INTRODUCTION TO
COMPUTER SCIENCE

Rutgers University

6. Input / Output



http://introcs.cs.rutgers.edu

4. Input and Output

- Standard input and output
- Standard drawing
- Fractal drawings (optional)
- Animation (optional)

DOWNLOAD and ADD to your program folder

- StdIn.java
- StdOut.java
- ints.txt
- names.txt

Canvas—FILES—TREES—CODE
Compile

OR

Go to Course Website—Lectures—Week 5—Download the Input Output Library

- *We need the above two files in our program folder.*

Basic building blocks for programming

any program you might want to write

objects

functions and modules

graphics, sound, and image I/O

arrays

conditionals and loops

Math | text I/O

primitive data types | assignment statements



Ability to interact with the outside world

Review: command-line input

Command-line input. An abstraction for providing arguments (strings) to a program.

Basic properties

- Strings you type after the program name are available as args[0], args[1], ... **at *run* time.**
- Arguments are available when the program *begins* execution.
- Need to call system conversion methods to convert the strings to other types of data.

```
public class RandomInt
{
   public static void main(String[] args)
   {
      int N = Integer.parseInt(args[0]);
      double r = Math.random();
      int t = (int) (r * N);  //t is random int [0,N)
      System.out.println(t);
   }
}
```

```
% java RandomInt 6
3

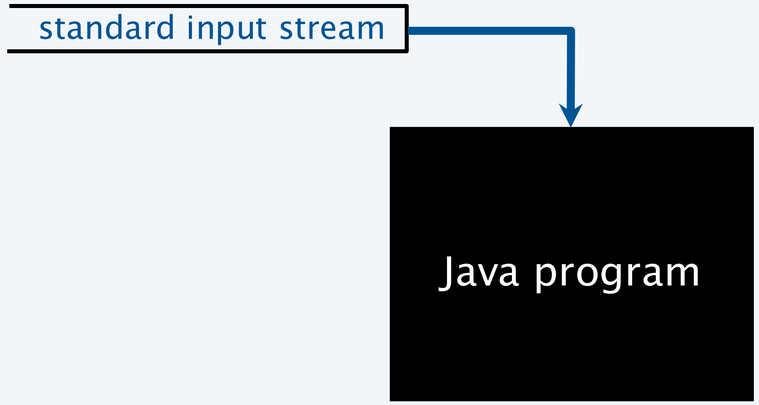
% java RandomInt 10000
3184
```

StdIn library (book version)

Developed for this book, but broadly useful

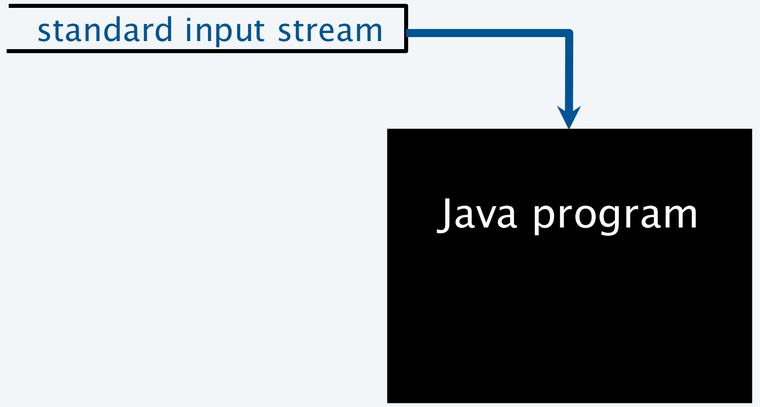
- Implement abstractions invented for UNIX in the 1970s.
- Available for download at book site.
- Included in introcs software you downloaded at the beginning of the course.

public class StdIn	
boolean isEmpty()	true *iff no more values*
int readInt()	*read a value of type* int
double readDouble()	*read a value of type* double
long readLong()	*read a value of type* long
boolean readBoolean()	*read a value of type* boolean
char readChar()	*read a value of type* char
String readString()	*read a value of type* String
String readAll()	*read the rest of the text*



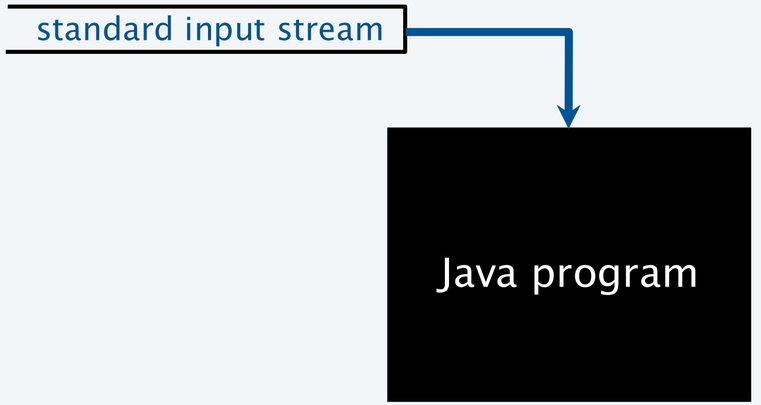
StdIn library (processing chars and Strings)

public class StdIn	
boolean hasNextChar()	true *iff standard input has more characters*
char readChar()	*read a value of type* char and returns
boolean hasNextLine()	true *iff standard input has more input lines*
String readLine()	*read remaining portion of line as type* String
String readAll()	*read remaining portion of input as type String*



StdIn library

public class StdIn	
double[] readAllDoubles()	*read all remaining doubles and return in an array*
int[] readAllInts()	*read all remaining ints and return in an array*
long[] readAllLongs()	*read all remaining longs and return in an array*
String[] readAllStrings()	*read all remaining Strings and return in an array*
String[] readAllLines()	*read all remaining lines and return in an array of Strings*

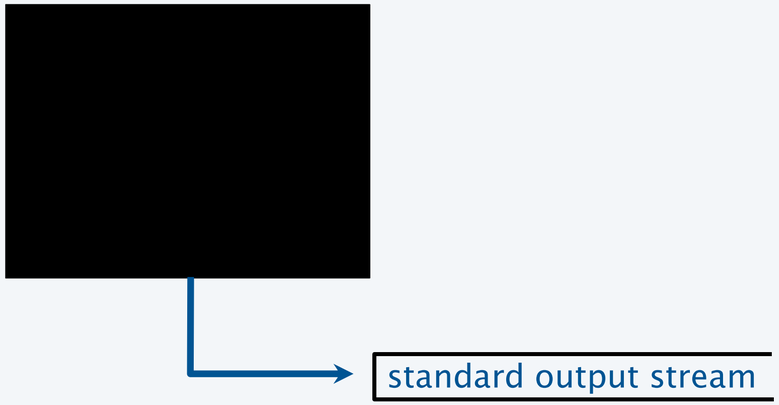


StdOut library

Developed for this course, but broadly useful

- Implement abstractions invented for UNIX in the 1970s.
- Included in introcs software you downloaded at the beginning of the course.

public class StdOut	
void print(String s)	*put s on the output stream*
void println()	*put a newline on the output stream*
void println(String s)	*put s, then a newline on the stream*
void printf(String f, ...)	*formatted output*



Q. These are the same as System.out. Why not just use System.out?

A. We provide a consistent set of simple I/O abstractions in one place.

A. We can make output *independent* of system, language, and locale.

use StdOut from now on

StdIn/StdOut warmup

Interactive input

- Prompt user to type inputs on standard input stream.
- Mix input stream with output stream.

```
public class AddTwo
{
   public static void main(String[] args {
      StdOut.print("Type the first integer: ");
      int x = StdIn.readInt();
      StdOut.print("Type the second integer: ");
      int y = StdIn.readInt();
      int sum = x + y;
      StdOut.println("Their sum is " + sum);
   }
}
```

```
% java AddTwo
Type the first integer:  1
Type the second integer: 2
Their sum is 3
```

StdIn application: average the numbers on the standard input stream

Average

- Read a stream of numbers.
- Compute their average.

Q. How do I specify **the end of the stream**?

A. <Ctrl-d> (standard for decades).

A. <Ctrl-z> (Windows).

Key points

- No limit on the size of the input stream.
- Input and output can be interleaved.

```
public class Average
{
   public static void main(String[] args)
   {
      double sum = 0.0;  // cumulative total
      int n = 0;         // number of values
      while (!StdIn.isEmpty())
      {
         double x = StdIn.readDouble();
         sum = sum + x;
         n++;
      }
      StdOut.println(sum / n);
   }
}
```

```
% java Average
 10.0   5.0     6.0     32.0
  3.0   7.0
<Ctrl-d>
10.5
```

Summary: prototypical applications of standard output and standard input

StdOut: Generate a stream of random numbers

```
public class RandomSeq
{
   public static void main(String[] args)
   {
      int N = Integer.parseInt(args[0]);
      for (int i = 0; i < N; i++)
         StdOut.println(Math.random());
   }
}
```

StdIn: Compute the average of a stream of numbers

```
public class Average
{
   public static void main(String[] args)
   {
      double sum = 0.0;  // cumulative total
      int count = 0;         // number of values
      while (!StdIn.isEmpty())
      {
         double x = StdIn.readDouble();
         sum = sum + x;
         count++;
      }
      StdOut.println(sum / count);
   }
}
```

Both streams are *infinite* (no limit on their size).

Q. Do I always have to type in my input data and print my output?

A. No! Keep data and results in *files* on your computer, or use *piping* to connect programs.

Redirection: keep data in files on your computer

Redirect standard output to a file

```
% java RandomSeq 1000000 > data.txt

% more data.txt
0.09474882292442943
0.2832974030384712
0.1833964252856476
0.2952177517730442
0.8035985765979008
0.7469424300071382
0.5835267075283997
0.3455279612587455
...
```

"redirect standard output to"

Redirect from a file to standard input

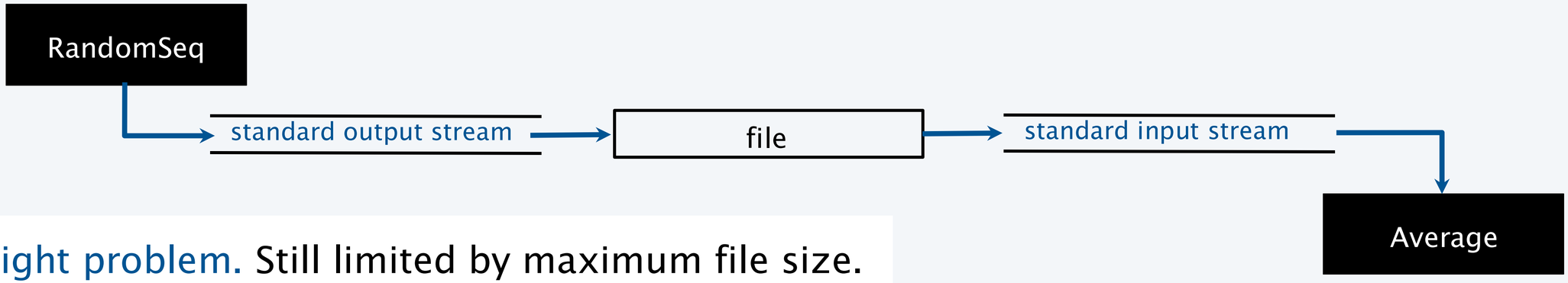
```
% java Average < data.txt
0.4947655567740991
```

"take standard input from"

If I want to get data from data.txt to Average:

For MAC it's: java Average < data.txt

For WINDOWS : cat data.txt | java Average



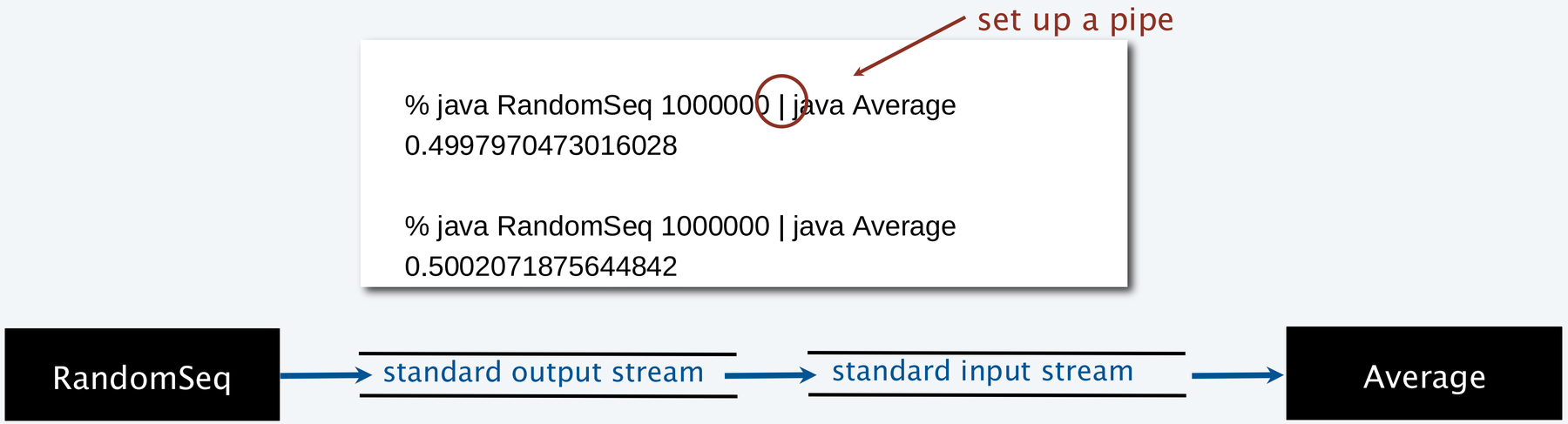
Slight problem. Still limited by maximum file size.

Piping: entirely avoid saving data

Q. There's no room for a huge file on my computer. Now what?

A. No problem! Use *piping*.

Piping. Connect standard output of one program to standard input of another.



Critical point. No limit *within programs* on the amount of data they can handle.

It is the job of the *system* to collect data on standard output and provide it to standard input.

INPUT-OUTPUT

If I want to get data from data.txt to Average:
For MAC it's: java Average < data.txt
For WINDOWS : cat data.txt | java Average

If I want to write RandomSeq results to data.txt
For MAC: java RandomSeq 10 > data.txt
For WINDOWS: java RandomSeq 10 > data.txt

If I want to pipe results from RandomSeq to Average
For MAC: java RandomSeq 10 | java Average
For WINDOWS: java RandomSeq 10 | java Average

Redirect from a file to standard input-More examples

```
public class TestInput{
    public static void main(String[] args) {
        int[] a = StdIn.readAllInts();
        for(int elt : a) {
              System.out.print (elt + " ");
         }
         System.out.println();
    }
}
```

ints.txt

```
12
21
23
32
43
34
54
45
65
56
87
78
98
89
90
13
25
336
48
59
```

```
$ javac TestInput.java
$ java TestInput < ints.txt
12 21 23 32 43 34 54 45 65 56 87 78 98 89 90 13 25 336 48 59
```

REMEMBER THIS???? 2D Problem—*ClassroomIO.java*

Consider a line of students waiting to be seated in a classroom.

- This is represented by an array of Strings (names)
- Consider the classroom is a rectangular array of seats. (5 x 4 array of Strings).
- Fill this classroom with the student waiting in line. Fill the seats in ROW-MAJOR order.

X X X X

X X X X

X X X X

X X X X

X X X X

Students will then leave the classroom in COL-MAJOR order.

Let's look at our Classroom again! *READ NAMES FROM A TEXT FILE!*

These are the ONLY changes to make to ClassroomIO.java!!!
Save as ClassroomIO.java
*Copy **names.txt** to your program folder*

```
/**
*
for (int count = 0; count < numStudents; count++ ) {
    studentLine[count] = args[count+ 2]; //get the students into the line (1D Array)
}
*/
```

```
int count = 0;
while (!StdIn.isEmpty() && count < numStudents){     //could have for loop as above
    String s = StdIn.readString();
    studentLine[count] = s;
    count++;
}
```

javac ClassroomIO.java
java ClassroomIO 5 4 < names.txt

REMEMBER THIS? 2D Problem

Consider a line of students waiting to be seated in a classroom.

- This is represented by an array of Strings (names)
- Consider the classroom is a rectangular array of seats. (5 x 4 array of Strings).
- Fill this classroom with the student waiting in line. Fill the seats in ROW-MAJOR order.

X X X X
X X X X
X X X X
X X X X
X X X X

WINDOWS: Get-Content names.txt | java Classroom 4 3

Students will then leave the classroom in COL-MAJOR order.

javac ClassroomIO.java
java ClassroomIO 5 4 < names.txt

```
$ javac Classroom2.java
$ java Classroom2 5 4 < names.txt
Students:   joe mary tomm alisha edwardo dana rohan shahir raul nicole hasham dylan zikun keisha lisa marly
owen elsa julian daniel

joe             mary            tomm            alisha
edwardo         dana            rohan           shahir
raul            nicole          hasham          dylan
zikun           keisha          lisa            marly
owen            elsa            julian          daniel
students leaving room by column: joe edwardo raul zikun owen mary dana nicole keisha elsa tomm rohan hasham
lisa julian alisha shahir dylan marly daniel
$
```

Image sources

http://paulbourke.net/fractals/gasket/cokegasket.gif

http://www.buzzfeed.com/atmccann/11-awesome-math-foods#39wokfk

http://sheilakh.deviantart.com/art/The-Legend-of-Sierpinski-308953447

http://commons.wikimedia.org/wiki/File:Lady_Fern_frond_-_normal_appearance.jpg

http://img3.wikia.nocookie.net/__cb20100707172110/jamescameronsavatar/images/e/e1/Avatar_concept_art-3.jpg

Image sources

http://www.digitalreins.com/wp-content/uploads/2013/05/Binary-code.jpg

http://en.wikipedia.org/wiki/Punched_tape#mediaviewer/File:Harwell-dekatron-witch-10.jpg



INTRODUCTION TO COMPUTER SCIENCE

Rutgers University

6. Input / Output

http://introcs.cs.rutgers.edu

Count Operations

```
READ n
WHILE n > 0
    DISPLAY n
    SUBTRACT 1 FROM n
ENDWHILE
```

```
int n = Integer.parseInt(arg[0]);
int sum = 0;
for (int i = 0; i < n; i++){
     sum  = sum + i;
}
System.out.println(sum)
```

```
READ cakeSize
IF cakeSize <= 0 THEN
    DISPLAY error
ELSE
    IF cakeSize < 6 THEN
        SET price TO 10.00
    ELSE
        IF cakeSize >= 6 AND cakeSize <= 12 THEN
            SET price TO 15.00
        ELSE
            SET price TO 20.00
        ENDIF
    ENDIF
    DISPLAY price
ENDIF
```