.println(moneyFormater.format(89));
        System.out.println(NumberFormat.getCurrencyInstance().format(12345.6789));
  }
}
```

Η κλάση DecimalFormat

Η κλάση **DecimalFormat** επιτρέπει την μορφοποίηση των αριθμών με πολλούς διαφορετικούς τρόπους. Πρώτα εισάγεται η κλάση με την εντολή: **import java.text.DecimalFormat;** και μετά δημιουργείται ένα αντικείμενο της κλάσης το οποίο χρησιμοποιείται με την μέθοδο **format** και ένα **προσδιοριστή** για την απεικόνιση των αριθμών υπό μορφή String.

```
import java.text.*;
public class DecimalFormatTest {
  public static void main( String[] args ) {
     double d = 12345.67890;
      DecimalFormat df = new DecimalFormat("#####.00");
      System.out.println(df.format(d));          // emfanizei 12.345,68
  }
}
```

Μερικοί σημαντικοί προσδιοριστές:

Προσδιοριστής	Παράδειγμα	Αποτέλεσμα
0000	12	0012
0000	12,5	0012
0000	1234567	1234567
##	12	12
##	12.3456	12
##	123456	123456
.00	12.3456	12,35
.00	.3456	,35
0.00	.789	0,79
#.000000	12.34	12,340000
,###	12345678.901	12.345.679
#.#;(#.#)	12345678.901	12345678,9
#.#;(#.#)	–12345678.901	(12345678,9)
,###.## \u00A4	12345.6789	12.345,68 i
,#00.00 \u00A4\u00A4	–12345678.9	–12.345.678,90 EUR
,#00.00 \u00A4\u00A4	0.1	00,10 EUR