

## Objectives

- Streams

Sept 29, 2008

Sprenkle - CS209

1

## Review

- What are the two types of *exceptions*?
- What are the two ways to deal with exceptions?
- What are the benefits of exceptions?
- What principle of Java do files break if we're not careful?

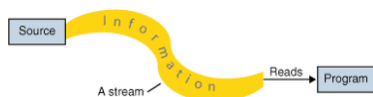
Sept 29, 2008

Sprenkle - CS209

2

## Streams

- Java handles input/output using **streams**, which are sequences of bytes



**input stream:** an object from which we can read a **sequence** of bytes

Abstract class: `java.io.InputStream`

Sept 29, 2008

Sprenkle - CS209

3

## Streams

- Java handles input/output using **streams**, which are sequences of bytes



**output stream:** an object to which we can write a **sequence** of bytes

Abstract class: `java.io.OutputStream`

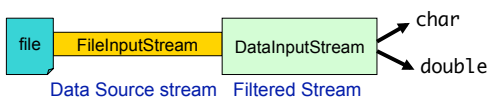
Sept 29, 2008

Sprenkle - CS209

4

## Connected Streams

- Think of a stream as a "pipe"
- `FileInputStream` knows how to read from a file
- `DataInputStream` knows how to read an `InputStream` into useful types
- Connect the **out** end of the `FileInputStream` to the **in** end of the `DataInputStream`...



Sept 29, 2008

Sprenkle - CS209

5

## Filtered Streams vs Data Source Streams

- Subclasses of `FilterInputStream` or `FilterOutputStream`
- Always contains another stream
- Adds functionality to other stream
  - Automatically buffered IO
  - Automatic compression
  - Automatic encryption
  - Automatic conversion between objects and bytes
- As opposed to **Data source streams**
  - communicate with a data source
    - file, byte array, network socket, or URL

Sept 29, 2008

Sprenkle - CS209

6

## Filtered Streams: Reading from a file

- If we wanted to read numbers from a file
  - `FileInputStream` reads bytes from file
  - `DataInputStream` handles numeric type reading
- Connect the `DataInputStream` to the `FileInputStream`
  - `FileInputStream` gets the bytes from the file and `DataInputStream` reads them as assembled types

```
FileInputStream fin = new
    FileInputStream("chicken.data");
DataInputStream din = new
    DataInputStream(fin); "wrap" fin in din
double num1 = din.readDouble();
```

Sept 29, 2008

Sprenkle - CS209

PetSurvey.java

7

## Buffered Streams

- Use a `BufferedInputStream` class object to buffer your input streams
  - A pipe in the chain that adds buffering
  - Speeds up access

```
DataInputStream din = new DataInputStream (
    new BufferedInputStream (
        new FileInputStream("chicken.data")));
```

Sept 29, 2008

Sprenkle - CS209

8

## A More Connected Stream



- `FileInputStream` reads bytes from the file
- `BufferedInputStream` buffers bytes
  - speeds up access to the file
- `DataInputStream` reads buffered bytes as types

Sept 29, 2008

Sprenkle - CS209

9

## Connected Streams

- Combine the many different types of streams to can get the functionality you want
- Similar for output
  - For buffered output to the file and to write types
    - Create a `FileOutputStream`
    - Attach a `BufferedOutputStream`
    - Attach a `DataOutputStream`
    - Perform typed writing using the methods of the `DataOutputStream` object

Sept 29, 2008

Sprenkle - CS209

10

## Text Streams

- Previous streams: operate on *binary* data, not text
- Java uses Unicode to represent characters /strings and some operating systems do not
  - Need something that converts characters from Unicode to whatever encoding the underlying operating system uses
  - Luckily, this is mostly hidden from you

Sept 29, 2008

Sprenkle - CS209

11

## Text Streams

- Derived from `Reader` and `Writer` classes
  - `Reader` and `Writer` generally refer to text I/O
- Ex: Make an input reader that reads from keyboard
  - Create an object of type `InputStreamReader`

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

- `in` reads characters from keyboard and converts them into Unicode for Java

Sept 29, 2008

Sprenkle - CS209

12

## Text Streams and Encodings

- Attach an `InputStreamReader` to a `FileInputStream` to read from a text file:

```
InputStreamReader in = new InputStreamReader(
    new FileInputStream("employee.data"));
```

- Assumes file has been encoded in the default encoding of underlying OS
- You can specify a different encoding in constructor of `InputStreamReader`...

```
InputStreamReader in = new InputStreamReader(
    new FileInputStream("employee.data"), "ASCII");
```

Sept 29, 2008

Sprenkle - CS209

13

## Convenience Classes

- Reading and writing to text files is common
- 2 convenience classes **combine** a `InputStreamReader` with a `FileInputStream`
  - Similar for output of text file

- For example,

```
FileWriter out = new FileWriter("output.txt");
```

is equivalent to

```
OutputStreamWriter out = new OutputStreamWriter(
    new FileOutputStream("output.txt"));
```

Sept 29, 2008

Sprenkle - CS209

14

## PrintWriters

- When writing text output, use `PrintWriter`
  - Easiest writer to use
  - Very similar to a `DataOutputStream`
- Methods: `print()` and `println()`
  - Similar to `System.out` to display strings

Sept 29, 2008

Sprenkle - CS209

15

## PrintWriters Example

```
PrintWriter out = new PrintWriter("output.txt");

String myName = "Homer Simpson";
double mySalary = 35700;

out.print(myName);
out.print(" makes ");
out.print(salary);
out.println(" per year.");
or
out.println(myName + " makes " + salary +
    " per year.");
```

Sept 29, 2008

Sprenkle - CS209

16

## PrintWriters and Buffering

- `PrintWriters` are *always* buffered
- Option: **autoflush** mode
  - Causes any writes to be executed directly on the target destination (in effect defeating the purpose of the buffering)
  - Constructor with second parameter set to true

```
// create an autoflushing PrintWriter
PrintWriter out = new PrintWriter("output.txt",
    true);
```

Sept 29, 2008

Sprenkle - CS209

17

## Formatted Output

- `printf()`
  - **PrintStream** new functionality since Java 1.5

```
double f1=3.14159, f2=1.45, total=9.43;
// simple formatting...
System.out.printf("%6.5f and %5.2f", f1, f2);
// getting fancy (%n = \n or \r\n)...
System.out.printf("%-6s%5.2f\n", "Tax:", total);
```

- Can make formatted output easy
  - Before 1.5, required `java.util.Formatter` objects to generate String passed to `System.out.println()`

Sept 29, 2008

Sprenkle - CS209

18

## Reading Text from a Stream

- There is no `PrintReader` class
- Use a `BufferedReader`
  - Call `readLine()`
    - Reads in a line of text and returns it as a `String`
    - Returns null when no more input is available

Sept 29, 2008

Sprenkle - CS209

19

## Reading Text from a Stream

- Make a `BufferedReader`

➤ Requires a `Reader` object

```
BufferedReader in = new BufferedReader(
    new FileReader("inputfile.txt"));
```

- Read file, line-by-line...

```
String line;
while ((line = in.readLine()) != null) {
    // process the line
}
```

Sept 29, 2008

Sprenkle - CS209

20

## Reading Text from a Stream

- You can also attach a `BufferedReader` to an `InputStreamReader`:

```
BufferedReader in2 = new BufferedReader(
    new InputStreamReader(System.in));
BufferedReader in3 = new BufferedReader(
    new InputStreamReader(url.openStream()));
```

➤ Used to be the best way to read from the console

Sept 29, 2008

Sprenkle - CS209

21

## java.util.Scanner

- New class for handling input
  - Since Java 1.5
- Many constructors
  - Read from file, input stream, string ...

```
Scanner sc = new Scanner(System.in);
```

- Many methods
  - `readNextXXXX` (int, long, line)
  - Skipping patterns, matching patterns, etc.

Sept 29, 2008

Sprenkle - CS209

22

## Using Scanners

- Use `nextXXX()` to read from it...

```
long tempLong;

// create the scanner for the console
Scanner sc = new Scanner(System.in);

// read in an integer and a string
int i = sc.nextInt();
String restOfLine = sc.nextLine();

// read in a bunch of long integers
while (sc.hasNextLong()) {
    tempLong = sc.nextLong();
}
```

Sept 29, 2008

Sprenkle - CS209

23

## Using Scanner

```
public static void main(String[] args) {
    // open the Scanner on the console input, System.in
    Scanner scan = new Scanner(System.in);

    System.out.print("Please enter the width of a square: ");
    int width = scan.nextInt();

    System.out.print("Please enter the height of a square: ");
    int length = scan.nextInt();

    System.out
        .println("The area of your square is " + length * width
            + ".");
}
```

ConsoleIODemo.java

Sept 29, 2008

Sprenkle - CS209

24

## Assignment 7

- Modifying Olympic Score generator
  - Read difficulty score from console
  - Read execution scores from a file
    - Filename comes from console or command-line arguments