CMSC 150
INTRODUCTION TO COMPUTING

LAB – WEEK 3

• STANDARD IO
• FORMATTING OUTPUT
• SCANNER
• REDIRECTING
INPUT AND OUTPUT

• Input devices
  - Keyboard
  - Mouse
  - Hard drive
  - Network
  - Digital camera
  - Microphone

• Output devices
  - Display
  - Speakers
  - Hard drive
  - Network
  - Printer
  - MP3 Player

• Goal. Java programs that interact with the outside world.
  • Java Libraries support these interactions
  • We use the Operating System (OS) to connect our program to them
WHAT HAVE WE SEEN SO FAR?

• **Command-line input.** Example: read an integer $N$ as command-line argument.

• **Standard output.**
  • The OS output stream for text
  • By default, standard output is sent to Terminal.
  • Example: `System.out.println()` goes to standard output.

```java
class RandomSeq {
    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        int i = 0;
        while (i < N) {
            System.out.println(Math.random());
            i++;
        }
    }
}
```
STANDARD INPUT AND OUTPUT
There is too much to cover in one slide, so here is a [link to help](#).

**Basics**
- Use `System.out.printf()` or `System.out.format()`.
- Their first argument is a string. Each time a `%` appears in the string, it is a directive to substitute it for a variable value. Attach each value after the string (comma separated):
  ```java
  System.out.printf("Hello %s", "World");
  ```
- Use `\n` in the string to add a new line.

**Examples**
- `System.out.printf("My int: %d", a);`
- `System.out.printf("My float: %f", d);`
FORMATTING OUTPUT

• The power of printf!

• Can control field width – how many characters are used to output item
  • Can right justify text

• Example %5d – always uses 5 characters to output an integer. Beginning would be white space, not zeroes

• Can also do the same on other types. Floats can determine number of decimal places: %5.7f means 5 characters before the decimal and 7 after

• The possibilities become infinite

1. public class PlayWithFormat {
2.   public static void main(String args[]) {
3.       System.out.printf("%5.7f\n",
4.           Double.parseDouble(args[0]));
5.   }
6. }
COMMAND-LINE INPUT VS. STANDARD INPUT

• Command-line input.
  • Use command-line input to read in a few user values.
  • Not practical for many user inputs.
  • Input entered before program begins execution.

• Standard input.
  • The OS stream for input
  • By default, standard input is received from Terminal window.
  • Input entered while program is executing.
STANDARD INPUT

• Use Scanner. Helpful links: API, Tutorial

• Basics:
  • Declare and initialize a scanner like:
    Scanner scanner = new Scanner(System.in);
  • Then use it like in the API:
    double d = scanner.nextDouble();
  • Usually you should prompt the user with
    System.out.print() to request input

```
1. import java.util.Scanner;
2. public class ReadSentence {
3.   public static void main(String[] args) {
4.     System.out.print("Enter sentence: ");
5.     Scanner s = new Scanner(System.in);
6.     String sentence;
7.     while(s.hasNext())
8.         sentence += s.next();
9.     System.out.println("Your sentence: " + s);
10. }
11. }
```

Note - hasNext() will return true until it sees Control+d.
DECODING YOUR TEXTBOOK

• Your textbook uses their own Java libraries StdIn.java and StdOut.java. These are synonymous (*BUT NOT EXACTLY THE SAME*) as Scanner and System.out respectively.

• *Please do not use StdIn or StdOut! These are not necessary libraries*
REDIRECTION AND PIPING
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
```
Redirecting standard input. Use OS directive to read standard input from a file (instead of terminal window).

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

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

```bash
% java RandomSeq 1000000 | java Average
0.4997970473016028
```

Another interesting item: Can pipe directly to command line arguments with xargs, e.g.,

```bash
java RandomSeq 10 | xargs java AverageCMDLine
```
EXERCISE – IN TRIPLETs

• Yes you have to be with someone!

• Starters: You work for JLDiablo Consultants Inc., which specializes in making software for Casino games (Cha-ching! $$$$). A new casino in Reno needs a slot game called Binary Slots 101010.
  • How it works:
    • A player enters a bet of their choice
    • Three Boolean values are randomly generated
    • If they are all true, then the player earns twice their money back!
EXERCISE – WHERE TO BEGIN

• When developing programs
  • Always think first!
  • Sketch out solution, i.e., plan
  • Implement solution
  • Test solution
  • Repeat!

• Called iterative development
EXERCISE – START THE PROGRAM

1. public class BinarySlots101010 {
2.     public static void main(String[] args) {
3.         System.out.println("Welcome to Binary Slots 101010!
4.     ");
5.     }
6. }
import java.util.Scanner;

public class BinarySlots101010 {
    public static void main(String[] args) {
        System.out.println("Welcome to Binary Slots 101010!

        Please enter your bet: ");
        Scanner scanner = new Scanner(System.in);
        double bet = scanner.nextDouble();
        System.out.printf("Your bet is \$.2f\n\n", bet);
    }
}
EXERCISE – GET BET ROBUSTLY

1. import java.util.Scanner;
2. public class BinarySlots101010 {
3.     public static void main(String[] args) {
4.         System.out.println("Welcome to Binary Slots 101010!\n\n\n");
5.         System.out.print("Please enter your bet: ");
6.         Scanner scanner = new Scanner(System.in);
7.         while(!scanner.hasNextDouble()) {
8.             System.out.println("Please enter a valid bet: ");
9.             scanner.next(); //Remember to eat up (read) bad input…
10.        }
11.     }
12.     double bet = scanner.nextDouble();
13.     System.out.printf("Your bet is $.2f\n\n", bet);
14. }
15. }
import java.util.Scanner;

public class BinarySlots101010 {
    public static void main(String[] args) {
        System.out.println("Welcome to Binary Slots 101010!

        System.out.print("Please enter your bet: ");
        Scanner scanner = new Scanner(System.in);
        while(!scanner.hasNextDouble()) {
            System.out.println("Please enter a valid bet: ");
            Scanner.next();
            //Remember to eat up (read) bad input…
        }
        double bet = scanner.nextDouble();
        System.out.printf("Your bet is $%.2f\n\n", bet);
        System.out.println("Spinning….match all to win!\n");
        boolean a = Math.random() < 0.5,
        b = Math.random() < 0.5,
        c = Math.random() < 0.5;
        System.out.println("Binary slots: " + 
                a + " " + b + " " + c + "\n");
        if(a && b && c)
            System.out.printf("You win 2x your bet! You won $%.2f\n", 
                    2*bet);
        else
            System.out.println("Sorry you lose…");
    }
}
EXERCISE

• Until the end of lab work with your team to improve the slots game. Possibilities:
  • Allow multiple bets without restarting the program
  • Track the user’s total money amount and allow them to cash out (leave the machine)
  • Modify game to have more Boolean values and allow different winning amount, i.e., if two are matched you get 1.2*bet, 4 matched you get 8*bet, or whatever works for you.
  • Use Unicode characters to allow more than two symbols
  • Make sure to protect all inputs (while and if statements)

• Always start with planning your program modification, then implement, and then test (ensure it works!)

• Save this for program for next week in your Box!