CS102
Introduction to data structures, algorithms, and object-oriented programming

DAY 5
Debugging Logical Errors

The point of testing is to find bugs -- semantic errors that show up as incorrect behavior rather than as compilation errors.

Most programming environments come with a debugger, which is a program that can help you find bugs by giving the value of different variables at a particular line in the code.

A more traditional approach to debugging is to insert debugging statements into your program. These are output statements that print out information about the state of the program.
break, continue, and labeled break

Java provides a general method for breaking out of the middle of any loop. It's called the break statement, which takes the form:

```
break;
```

If you use a `break` statement inside a nested loop, it will only break out of innermost loop that contains the break, not out of the loop that contains the nested loop. There is something called a labeled `break` statement that allows you to specify which loop you want to break out of by naming the loop.

A `continue` statement tells the computer to skip the rest of the current iteration of the loop. However, instead of jumping out of the loop altogether, it jumps back to the beginning of the loop and continues with the next iteration. There are labeled `continue` statements too.
One place a break is used is in a while loop that expects a certain type of input:

```java
int i = 0;
while (true) {
    System.out.println("Please enter a positive whole number");
    i = TextIO.getlnInt();
    if (i > 0) {
        break;  // correct input entered, end while loop
    }
    // incorrect input entered, ask user for input again
    // by returning to the top of the while loop.
    System.out.println("Input must be positive.");
}
```

This loop will continue until the user entered a whole number greater than 0.
continue

This statement can be used in a loop to skip subsequent lines in loop and go back to the start of the loop:

```java
int i = 0;
int sumEven = 0;
int count = 0;
while (true) {
    System.out.println(``Please enter 10 positive whole numbers’’);
    System.out.println(“and I will add the even numbers.”);
    i = TextIO.getlnInt();
    if (i < 0) {
        System.out.println(“Oops, that was a negative number.”);
        continue;  // skip lines below if and go back to while
    }
    sumEven = sumEven + i; // i must be positive
    count++;
    if (count == 10) {
        System.out.println(“The sum is “ +
```
```
Variations of the for loop § 3.4.1

Give three variations of for loops to print all the odd numbers between 1 and 21:

for (int i = 1; i <= 19; i += 2) {
    System.out.println(i);
} // end for

for (int j = 1; j < 21; j++) {
    if (j % 2 == 1) {
        System.out.println(j);
    } // end if
} // end for

for (int k = 0; k <= 10; k = 2k + 1) {
    System.out.println(2k + 1);
} // end for

/** Use a for loop to count 1, 3, ..., 21
* The numbers we want to print are 1, 1+2, 3+2, 5+2, ... ,19+2
*/

/** Use a for loop to count 1...21, but only print the numbers that are odd
*/

/** Use a for loop to count 1...10 and and print the numbers 2k + 1.
*/

for loops can also count down instead of up.
Nested For Loops § 3.4.3

Control structures can contain other control structures. In particular, for loops are often nested.

```java
for ( int rowNumber = 1; rowNumber <= 12; rowNumber++ ) {
    // for each row, process all columns
    for ( N = 1; N <= 12; N++ ) {
        System.out.printf("%4d", N * rowNumber);
        // print ints in 4-character columns; No newline
    }
    System.out.println(); // Add a newline
}
```
Nested loops

Often used in coding matrices but in many other instances too.

```java
String str;  // Line of text entered by the user.
int count;   // Number of different letters found in str.
char letter; // A letter of the alphabet.

System.out.println("Please type in a line of text.");
str = TextIO.getln(); // call to static getln methods in class TextIO
str = str.toUpperCase(); // call on non-static method in object str
count = 0;  // initialize count
System.out.println("Your input contains the following letters:");
System.out.println();

for ( letter = 'A'; letter <= 'Z'; letter++ )
{
    int i;   // Position of a character in str.
    for ( i = 0; i < str.length(); i++ ) {
        if ( letter == str.charAt(i) ) {
            System.out.print(letter);
            System.out.print(' ');
            count++;
            break;
        }
    }
}
```
A switch statement allows you to test the value of an expression and, depending on that value, to jump directly to some location within the switch statement.

The value of the expression can be one of the primitive integer types int, short, or byte. It can also be the primitive char type or it can be a String.

The expression **cannot** be a double or float value.
switch example

switch ( N ) {  // (Assume N is an integer variable or exp.)
    case 1:    // if N == 1
        System.out.println("The number is 1.");
        break;
    case 2:
    case 4:
    case 8:    // if N = 2, 4, or 8
        System.out.println("The number is 2, 4, or 8.");
        System.out.println("(That's a power of 2!)");
        break;
    case 3:
    case 6:
    case 9:    // if N = 3, 6, or 9
        System.out.println("The number is 3, 6, or 9.");
        System.out.println("(That's a multiple of 3!)");
        break;
    case 5:    // if N = 5
        System.out.println("The number is 5.");
        break;
    default:
        System.out.println("The number is 7 or is outside the");
        System.out.println(" range 1 to 9.");
}
Skip 3.7 for now (covers try-catch statements)
A data structure in which the items are arranged as a numbered sequence, so that each individual item can be referred to by its position number.

All the items in an array must be of the same type, and the numbering always starts at zero. An array is a list of variables, each accessible by the array name and position number of the variable.

An array is, technically, an object, so the process of creating one requires an instantiation with the keyword new.
arrays (cont.)

An array can be of any type and must first be **declared**:

```java
String[] name;     // declaration of String array
int[] age;            // declaration of int array
boolean[] leftHanded; // declaration of boolean array
```

Then the array must be **instantiated**:

```java
name = new String[1000];  // each with initial value null
age = new int[5];       // each with initial value 0
leftHanded = new boolean[100]; // each is false init
```

After instantiation, the specified number of boxes will be created in memory and reserved for that type.
arrays (cont.)

To put values into the array, you use the array name and position number to store a value at that position:

```java
name[5] = "Penny";
```

The length of an array is stored with the array as a field name accessible as, for example:

```java
name.length  // notice these are not method calls
age.length   // not method calls
```

Having access to the length of every array allows them to be easily used with a for loop to go through each element:

```java
// this for loop prints out all the elements in array age
for (i = 0; i < age.length; i++) {
    System.out.println( age[i] );
} // end for
```
2-dimensional arrays

Declaration and instantiation example:

    int[][][] matrix = new matrix[10][5];

This line would create a matrix with 10 rows and 5 columns, initially all 0. You would need to add values for each of the 50 ints in the array.

Often printed in nested for loops:

    int row, col; // loop-control-variables
    for ( row = 0; row < 5; row++ ) {
        for ( col = 0; col < 7; col++ ) {
            System.out.printf( "%7d", A[row][col] );
        } // end inner for
        System.out.println();
    } // end outer for
random numbers

The random method is a static member of the Math class. The call Math.random() produces a double between 0.0 and 1.0, inclusive. To use the Math.random() function to get a number between 1 and 10, you would use the following call:

\[
\text{int rNum} = (\text{int})(\text{Math.random()} \times 10) + 1
\]

The (int) operator truncates the real number to produce an int. This type of operator is called a “cast”. 