

CSE 8B—Intro to CS: Java

Winter, 2006

February 1 (Day 8)
Text Files

Instructor: Neil Rhodes

Filesystem

Files are located in **directories**

Directories can contain

- Files
- Other directories

Files are of two types

- Text: contains human-readable text
 - Sometimes different from machine to machine
 - Line endings often different
 - Usually only contain printable ASCII characters
 - 'A' is the byte 65 in ASCII, space is the byte 32
 - Example:
- Binary: non-human readable
 - A sequence of bytes
 - Identical from machine to machine
 - Sometimes hard to read on other machines, though
 - Examples:

Book says text files same,
binary files different

2

Java classes dealing with files

File: object that represents a file or directory

- Used to manipulate files (as a whole) and directories
- Functionality:
 - Create
 - Delete
 - Rename
 - Find what's in a directory
 - Find information about files/directories

Other classes

- Used to manipulate *contents* of files
 - We'll see in great detail later this lecture and next

3

Example use of File

Create a file in the current directory named "foo"

```
File f = new File("foo");
try {
    if (f.createNewFile()) {
        // it worked
    } else {
        // the file already existed
    }
} catch (IOException e) {
    // there was an Input/Output error
}
```

Get list of contents of the "/tmp" directory

```
File tmpDir = new File("/tmp");
File []contents = tmpDir.listFiles();
for (int i = 0; i < contents.length; i++)
    System.out.println(contents[i]);
```

Alternate for loop syntax: `for (File f: contents)`
`System.out.println(f);`

4

Important methods in File

```
File(File parent, String child)
File(String pathname)
File(String parent, String child)
File(URI uri)
boolean createNewFile()
boolean delete()
boolean exists();
boolean isDirectory();
boolean isFile();
File[] listFiles();
boolean mkdir();
boolean renameTo();
```

5

Pathnames

How to separate parts of a directory?

- Unix: / (forward slash)
- Windows: \ (backward slash)
- Mac OS 9: : (colon)

Solution provided by File:

- `public static final char separatorChar`
 - On Unix is a `'/'`
 - On Windows is a `'\'`
 - On Mac OS 9 is a `':'`

6

Streams

In Java, we use **streams** to do input and output (I/O)

An input stream is a stream of bytes that comes *into* the program

- For example, `System.in`

An output stream is a stream of bytes that goes *out of* the program

- For example, `System.out` or `System.err`
- Normally, errors are sent to `System.err` and standard output is sent to `System.out`
- That way, when running a program from the command line sending output to a file
 - `java MyProgram > outputFile`
- errors are still sent to the screen

7

Input streams

Input streams provide a sequence of bytes from somewhere

- `InputStream` is abstract
 - methods to read a byte, or sequence of bytes, or skip some bytes
- Concrete subclasses:
 - `FileInputStream` provides stream of bytes from a file
 - Constructed with a `File` or filename
 - Construction can throw an exception
 - `StringBufferInputStream` reads from a string
 - `SequenceInputStream` reads from a set of other streams. It reads all from the first stream, then all from the second, etc.

The `Scanner` class allows reading different types from an input stream.

- It must be constructed with an input stream

8

Example using Scanner

Copies all lines from an input file to System.out

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

class ReadFile {
    public static void main(String[] args) {
        Scanner f = null;
        try {
            f = new Scanner(new FileInputStream(args[0]));
        } catch (FileNotFoundException e) {
            System.err.println("can't open file: " + e.getMessage());
            System.exit(1);
        }
        while (f.hasNextLine())
            System.out.println(f.nextLine());
    }
}
```

9

Important Scanner methods

next... methods will throw an exception if they can't work

```
String nextLine()
Boolean nextBoolean()
Byte nextByte()
int nextInt()
```

hasNext... methods are there to tell whether the next method will work

```
Boolean hasNextLine()
Boolean hasNextBoolean()
Boolean hasNextByte()
Boolean hasNextInt()

void close()
```

10

Output streams

Output streams take a stream of bytes and put them somewhere

- **OutputStream** is abstract
 - Only has methods to write one or multiple bytes
- Concrete subclasses:
 - **FileOutputStream** writes a stream of bytes to a file
 - Constructed with a File or filename
 - Construction can throw an exception
 - **ByteArrayOutputStream** writes to an array of bytes
 - **FilterOutputStream** writes to another stream, but makes changes along the way.

The **PrintWriter** class allows writing different types to an output stream.

- It must be constructed with an output stream

11

Example using PrintWriter

Copies all lines from a System.in to an output file

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

class Echo {
    public static void main(String [] args) {
        PrintWriter p = null;

        try {
            p = new PrintWriter(new FileOutputStream(args[0]));
        } catch (FileNotFoundException e) {
            System.err.println("error creating outfile: " +
                e.getMessage());
            System.exit(1);
        }
        Scanner scan = new Scanner(System.in);
        while (scan.hasNextLine()) {
            p.println(scan.nextLine());
        }
        p.close();
    }
}
```

12

Important PrintWriter methods

Methods don't throw exceptions, instead set an error flag

- retrieve the error with `checkError()`

Print methods also have `println` equivalents

```
void print(Boolean b)
void print(char c)
void print(String s)
void print(Object o)
void print(int i)
```

```
void close()
```

```
void flush()
```

13

Buffering

Without buffering, each call to `print` must call to operating system, which would then write to the file

With buffering, each call to `print` just writes into a *buffer*. When you call `println`, it calls the operating system to write all the bytes in the buffer which are then written to the file.

- Faster because calls the operating system fewer times

The buffer is also flushed

- on `close`
- on explicit `flush`

14

FileOutputStream

What if the file already exists?

- Then it is *truncated* (made 0 size)

If you want to append to a file

- Use the `FileOutputStream` constructor with 2 parameters
 - `FileOutputStream(String filename, Boolean append)`
 - `FileOutputStream(File f, Boolean append)`

15