Προγραμματισμός ΙΙ (Java)

5. Διαχείριση αρχείων

Ρεύματα

- Όταν ένα πρόγραμμα επικοινωνεί με την οθόνη, το πληκτρολόγιο, ένα αρχείο κλπ., χρησιμοποιεί ρεύματα χαρακτήρων (streams) για να στείλει ή να πάρει δεδομένα.
- Τρία προκαθορισμένα ρεύματα στη Java:
  - System.in: Διαβάζει bytes ***εξ ορισμού*** από το πληκτρολόγιο.
  - System.out: Στέλνει bytes ***εξ ορισμού*** στην οθόνη.
  - System.err: Στέλνει μηνύματα λάθους ***εξ ορισμού*** στην οθόνη.
- «συνήθως». Μπορεί όμως να συνδεθεί με άλλη πηγή με τις μεθόδους setIn, setOut, setErr της System
- π.χ. FileOutputStream f = new FileOutputStream("file.txt");
  System.setOut(new PrintStream(f));

Προσωρινή αποθήκευση (buffering)

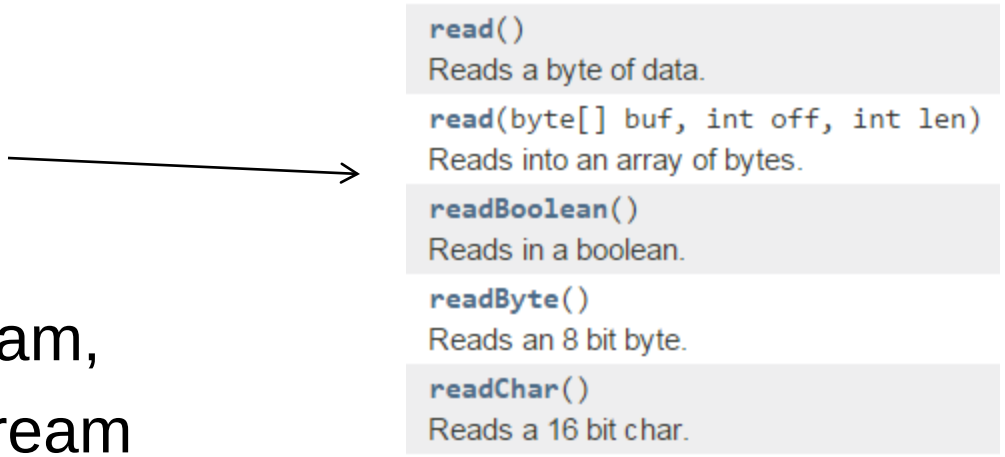
- Κάθε byte που διαβάζουμε από το αρχείο (ή το πληκτρολόγιο) τοποθετείται σε μια προσωρινή μνήμη. Μόλις συγκεντρωθούν «αρκετά» bytes τα χειριζόμαστε ανάλογα (π.χ. τα εμφανίζουμε στην οθόνη, τα μετατρέπουμε στον κατάλληλο τύπο δεδομένων κλπ)
- Με buffer: γραφή και ανάγνωση σε τμήματα (“chunks”)
  - Καθυστερεί ο χειρισμός ορισμένων bytes (καθώς περιμένουν να γεμίσει ο buffer) π.χ. σε ένα 16-byte buffer, μπορεί να περιμένουμε να διαβαστούν όλα τα bytes πριν χρησιμοποιήσουμε τα 4 πρώτα για έναν int ή δεν γράφουμε τον ακέραιο στο αρχείο, μέχρι να γεμίσει ο buffer, οπότε γράφουμε 4 int μαζί
  - Λιγότερος επιπλέον φόρτος πρόσβασης στο δίσκος (I/O overhead)
- Χωρίς buffer: επεξεργαζόμαστε κάθε byte μόλις έρθει στη μνήμη
  - Μικρότερη καθυστέρηση για το χειρισμό του ενός byte
  - Μεγάλος επιπλέον φόρτος γραφής και ανάγνωσης από το δίσκο

Ρεύματα για bytes (Java 1.0)

- Όλα τα ρεύματα που θα δούμε βρίσκονται στο πακέτο io.
- Βρίσκονται κάτω από τις abstract τάξεις InputStream και OutputStream οι οποίες περιέχουν τις βασικές μεθόδους για είσοδο και έξοδο bytes.
  - *int read(): διαβάζει το επόμενο byte*
  - *int read(byte[] b): διαβάζει τα επόμενα b.length bytes στον πίνακα b*
  - *long skip(long n): αγνοεί τα επόμενα n bytes*
  - *int available(): επιστρέφει τον αριθμό των διαθέσιμων bytes*
  - *void mark(*int readlimit*): σημαδεύει τη συγκεκριμένη θέση στο αρχείο*
  - *void reset(): επαναφέρει το δείκτη στην αρχή του αρχείο ή στο τελευταίο σημάδι*
  - *void close(): κλείνει το αρχείο*

Υποκλάσεις των Input/Output Streams

- Προσθέτουν περισσότερες λειτουργίες στα ρεύματα
- π.χ οι FilterInputStream και FilterOutputStream.
  - Κατασκευαστής: *FilterInputStream(InputStream in)*
  - Πεδία: *in* το InputStream με το οποίο συνδέεται
- AudioInputStream, ByteArrayInputStream,
- FileInputStream,
- FilterInputStream,
- ObjectInputStream,
- PipedInputStream,
- SequenceInputStream,
- StringBufferInputStream

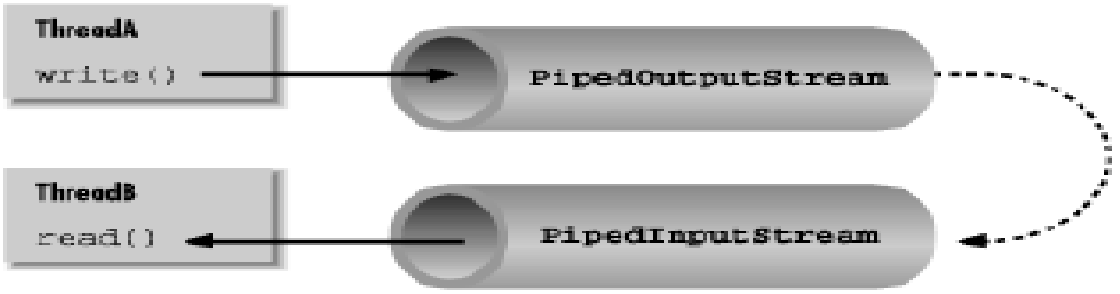


Ρεύματα για bytes (2)

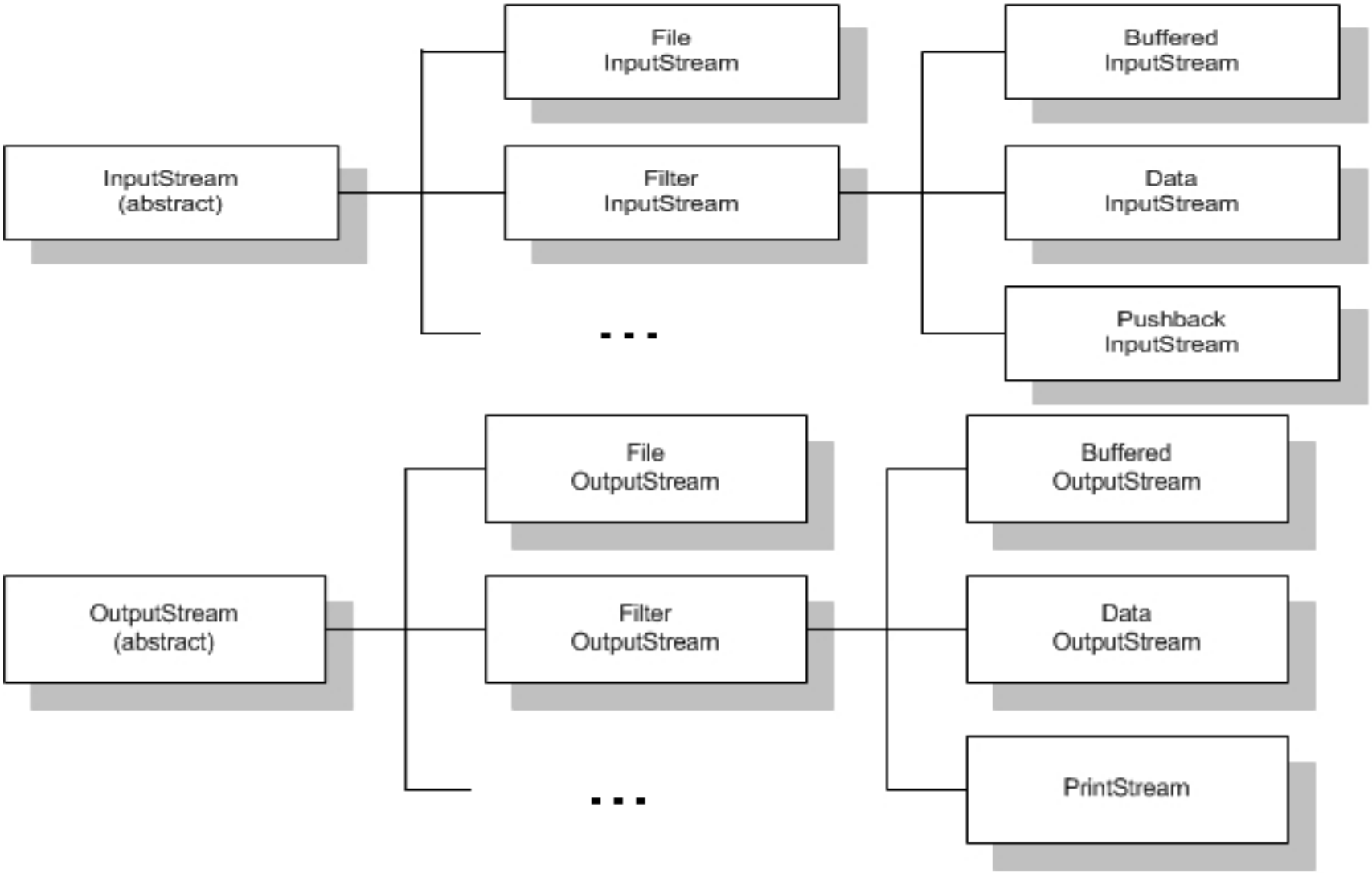
- Τάξεις που τις κληρονομούν είναι οι PrintStream (π.χ. η System.out), οι BufferedInputStream και BufferedOutputStream που έχουν κάποιο buffer,
  - Κατασκευαστής: *BufferedInputStream(InputStream in)*
  - Πεδία: *buf :* ένα byte[] που αποθηκεύει τα δεδομένα
    count: το πλήθος των bytes στο buffer
- οι DataInputStream και DataOutputStream που διαβάζουν βασικούς τύπους υλοποιώντας αντίστοιχα interfaces (DataInput και DataOutput).
  - Κατασκευαστής: *DataInputStream(InputStream in)*
  - Μέθοδοι: *readFloat*(), διαβάζει 4 bytes και επιστρέφει float

Ρεύματα για bytes (3)

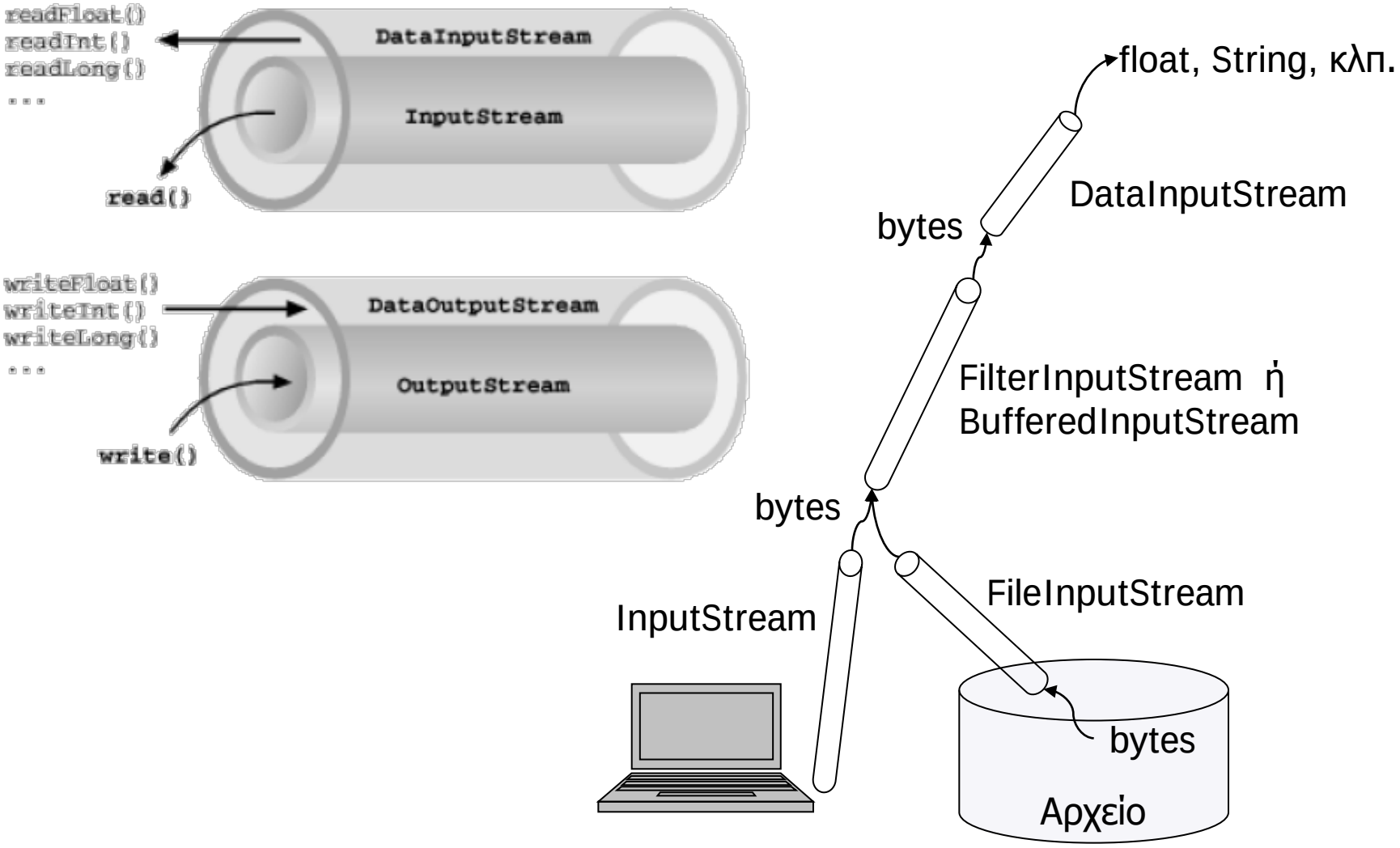
- Για ανταλλαγή δεδομένων όταν έχουμε πολλά threads υπάρχουν οι PipedInputStream / PipedOutputStream
  - Κατασκευαστής: PipedInputStream()
    PipedOutputStream(PipedInputStream src)
  - Μέθοδοι: connect(PipedInputStream snk)
    write(byte[] b, int off, int len)
- Για αρχεία υπάρχουν οι FileInputStream και FileOutputStream.
  - Κατασκευαστής: *FileInputStream(String filename)*
    throws FileNotFoundException



Ιεραρχία

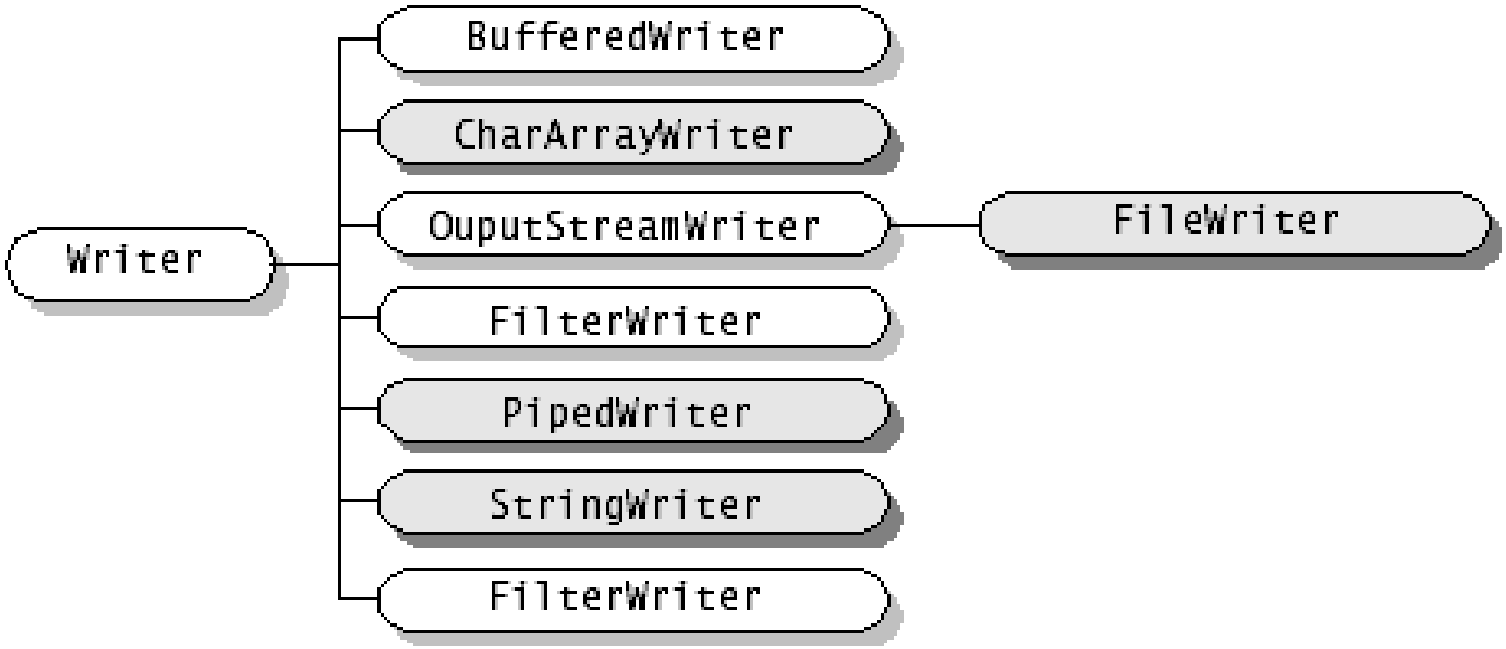

InputStream
(abstract)
File
InputStream
Filter
InputStream
...
Buffered
InputStream
Data
InputStream
Pushback
InputStream
OutputStream
(abstract)
File
OutputStream
Filter
OutputStream
...
Buffered
OutputStream
Data
OutputStream
PrintStream

Συνδυασμός ρευμάτων bytes



Ρεύματα για χαρακτήρες (Java 1.1)

- Για τη διαχείριση ρευμάτων 16-bit (Unicode) χαρακτήρων υπάρχουν οι abstract τάξεις Reader και Writer με αντίστοιχες ιεραρχίες



Βασικές μέθοδοι

- Οι Reader/Writer έχουν μεθόδους για χαρακτήρες

  int read()

  // ο ακέραιος αντιστοιχεί σε κάποιο char με δεδομένο charset

  // -1 όταν το ρεύμα είναι άδειο (π.χ. EOF)

  int read(char cbuf[])

  int read(char cbuf[], int offset, int length)

  int write(int c)

- Οι InputStream/OutputStream αντίστοιχα για bytes

  π.χ. **int write(byte b[], int offset, int length)**

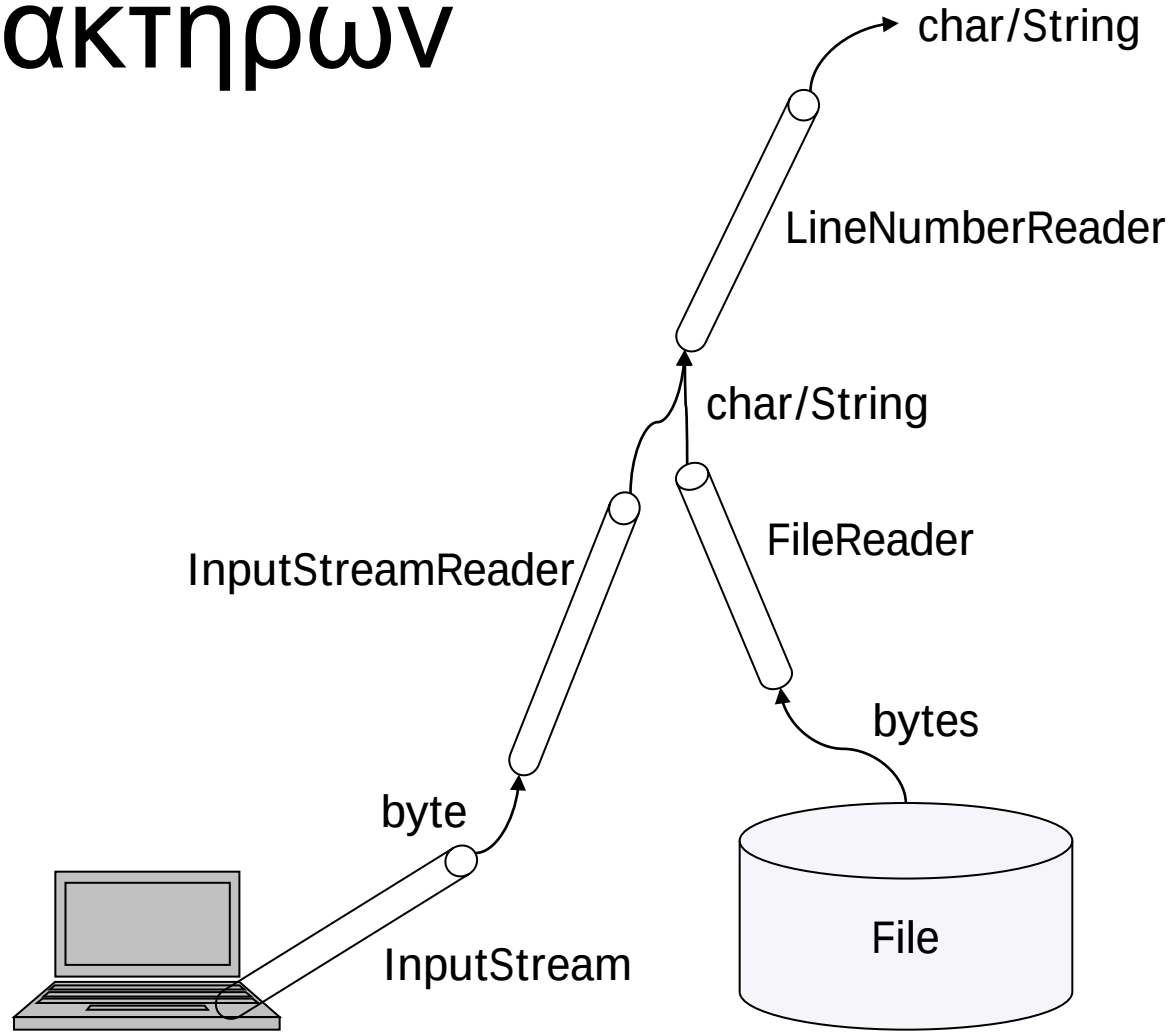
- Οι InputStreamReader και OutpuStreamWriter μπορούν να χρησιμοποιηθούν ως γέφυρες
  - Κατασκευαστής: *InputStreamReader(InputStream in)*

Ρεύματα για χαρακτήρες (2)

- LineNumberReader
  - Κατασκευαστής: *LineNumberReader(Reader in)*
  - *Μέθοδοι: readLine()* : διαβάζει μια γραμμή σε String
    getLineNumber(): επιστρέφει τον αριθμό γραμμής
- *PrintWriter*
  - Κατασκευαστές: *PrintWriter(File file)*
    PrintWriter(OutputStream out)
    PrintWriter(String fileName)
  - *Μέθοδοι: print(float f), print(String s), println(Object o)*
    write(String s)

ΠΡΟΣΟΧΗ: Όταν δε χρειαζόμαστε πλέον ένα stream, ένα reader ή writer τον κλείνουμε με τη μέθοδο ***close()***

Συνδυασμός ρευμάτων χαρακτήρων



Ρεύματα και αρχεία

Η κλάση File

- Είναι η βασική κλάση για **αρχεία *και* φακέλους**
- Μέθοδοι της File
  - `exists`: ελέγχει αν υπάρχει το αρχείο
  - `canRead`: ελέγχει αν μπορούμε να διαβάσουμε από το αρχείο
  - `canWrite`: ελέγχει αν μπορούμε να γράψουμε στο αρχείο
  - `delete`: διαγράφει το αρχείο και επιστρέφει true (αν πετύχει)
  - `length`: επιστρέφει το μέγεθος του αρχείου σε bytes
  - `getName`: επιστρέφει το όνομα του αρχείου (μόνο)
  - `getPath`: επιστρέφει το πλήρες μονοπάτι του αρχείου

```
File numFile = new File(“numbers.txt”);
if (numFile.exists())
  System.out.println(numfile.length());
```

Αρχεία Text και Binary

- Αρχεία Text: περιέχουν τα δεδομένα σε μορφή εκτυπώσιμων χαρακτήρων
  - Ένα byte για κάθε χαρακτήρα (για την κωδικοποίηση ASCII)
  - Δύο bytes για κάθε χαρακτήρα (character) (για την κωδικοποίηση Unicode, και για πολλές γλώσσες)
  - Μπορούμε να διαβάσουμε τα αρχεία από ένα "text editor"
- Αρχεία Binary: περιέχουν κωδικοποιημένα δεδομένα, όπως εντολές προς εκτέλεση ή αριθμητικά δεδομένα
  - Δεν είναι κατάλληλα για εκτύπωση
  - Μπορούν να διαβαστούν από υπολογιστές αλλά όχι από ανθρώπους
  - Είναι πιο εύκολο για ένα πρόγραμμα να τα χειριστεί.

Ρεύματα εισόδου/εξόδου για κείμενο

- Κλάσεις για έξοδο (σε αρχείο):
  - PrintWriter
  - FileOutputStream [ή FileWriter]
- Κλάσεις για είσοδο (από αρχείο):
  - BufferedReader
  - FileReader
- Η FileOutputStream και η FileReader δέχονται ως όρισμα ένα όνομα αρχείου
- Η PrintWriter και η BufferedReader έχουν χρήσιμες μεθόδους για γραφή και ανάγνωση

Ρεύματα αρχείων

java.lang.Object
java.io.InputStream
java.io.FileInputStream

Constructors

Constructor and Description
`FileInputStream(File file)` Creates a `FileInputStream` by opening a connection to an actual file, the file named by the `File` object `file` in the file system.
`FileInputStream(FileDescriptor fdObj)` Creates a `FileInputStream` by using the file descriptor `fdObj`, which represents an existing connection to an actual file in the file system.
`FileInputStream(String name)` Creates a `FileInputStream` by opening a connection to an actual file, the file named by the path name `name` in the file system.

Constructors

Constructor and Description
`FileOutputStream(File file)` Creates a file output stream to write to the file represented by the specified `File` object.
`FileOutputStream(File file, boolean append)` Creates a file output stream to write to the file represented by the specified `File` object.
`FileOutputStream(FileDescriptor fdObj)` Creates a file output stream to write to the specified file descriptor, which represents an existing connection to an actual file in the file system.
`FileOutputStream(String name)` Creates a file output stream to write to the file with the specified name.
`FileOutputStream(String name, boolean append)` Creates a file output stream to write to the file with the specified name.

Παράδειγμα

```
File smileyFile = new File(“smiley.txt”);

FileInputStream smileyInStream = new
  FileInputStream(smileyFile);

FileDescriptor fileID = smileyFile.getFD();

FileInputStream anotherFile = new FileInputStream
  (fileID);

if (smileyFile.canRead()){
…
}
```

Διάβασμα από αρχεία

- Η τάξη File (στο πακέτο java.io.*) αντιπροσωπεύει:
  - Ένα αρχείο:
    File arxeio = new File("c:\\1.txt");
  - Ένα κατάλογο:
    File katalogos = new File(".");
    String path=katalogos.getAbsolutePath();
    File[] files=katalogos.listFiles();
- Η τάξη FileReader (FileWriter) δημιουργεί ένα ρεύμα εισόδου (εξόδου) από αρχείο:
  - FileReader arxeio = new FileReader("c:\\1.txt"));

Ανάγνωση - Εγγραφή

- Ανάγνωση: Όπως και με το πληκτρολόγιο συνδέουμε το πρόγραμμά μας με το αρχείο με έναν BufferedReader

```
BufferedReader eisodos = new BufferedReader(new
                                FileReader("c:\\1.txt"));
```

- Εγγραφή: Συνδέουμε το πρόγραμμά μας με το αρχείο με ένα BufferedWriter

```
PrintWriter exodos =new PrintWriter(new BufferedWriter(new
                                    FileWriter(“c:\\copy.txt")));
String s;
while((s = eisodos.readLine()) != null )
  exodos.println(lineCount++   + ": " + s);
exodos.close();
```

Παράδειγμα - Αντιγραφή

```
import java.io.*;
class test{
  public static void main(String args[]){
      try{
        BufferedReader eisodos = new BufferedReader(new
                              FileReader("c:\\autoexec.bat"));
        PrintWriter exodos =new PrintWriter(new BufferedWriter(new
              FileWriter(“c:\\copy.txt")));
        String s;
        while((s = eisodos.readLine()) != null )
              exodos.println(s);
        exodos.close();
      }
      catch (Exception ex){ System.out.println(ex);}
  }
}
```

Binary αρχεία

Σειριακή μεταφορά αντικειμένων

- Όταν θέλουμε να στείλουμε ολόκληρα αντικείμενα σε ένα ρεύμα χρησιμοποιούμε τις τάξεις *ObjectInputStream και ObjectOutputStream*
- Απεικονίζουμε το αντικείμενο με σειριακή μορφή ώστε να μπορούμε να το ανακτήσουμε (serialization)
  - Κατασκευαστής: ObjectOutputStream(OutputStream out)
  - Μέθοδοι: writeInt(int val), writeObject(Object o), flush()
  - Κατασκευαστής: ObjectInputStream(InputStream in)
  - Μέθοδοι: int readInt(), Object writeObject(), flush()
- Η κλάση του αντικειμένου πρέπει να υλοποιεί το Serializable interface

Παράδειγμα

```
try{
    FileOutputStream out = new FileOutputStream(“output.dat");
    ObjectOutputStream s = new ObjectOutputStream(out);
    s.writeObject("Today");
    s.writeObject(new Date());
    s.flush();
    s.close();
    FileInputStream in = new FileInputStream("output.dat");
    ObjectInputStream t=new ObjectInputStream(in);
    String today = (String)t.readObject();
    Date date = (Date)t.readObject();
    t.close();
    System.out.println(today+”\n”+date);
}
catch (Exception e){
    e.printStackTrace();
}
```

`Serializable` Interface

- Ένα αντικείμενο είναι serializable αν η κλάση του υλοποιεί το java.io.Serializable interface.
- Το interface δεν έχει επιπλέον μεθόδους για να υλοποιήσουμε. Είναι marker interface.
- Με τη δήλωση επιτρέπουμε στην Java να αυτοματοποιήσει το μηχανισμό αποθήκευσης των αντικειμένων από/σε αρχεία
- Μπορούμε επίσης να καθορίζουμε τι θα αποθηκεύεται και τι όχι

Προσοχή

- Τι γίνεται αν το αντικείμενο περιέχει γνωρίσματα μη σειριοποιήσιμα;
  - Δεν σειριοποιείται.
- Όμοια και ένας πίνακας αντικειμένων
- Μπορούμε να σειριοποιήσουμε τα υπόλοιπα γνωρίσματα δηλώνονται ως transient αυτά που θα αγνοηθούν

```
public class Foo implements java.io.Serializable {
  private int v1; // σειριοποιείται
  private static double v2;
  private transient A v3 = new A();
}
```

Αρχεία τυχαίας πρόσβασης

- Η τάξη RandomAccessFile δημιουργεί αρχεία στα οποία η ανάγνωση και γραφή δε γίνεται σειριακά αλλά σε οποιαδήποτε θέση.
  - Κατασκευαστής:

    RandomAccessFile(String name, String mode)

    RandomAccessFile(File file, String mode)

    mode: “r” μόνο για ανάγνωση, “rw” για ανάγνωση/εγγραφή
  - Μέθοδοι:

    skipBytes(int n) //προχωρά το δείκτη κατά n bytes

    seek(long pos) //πάει το δείκτη στη θέση pos του αρχείου

    long getFilePointer() //επιστρέφει τη θέση του δείκτη

Πώς ελέγχω το τέλος του αρχείου;

- Βάζω έναν ειδικό χαρακτήρα και σταματώ μόλις τον διαβάσω
- Ψάχνω για έναν ειδικό χαρακτήρα στο τέλος του αρχείου (τα text αρχεία έχουν τέτοιο χαρακτήρα)
- Χρησιμοποιώ την κλάση Scanner

  ```
  while (inFile.hasNext())
  while (inFile.next()!=null)
  ```

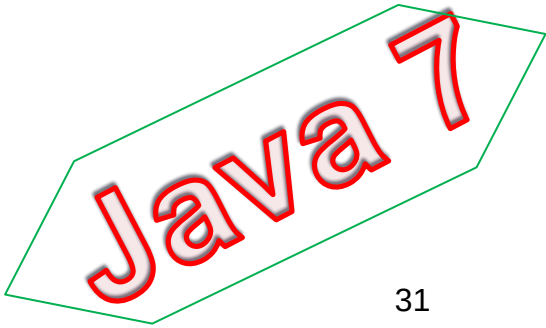
- Πυροδοτώ μια εξαίρεση τέλους-αρχείου και την ανιχνεύω στον κώδικά μου

Java 7

Νέο πακέτο java.nio.file

- Περιέχει κλάσεις και interfaces όπως οι Path, Paths, Files, DirectoryStream, WatchService κλπ.
- Παράδειγμα: αντιγραφή αρχείου:

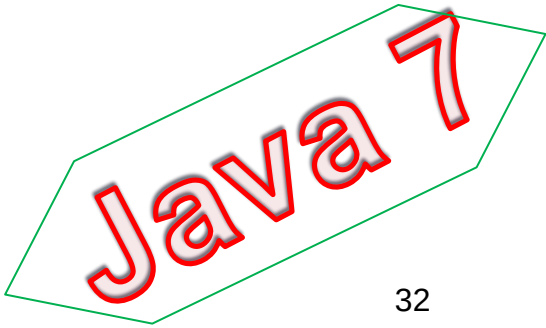
```
Path src = Paths.get("/home/fred/readme.txt");
Path dst = Paths.get("/home/fred/copy_readme.txt");
Files.copy(src, dst,
           StandardCopyOption.COPY_ATTRIBUTES,
           StandardCopyOption.REPLACE_EXISTING);
```



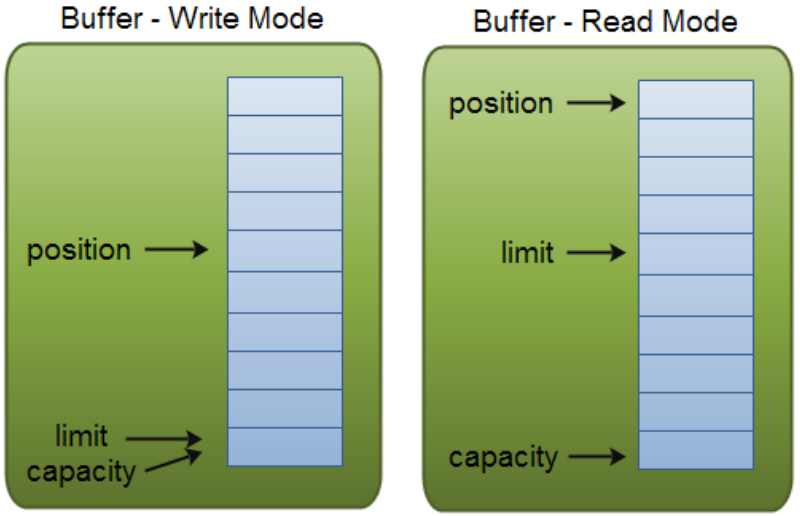
http://cs.brown.edu/courses/cs161/papers/j-nio-ltr.pdf

java.nio.channels

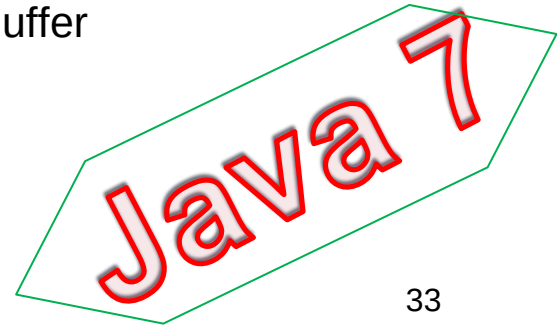
- java.nio.Channel → Stream
  - Read & Write → Read or write
  - Read & Write asynchronously
  - Πάντα επικοινωνούμε μέσω ενός Buffer αντικειμένου
- Implementations
  - FileChannel για αρχεία
  - DatagramChannel για συνδέσεις UDP.
  - SocketChannel για συνδέσεις TCP
  - ServerSocketChannel για ακροατές εισερχόμενων TCP συνδέσεων (web server). Κάθε σύνδεση δημιουργεί ένα SocketChannel



java.nio.Buffer

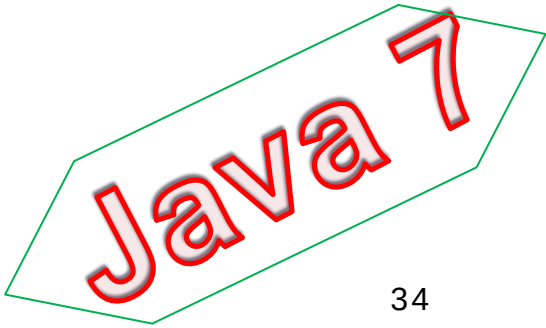


- `Abstract class`
  - Αναλαμβάνει την επικοινωνία με ένα channel
- Implementations
  - ByteBuffer, MappedByteBuffer, CharBuffer, DoubleBuffer, FloatBuffer, IntBuffer, LongBuffer, ShortBuffer
- Methods
  - flip(): switches a Buffer from writing mode to reading mode.
  - rewind() sets the position back to 0, so you can reread all the data in the buffer
  - clear() the position is set back to 0 and the limit to capacity
  - compact() copies all unread data to the beginning of the Buffer
  - Buffer.mark() marks a given position in a Buffer.
  - You can reset to that position by calling the Buffer.reset()



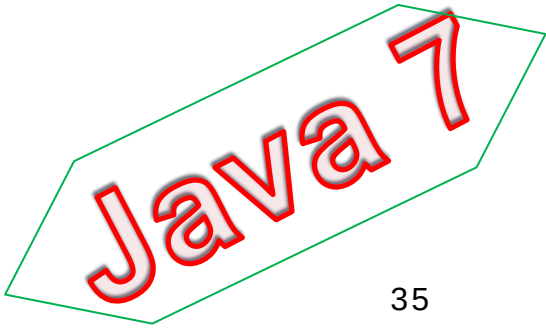
Παράδειγμα - Read

```
RandomAccessFile fin = new RandomAccessFile( "readbytes.txt“, “rw” );
FileChannel fc = fin.getChannel();
ByteBuffer buffer = ByteBuffer.allocate( 1024 );
int bytesRead = inChannel.read(buf); //write from channel to buffer
while (bytesRead != -1) {
    System.out.println("Read " + bytesRead);
    buf.flip(); //prepare buffer for read
    while(buf.hasRemaining()){
        System.out.print((char) buf.get());
    }
    buf.clear();
    bytesRead = inChannel.read(buf);
}
fin.close();
```



Παράδειγμα - Write

```
//WRITE
FileOutputStream fout = new FileOutputStream("writebytes.txt" );
FileChannel fc = fout.getChannel();
ByteBuffer buffer = ByteBuffer.allocate( 1024 );
for (int i=0; i<message.length; i++) {
    buffer.put( message[i] );
}
buffer.flip();  //prepare buffer for read
fc.write( buffer ); //read from buffer to channel
```



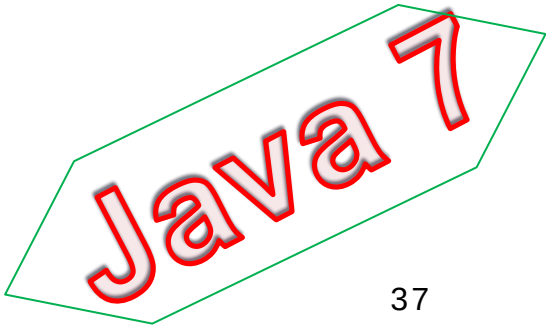
Java 7

Αυτόματη διαχείριση πόρων

```
public void oldTry() {
       try {
         fos = new FileOutputStream(“a.txt");
         dos = new DataOutputStream(fos);
         dos.writeUTF("Java 6");
       } catch (IOException e) {
              e.printStackTrace();
       } finally {
              try {
                  fos.close();
                  dos.close();
                  } catch (IOException e) {
                  // log the exception
              }
          }
       }
```

try-with-resources

```
public void newTry() {
   try (
      FileOutputStream fos = new
                FileOutputStream(“a.txt");
      DataOutputStream dos = new
                DataOutputStream(fos)
      ) {
      dos.writeUTF("Java 7");
   }
   catch (IOException e) {
              // log the exception
   }
}
```



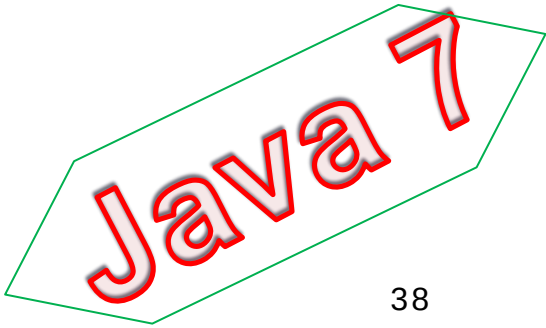
Ο τελεστής <>

- diamond operator
- Αντί για

```
HashMap<String, Stack<String>> map =
    new HashMap<String, Stack<String>>();
```

- Μπορώ να γράψω:

```
HashMap<String, Stack<String>> map =
    new HashMap<>;
```



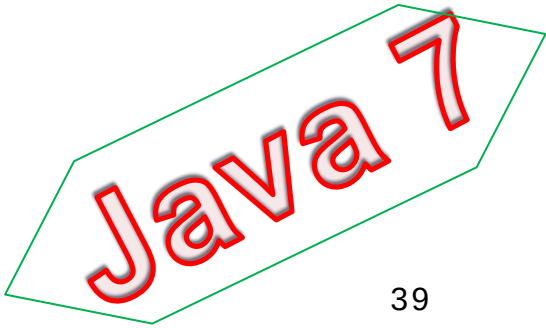
Strings στη `switch`

- Αντί για

```
if (input.equals(“yes”)) {
    return true;
} else if (input.equals(“no”)) {
    return false;
} else {
    askAgain();
}
```

- Γράφουμε

```
switch(input) {
    case “yes”: return true;
    case “no”: return false;
    default: askAgain();
}
```



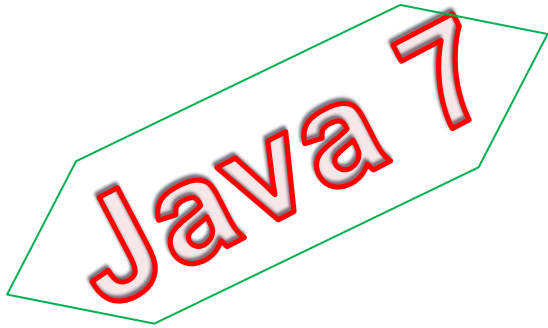
Διαχωριστικά χιλιάδων

- Αντί για

```
int million = 1000000;
```

γράφουμε

```
int million = 1_000_000;
```



Multi-catch

```
public void newMultiMultiCatch() {
    try {
        methodThatThrowsThreeExceptions();
    } catch (ExceptionOne e) {
        // deal with ExceptionOne
    } catch (ExceptionTwo | ExceptionThree e) {
        // deal with ExceptionTwo and ExceptionThree
    }
}
```