CSC220 (CSI)
Computational Problem Solving

Input and Output

The College of New Jersey

Please turn off your cell phone!

Input and Output

- Input devices.



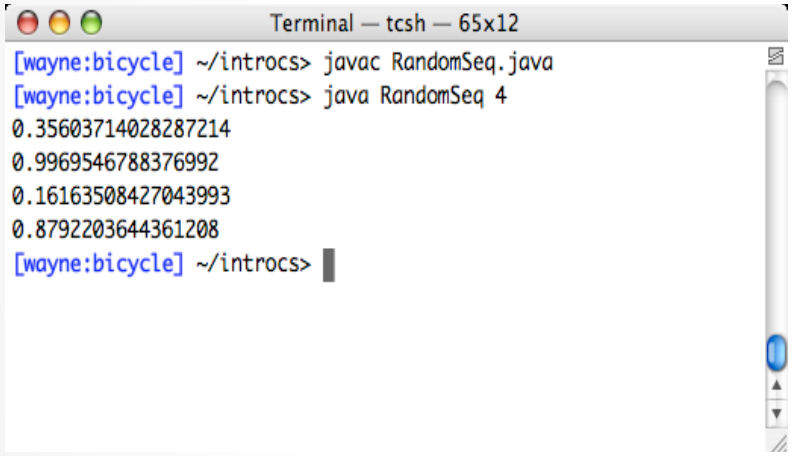

- Output devices.

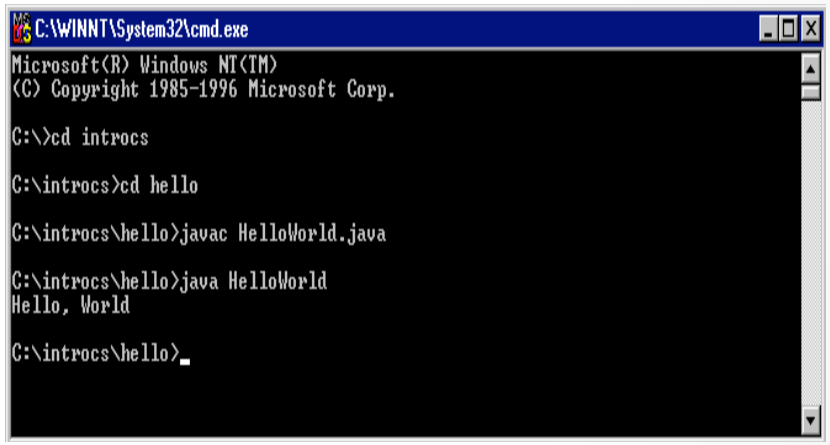




Terminal

- Terminal. Application where you can type commands to control the operating system.





Command-Line Input and Standard Output

- Command-line input. Read an integer N as command-line argument.
- Standard output.
  - Flexible OS abstraction for output.
  - In Java, output from System.out.println() goes to stdout.
  - By default, stdout is sent to **Terminal.**

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

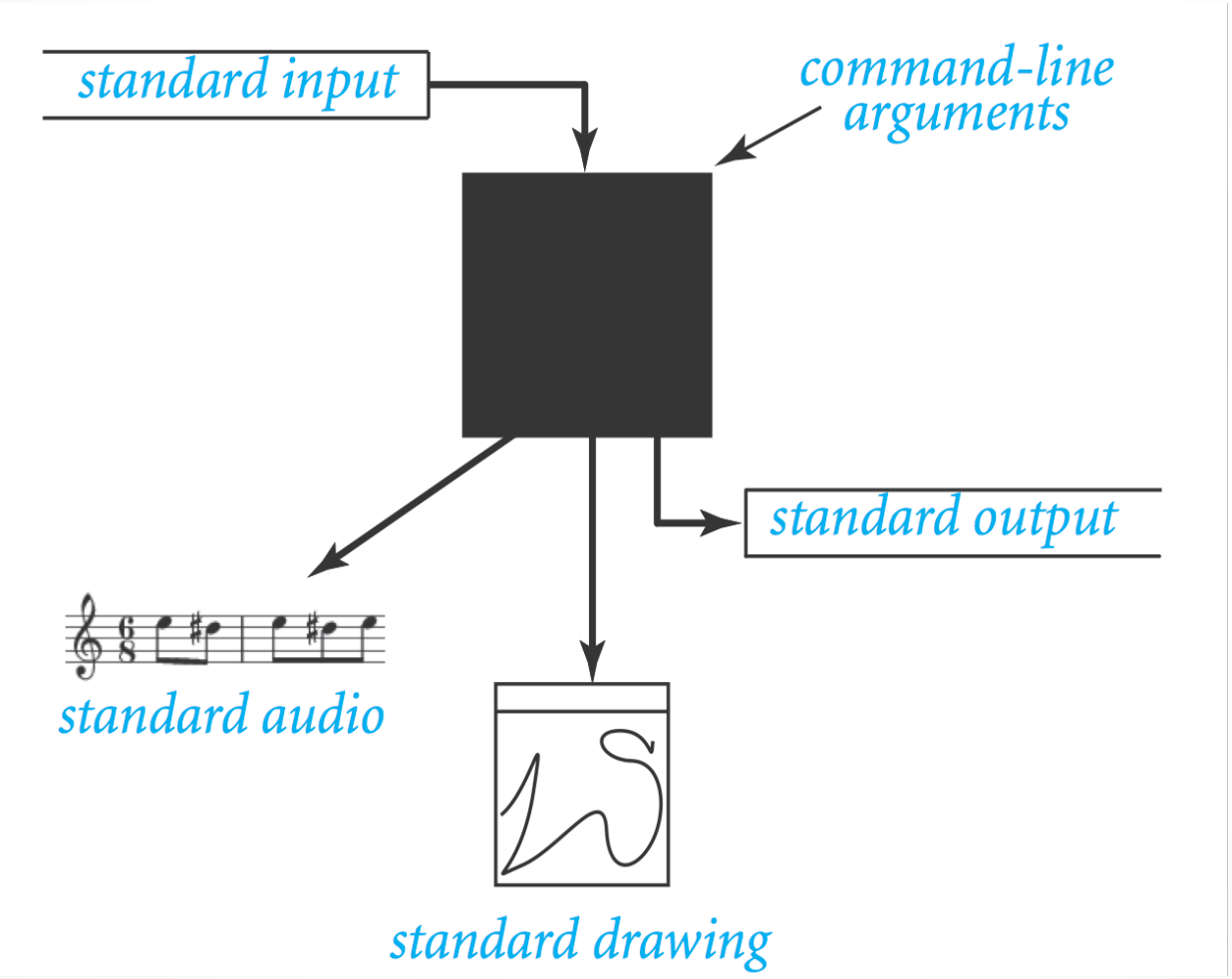
```
% java RandomSeq 4
0.9320744627218469
0.4279508713950715
0.08994615071160994
0.6579792663546435
```

Input from keyboard

```
import java.util.Scanner;

public class RandomSeqBoard {
    public static void main(String[] args) {
        System.out.print("please input one integer: ");
        Scanner scan = new Scanner (System.in);
        int N = scan.nextInt();
        for (int i = 0; i < N; i++) {
            System.out.println(Math.random());
        }
    }
}
```

Bird's Eye View



Redirection and Piping

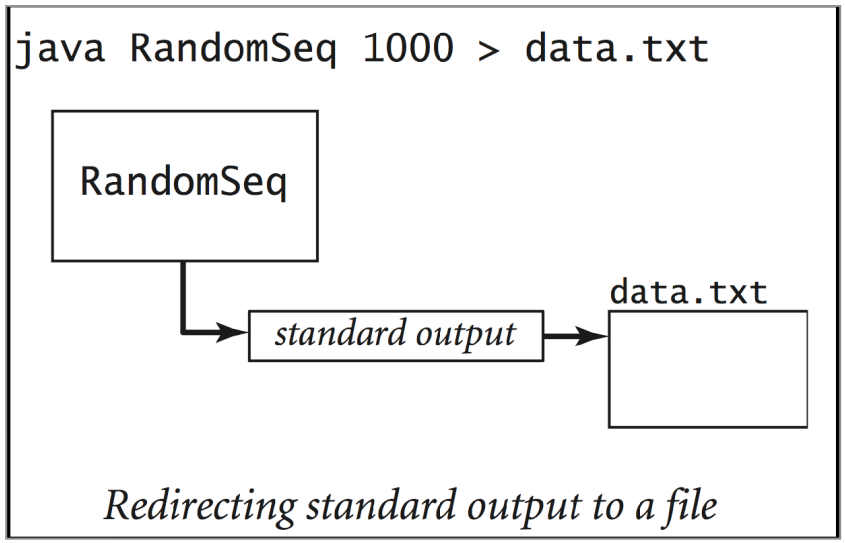
```
import java.util.Scanner;
public class Average {
  public static void main(String[] args) {
    double sum = 0.0; // cumulative total
    int n = 0;        // number of values
    Scanner scan = new Scanner (System.in);
    while (scan.hasNextInt()) {
      int x = scan.nextInt();
      sum = sum + x;
      n++;
    }
   System.out.println(sum / n);
  }
}
```

```
java Average
3
4
5
6
7
8
.
5.5
```

Note <**Ctrl-d**> signifies the end of file on Unix. On windows use <**Ctrl-z**>.

Redirecting Standard Output

- Redirecting standard output. Use OS directive to send standard output to a file for permanent storage (instead of terminal window).



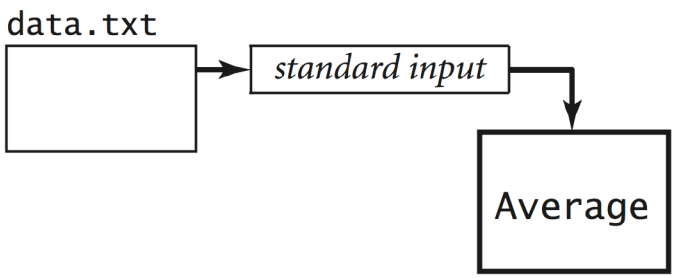
% java RandomSeq 1000 > data.txt

redirect stdout

Redirecting Standard Input

- Redirecting standard input. Use OS directive to read standard input from a file (instead of terminal window).

```
java Average < data.txt
```

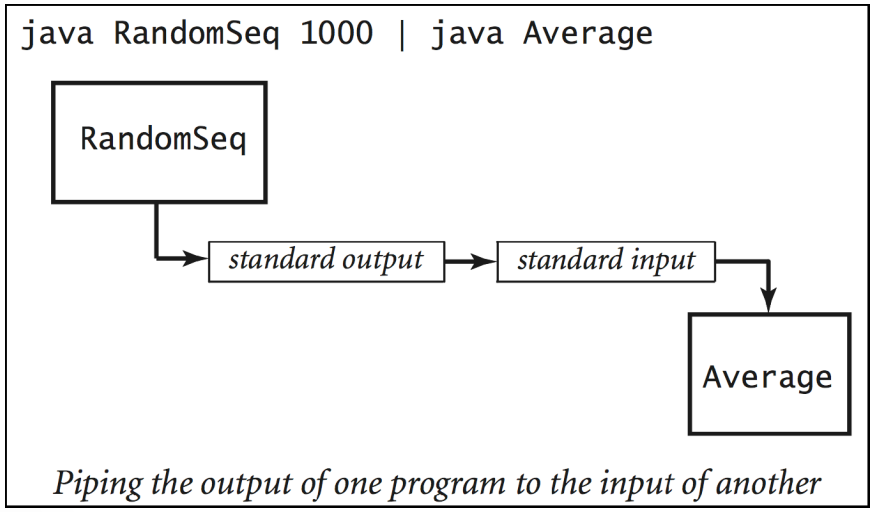


Redirecting from a file to standard input

```
% more < data.txt
11
23
4
5
…
% java Average < data.txt
7.9
```

Connecting Programs

- Piping. Use OS directive to make the standard output of one program become the standard input of another.



Piping the output of one program to the input of another

```
% java RandomSeq 1000000 | java Average
7.9
```

Standard Output

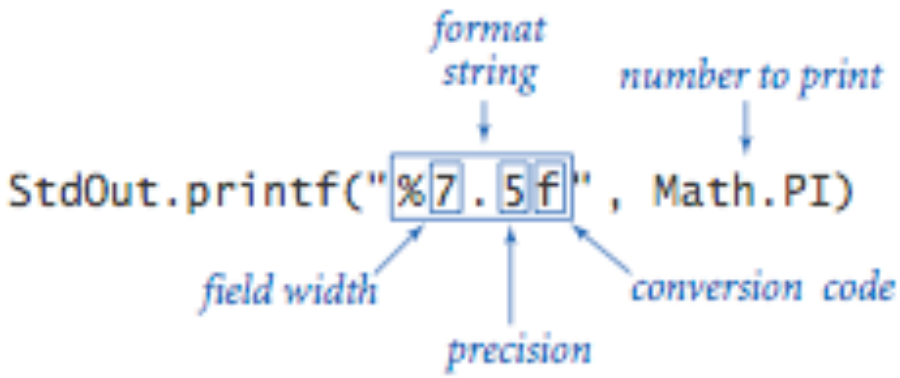
- Java's print() and println() methods, invoked with System.out, implement the standard output abstraction that we need, but to treat standard input and standard output in a uniform manner, we use the methods defined in the following API:

```
public class StdOut
  void print(String s)          print s
  void println(String s)        print s, followed by newline
  void println()                print a new line
  void printf(String f, ... )   formatted print
```

API for our library of static methods for standard output

Standard Output printf()

- printf() takes two arguments. The first argument, a string, contains a format that describes how the second argument is to be converted to a string for output.



Anatomy of a formatted print statement

Output Formats

type	*code*	*typical literal*	*sample format strings*	*converted string values for output*
int	d	512	"%14d" "%-14d"	" 512" "512 "
double	f e	1595.1680010754388	"%14.2f" "%.7f" "%14.4e"	" 1595.17" "1595.1680011" " 1.5952e+03"
String	s	"Hello, World"	"%14s" "%-14s" "%-14.5s"	" Hello, World" "Hello, World " "Hello "

Output Formatting

Format specifier	Data Type(s)	Notes
%c	char	Prints a single Unicode character
%d	int, long, short	Prints a decimal integer value.
%o	int, long, short	Prints an octal integer value.
%h	int, char, long, short	Prints a hexadecimal integer value.
%f	float, double	Prints a floating-point value.
%e	float, double	Prints a floating-point value in scientific notation.
%s	String	Prints the characters in a String variable or literal.
%%		Prints the '%' character.
%n		Prints the platform-specific new-line character.

Output formatting

- %(flags)(width)(.precision)specifier

- Flags
  - - Left justify
  - + Print a preceding + sign for positive values
  - 0 padding the values with zeros

Scanner reads a file

```
import java.io.File;
import java.util.Scanner;
import java.io.FileNotFoundException;
public class scannerRead{
   public static void main(String[] args) {
      try{
            File file = new File("input.txt");
            Scanner input = new Scanner(file);
             while(input.hasNext())
                 System.out.println(input.next());
              input.close();
       }
       catch (FileNotFoundException e){
            e.printStackTrace();
       }
   }
}
```

Writing to a file

```
import java.io.*;
public class writeFile
{
    public static void main (String [] args) throws IOException
    {
        File outFile = new File ("output.txt");
        FileWriter fWriter = new FileWriter (outFile);
        PrintWriter pWriter = new PrintWriter (fWriter);
        pWriter.println ("This is the first line.");
        pWriter.println ("This is the second line.");
        pWriter.close();
    }
}
```