Pemrograman Jaringan 4

anton@ukdw.ac.id

Java IO

Java IO

- Java IO dibutuhkan ketika kita **membaca** dan **menulis,** baik ditampilkan pada layar maupun disimpan pada file
- Dalam pemrograman jaringan, Java IO dibutuhkan ketika kita hendak mengirimkan **byte data** maupun membaca **data dari server**
- Input/Output dalam Java dipaketkan dalam `java.io.`
- Selain kelas-kelas, paket ini juga mengandung interface yang menangani aliran (stream) data output dan input.
- Streams are **sequences of data (whose elements may be computed on demand)**

Applications of Streams

- **Streams** are natural models of many real-world systems:
  - Mouse/keyboard/monitor input
  - Human input to a program
  - Contents of a file

Class Stream

- **Byte stream**: kelas dan interface ini digunakan untuk menangani data biner
- **Character stream**: kelompok kelas ini digunakan untuk menangani proses baca tulis karakter, termasuk Unicode.
  - Kelas ini merupakan pengembangan dari kelas Byte Stream sehingga lebih efisien.

Byte Stream: OutputStream

public **abstract** class **OutputStream**

Methodnya:

- public abstract void write(int b) throws IOException
- public void write(byte[] data) throws IOException
- public void write(byte[] data, int offset, int length) throws IOException
- public void flush( ) throws IOException
- public void close( ) throws IOException

Byte Stream: InputStream

public **abstract** class **InputStream**

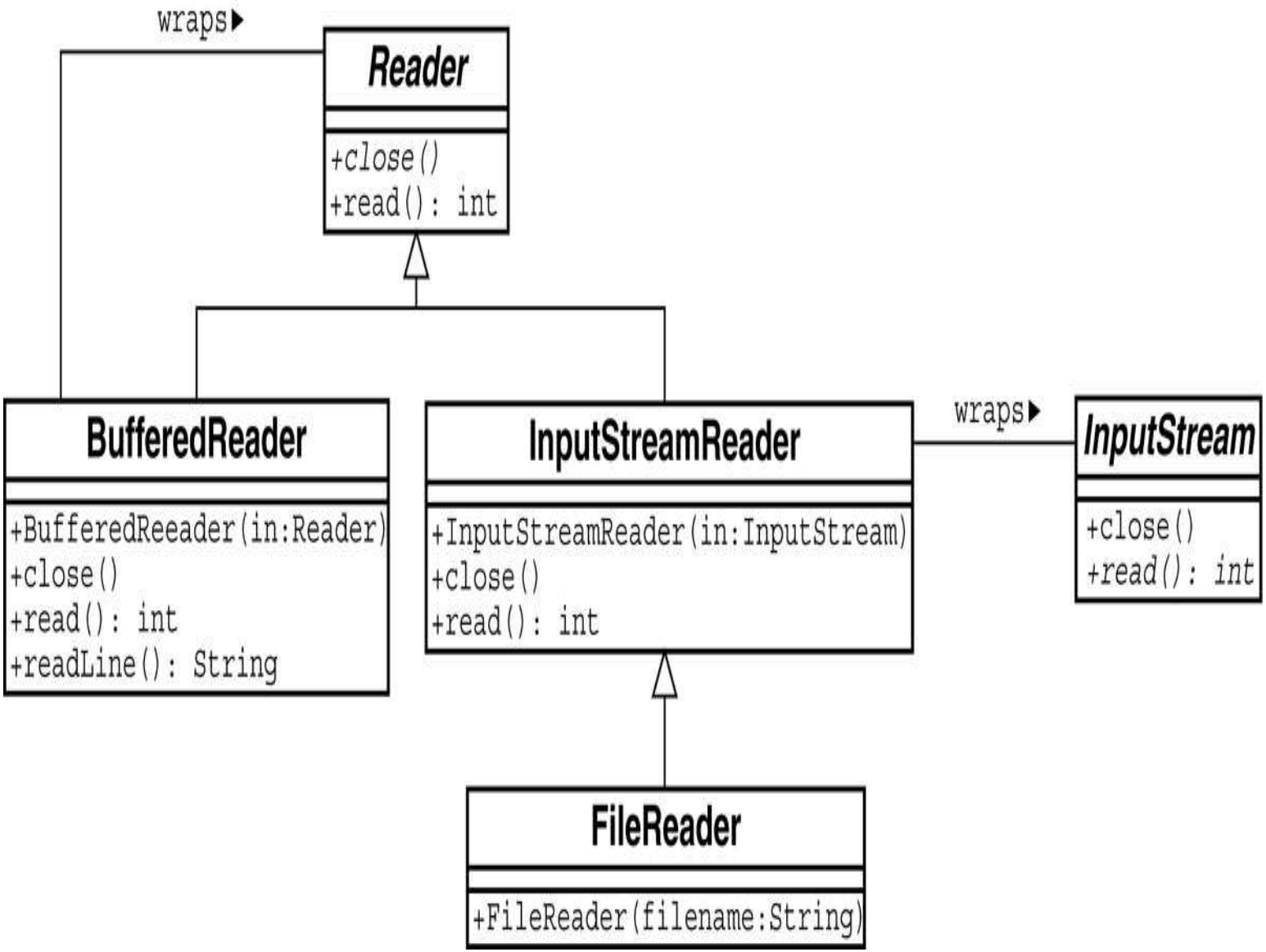
Sedangkan method-methodnya adalah:

- public abstract int read( ) throws IOException
- public int read(byte[] input) throws IOException
- public int read(byte[] input, int offset, int length) throws IOException
- public long skip(long n) throws IOException
- public int available( ) throws IOException
- public void close( ) throws IOException

Hati-hati

- OutputStream dan InputStream adalah kelas **abstract**, sehingga tidak bisa langsung diinstansiasi, harus **diturunkan**!
- Streams can also be buffered, but…

- The **flush**() method breaks the deadlock by forcing the buffered stream to send its data even if the buffer isn't yet full.
- Finally, **close**() it


wraps
Reader
+close()
+read(): int
BufferedReader
+BufferedReeader(in:Reader)
+close()
+read(): int
+readLine(): String
InputStreamReader
+InputStreamReader(in:InputStream)
+close()
+read(): int
wraps
InputStream
+close()
+read(): int
FileReader
+FileReader(filename:String)

Wrapper Classes

- Class W is said to *wrap* class Y if:
  1. Y is a **concrete** (not abstract) class
  2. W's constructor takes Y as an **argument** and stores a local copy of Y
  3. W **reimplements** all of Y's methods

- A wrapper can **wrap a class** and be the **subclass** of another class at the same time

File Input

- Java classes that support file input are found in the `java.io` package
- `FileReader` allows us to open a file for reading
- `BufferedReader` is a wrapper class that provides methods that
  - allow us to treat the file as a stream of characters
  - increases the efficiency of reading
  - allows line-oriented reading

`BufferedReader`

- A type of **`Reader`** that does internal buffering.
  - more efficient.
- Provides everything from **`Reader`** , plus:

`String readLine()`

- reads up to **`'\n'`**, **`'\r'`** (or both).

Attaching a `BufferedReader` to stdin

```
InputStreamReader isr =
  new InputStreamReader(System.in);

BufferedReader bf =
  new BufferedReader(isr);

String foo = bf.readLine();
```

Stream Standard

- Kelas **java.lang.System** berkaitan standar input, output dan error.
- System.in merupakan objek dari InputStream
- System.out dan System.err merupakan objek dari PrintStream.
- Dalam java cara membaca inputan dari keyboard adalah menggunakan **System.in**. Agar mempermudah proses maka obyek System.in dibungkus dengan obyek BufferedReader
- **BufferedReader br = new BufferedReader(new InputStreamReader(System.in));**

Strings

- Java provides a number of methods for operating on `String` objects
- `String` objects are **immutable**
- Immutable objects cannot be changed once they are created

```
String s = "ABC";
s.toLowerCase();

s = s.toLowerCase();
```

StringBuffer

- Java provides a **mutable** staring class called `StringBuffer` that allows strings to grow dynamically during program execution
- Several `StringBuffer` methods are the same as those found in `String`
- The `StringBuffer` class also contains a `ToString` method to allow easier output
- Some Method:
  - **capacity**()
  - **setCharAt**(int index, char ch)
  - **insert**(int offset, char c)
  - **delete**(int start, int end)
  - **replace**(int start, int end, String str)
  - **reverse**()
  - **append**(String str)

String vs StringBuffer

String str = new String ("Stanford ");
str += "Lost!!";

```
0 new #7 <Class java.lang.String>
3 dup
4 ldc #2 <String "Stanford ">
6 invokespecial #12 <Method java.lang.String(java.lang.String)>
9 astore_1
10 new #8 <Class java.lang.StringBuffer>
13 dup
14 aload_1
15 invokestatic #23 <Method java.lang.String valueOf(java.lang.Object)>
18 invokespecial #13 <Method java.lang.StringBuffer(java.lang.String)>
21 ldc #1 <String "Lost!!">
23 invokevirtual #15 <Method java.lang.StringBuffer append(java.lang.String)>
26 invokevirtual #22 <Method java.lang.String toString()>
29 astore_1
```

StringBuffer str = new StringBuffer ("Stanford ");
str.append("Lost!!");

```
0 new #8 <Class java.lang.StringBuffer>
3 dup
4 ldc #2 <String "Stanford ">
6 invokespecial #13 <Method java.lang.StringBuffer(java.lang.String)>
9 astore_1
10 aload_1
11 ldc #1 <String "Lost!!">
13 invokevirtual #15 <Method java.lang.StringBuffer append(java.lang.String)>
16 pop
```

Baca Karakter

```
import java.io.*;

class BacaKarakter{
    public static void main(String[] args){
        char c;
        try{
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
            System.out.println("Masukkan karakter (akhiri dengan \"q\") : ");
            do {
                c = (char) br.read();
                System.out.println("Karakter terbaca : "+c);
            } while (c!='q');
        } catch(IOException e){
            System.out.println("Ada error IO");
            System.exit(0);
        }
    }
}
```

Baca String

```
import java.io.*;

class BacaString{
    public static void main(String[] args){
        String str;
        try{
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
            System.out.println("Masukkan string (akhiri dengan \"end\") : ");
            do {
                str = br.readLine();
                System.out.println("Kata terbaca : "+str);
            } while (str.equalsIgnoreCase("end")==false);
        } catch(IOException e){
            System.out.println("Ada error IO");
            System.exit(0);
        }
    }
}
```

Sequential Files

- Files are stored on disks
- In this section we will assume that files consist of multiple lines composed of characters
- Each line ends with an end of line character
- The file itself may have an end of file character
- Programmers often need to read or write files stored on disks

Class File

- A File object can refer to either a file or a directory

  File file1 = new File("data.txt");
  File file1 = new File("C:\java");

- To obtain the path to the current working directory use:

  System.getProperty("user.dir");

- To obtain the file or path separator use

  System.getProperty("file.separator");
  System.getProperty("path.separator");
  or
  File.separator() ;
  File.pathSeparator() ;

Useful File methods

- public boolean **canRead()**
- public boolean **canWrite()**
- public boolean **createNewFile()**
- public boolean **delete()**
- public boolean **exists()**
- public boolean **isFile()**
- public boolean **isDirectory()**
- public long **lastModified()**
- public long **length()**
- public boolean **mkdir()**
- public boolean **mkdirs()**
- public boolean **renameTo(File newfilename)**

Contoh 1 – property file

```
import java.io.*;
class cobaFile{
	public static void main(String[] args) throws IOException {
	if(args.length != 1){
		System.out.println("Usage : java cobaFile <filename>");
		System.exit(1);
	}
	File f = new File(args[0]);
	System.out.println("Nama : "+f.getName());
	if(f.exists()) System.out.println("File sudah ada!");
	else {
		System.out.println("File belum ada, buat baru..");
		if(f.createNewFile()) System.out.println("Pembuatan selesai");
		else {
			System.out.println("Pembuatan gagal!");
			System.exit(0);
		}
	}
	System.out.println("File dapat dibaca ? "+f.canRead());
	System.out.println("File dapat ditulisi ? "+f.canWrite());
	System.out.println("File adalah file ? "+f.isFile());
	System.out.println("File adalah direktori ? "+f.isDirectory());
	java.util.Date d = new java.util.Date(f.lastModified());
	System.out.println("Last modified : "+d.toString());
	System.out.println("File size : "+f.length());
	}
}
```

Contoh 2 – Rename

```
import java.io.*;
class cobaFile2{
      public static void main(String[] args) {
            if(args.length != 2){
                  System.out.println("Usage : java cobaFile2 <source> <dest>");
                  System.exit(1);
            }
            File f = new File(args[0]);
            if(f.renameTo(new File(args[1]))) System.out.println("Success!");
            else System.out.println("Failed!");
      }
}
```

Contoh 3 - Direktori

```
import java.io.*;
class cobaFile3{
      public static void main(String[] args) {
            if(args.length != 1){
                  System.out.println("Usage : java cobaFile3 <dir>");
                  System.exit(1);
            }
            File f = new File(args[0]);
            if(f.mkdir()) System.out.println("Success!");
            else System.out.println("Failed!");
      }
}
```

Contoh 4 – Read file

```
import java.io.*;
class cobaFile4{
	public static void main(String[] args) {
		if(args.length != 1){
			System.out.println("Usage : java cobaFile4 <filetoread>");
			System.exit(1);
		}
		try{
			FileReader f = new FileReader(args[0]);
			BufferedReader r = new BufferedReader(f);
			String s = null;
			while((s=r.readLine())!=null){
				System.out.println(s);
			}
			r.close();
			f.close();
		} catch(FileNotFoundException e){
			System.out.println("File not found!");
			System.exit(1);
		} catch(IOException e){
			System.out.println("IO Error!");
			System.exit(1);
		}
	}
}
```

Contoh 5 – Write file

```
import java.io.*;
class cobaFile5{
      public static void main(String[] args) {
            if(args.length != 1){
                  System.out.println("Usage : java cobaFile5 <filetowrite>");
                  System.exit(1);
            }
            try{
                  FileWriter f = new FileWriter(args[0]);
                  BufferedWriter r = new BufferedWriter(f);
                  String s = "percobaan menulis 1 2 3 4 sukses!";
                  r.write(s);
                  r.close();
                  f.close();
            } catch(IOException e){
                  System.out.println("IO Error!");
                  System.exit(1);
            }
      }
}
```

DirListing Example

```
import java.io.*;

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

    File dir = new File(System.getProperty("user.dir"));

    if(dir.isDirectory()){
      System.out.println("Directory of " + dir);
      String[] listing = dir.list();
      for(int i=0; i<listing.length; i++) {
        System.out.println("\t" + listing[i]);
      }
    }
  }
}
```

FileOutput with Encoding

```
import java.io.*;

public class CharacterFileOutput {
  public static void main(String[] args) {
    FileWriter out = null;

    try {
      out = new FileWriter("book.txt");
      System.out.println("Encoding: " + out.getEncoding());
      out.write("Core Web Programming");
      out.close();
      out = null;
    } catch(IOException ioe) {
      System.out.println("IO problem: " + ioe);
      ioe.printStackTrace();
      try {
        if (out != null) {
          out.close();
        }
      } catch(IOException ioe2) { }
    }
  }
```

Encoding

To change the system default encoding use

- System.setProperty("file.encoding", "encoding");

- To specify the encoding when creating the output steam, use an OutputStreamWriter

```
OutputStreamWriter out = new
OutputStreamWriter( new
FileOutputStream("book.txt", "8859_1"));
```

FileInput with Encoding

```
import java.io.*;

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

    File file = new File("book.txt");
    FileReader in = null;

    if(file.exists()) {
      try {
        in = new FileReader(file);
        System.out.println("Encoding: " + in.getEncoding());
        char[] buffer = new char[(int)file.length()];
        in.read(buffer);
        System.out.println(buffer);
        in.close();
      } catch(IOException ioe) {
        System.out.println("IO problem: " + ioe);
        ioe.printStackTrace();
        ...
      }
    }
  }
```

Alternative Reading File

```
BufferedReader in = new
BufferedReader(new FileReader(file));
String lineIn;
while ((lineIn = in.readLine()) != null) {
    System.out.println(lineIn);
}
```

BinaryFileOutput

```
import java.io.*;

public class BinaryFileOutput {

   public static void main(String[] args) {
     int[] primes = { 1, 2, 3, 5, 11, 17, 19, 23 };
     DataOutputStream out = null;

     try {
       out = new DataOutputStream(
                 new FileOutputStream("primes.bin"));

       for(int i=0; i<primes.length; i++) {
         out.writeInt(primes[i]);
       }
       out.close();
     } catch(IOException ioe) {
        System.out.println("IO problem: " + ioe);
        ioe.printStackTrace();
     }
   }
}
```

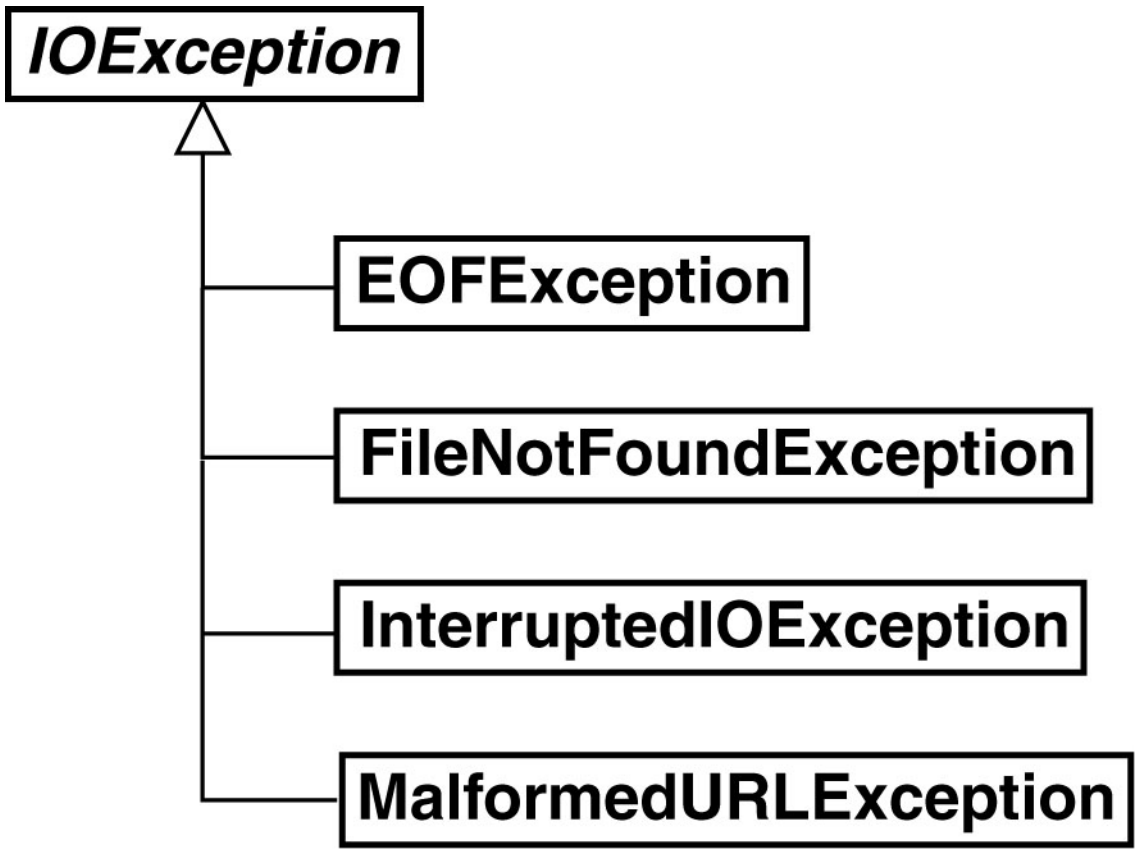
BinaryFileInput

```
import java.io.*;

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

    DataInputStream in = null;
    File file = new File("primes.bin");
    try {
      in = new DataInputStream(
               new FileInputStream(file));
      int prime;
      long size = file.length()/4;  // 4 bytes per int
      for(long i=0; i<size; i++) {
        prime = in.readInt();
        System.out.println(prime);
      }
      in.close();
    } catch(IOException ioe) {
       System.out.println("IO problem: " + ioe);
       ioe.printStackTrace();
    }
  }
```

Copy File

```
File fromFile = new File(fromFileName);
File toFile = new File(toFileName);

FileInputStream from = null;
FileOutputStream to = null;

from = new FileInputStream(fromFile);
to = new FileOutputStream(toFile);

byte[] buffer = new byte[4096];
int bytesRead;

while ((bytesRead = from.read(buffer)) != -1)
        to.write(buffer, 0, bytesRead); // write

System.out.println("File Copied.");
```

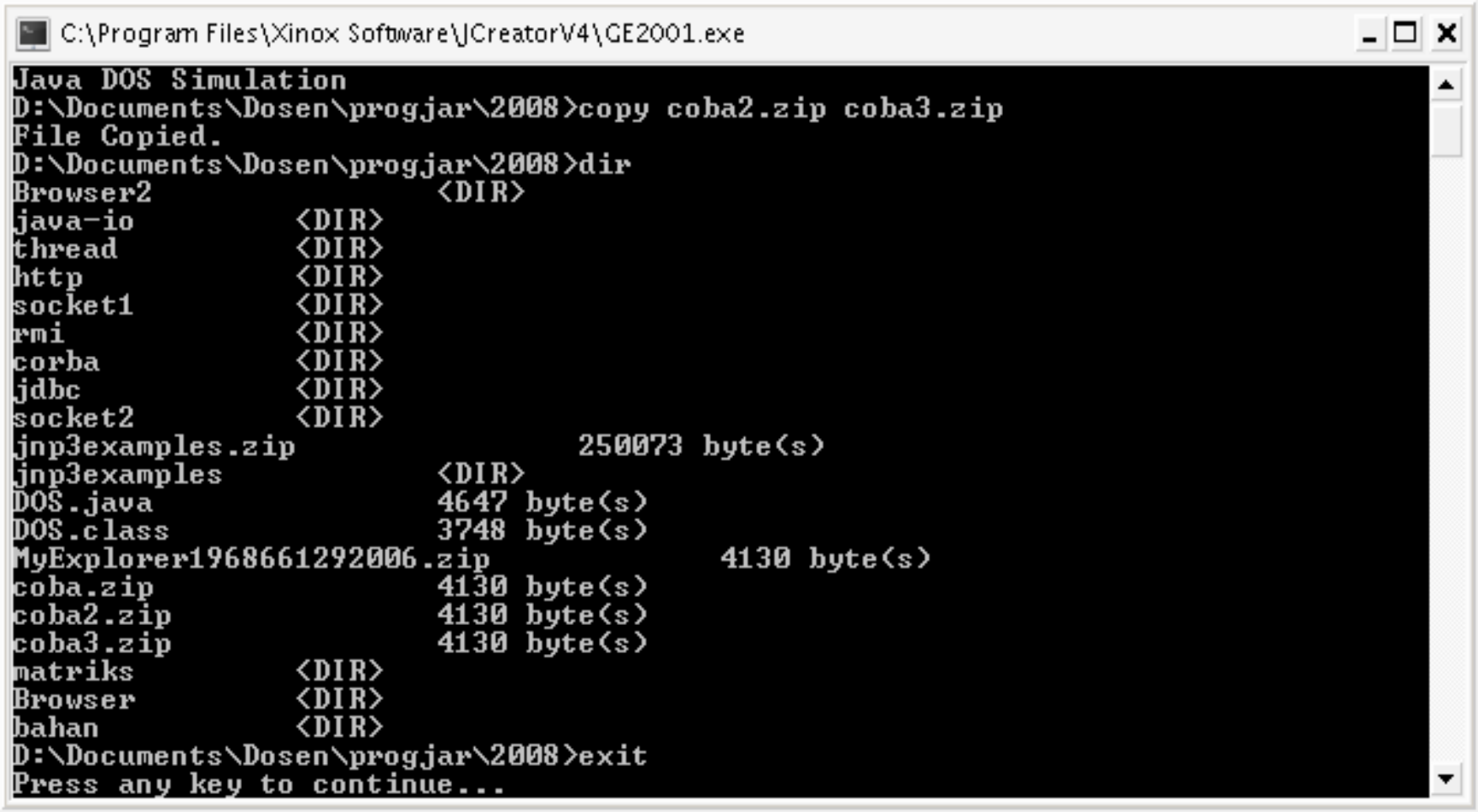
```
import java.io.*;
import java.util.StringTokenizer;

public class WordCount {
    public static void main( String[] args ) throws IOException {
        String delimiters = " .?!()[]{}|?/&\\,;:-\'\"\t\n\r";

        BufferedReader inputFile = new BufferedReader( new FileReader( args[0] ) );

        String buffer = null;
        int    chars  = 0;
        int    words  = 0;
        int    lines  = 0;

        while( true ) {
            buffer = inputFile.readLine();

            if ( buffer == null ) break;

            lines++;

            buffer = buffer.toLowerCase();
            StringTokenizer tokens = new StringTokenizer( buffer, delimiters );

            while( tokens.hasMoreElements() ) {
                String word = tokens.nextToken();
                words++;
                chars += word.length();
            } // end while
        } // end while( true )...

        System.out.println( "" + lines + " " + words + " " + chars );
    } // end main
} // end class WordCount
```


IOException
EOFException
FileNotFoundException
InterruptedIOException
MalformedURLException

Tugas

- Buatlah “DOS PROMPT” simulation
- Berisi beberapa fungsi manipulasi file yang sudah diajarkan:
  - Dir <dir>
  - Copy <source> <dest>
  - Rename <source> <dest>
  - Move <source> <dest>
  - Properties <filename>
  - Date-Time
  - Make <filename>
  - Write <filename>
  - Del <filename>
  - Read <filename>
  - Mkdir <dirname>
  - Find <filename> in active dir

Contoh


C:\Program Files\Xinox Software\JCreatorV4\GE2001.exe
Java DOS Simulation
D:\Documents\Dosen\progjar\2008>copy coba2.zip coba3.zip
File Copied.
D:\Documents\Dosen\progjar\2008>dir
Browser2 <DIR>
java-io <DIR>
thread <DIR>
http <DIR>
socket1 <DIR>
rmi <DIR>
corba <DIR>
jdbc <DIR>
socket2 <DIR>
jnp3examples.zip 250073 byte(s)
jnp3examples <DIR>
DOS.java 4647 byte(s)
DOS.class 3748 byte(s)
MyExplorer1968661292006.zip 4130 byte(s)
coba.zip 4130 byte(s)
coba2.zip 4130 byte(s)
coba3.zip 4130 byte(s)
matriks <DIR>
Browser <DIR>
bahan <DIR>
D:\Documents\Dosen\progjar\2008>exit
Press any key to continue...