Welcome to Loops and Debugging Lab!

Learning Outcomes

By the end of this lab:
- Be able to define chapter 6 terms
- Describe various loops and how they differ from branches.
- Describe translating a while to a for loop.
- Describe when to use each loop, providing examples of each.
- Trace, explain and write loop code, including nested loops, utilizing break and continue statements.
- Identify and fix bugs in code using println statements.

Exercise A: Terms

looping vs branching, while vs for vs do-while, loop body, iteration, loop expression, infinite loop, loop variable, loop variable initialization, loop variable update, increment and decrement operators, nested loops, inner loop vs outer loop, incremental development, break statement, continue statement, scope, block, bug, debugging, debug output statements

Exercise B: Trace and Explain

A. int x = 34;
   int y = ++x;

B. int x = 34;
   int y = x++;

C. int k = 5;
   k = k++;

D. int x = 20;
   int j = 4;
   int y = x++ + --j;

E. int m = 14;
   int n = --m + m--;

F. int m = 4;
   int n = m -- -- m;

G. Scanner input = new Scanner(System.in);
   boolean done = false;
   while ( !done) {
      System.out.print("Enter an integer: ");
      if ( input.nextInt() < 0) {
         done = true;
   }
H. boolean done = false;
    while ( done = true) {
        System.out.println( "doing my work");
    }

I. Scanner input = new Scanner(System.in);
    boolean goodInput = false;
    do {
        System.out.print("Enter password: ");
        String line = input.nextLine();
        if ( line.contains( "password") ) {
            System.out.println("Really?");
        }
    } while ( !goodInput);

J. int i=10;
    while(i>1){
        System.out.println(i--);
    }

K. int i=10;
    while(i>1){
        System.out.println(i + 1);
    }

L. int i=10;
    while(i>1){
        System.out.println(++i);
    }

M. int i = 4;
    while ( i != 1 ) {
        System.out.println( i);
        i -= 2;
    }

N. for ( int i = 0; i < 10; ++i) {
        for ( int j = i; j < 10; j+= 2) {
            System.out.print("* ");
        }
        System.out.println();
    }

O. for ( int i = 0; i < 5; ++i) {
    if ( i == 0)
break;
}

P. for ( int i = 0; i < 5; ++i) {
  if ( i < 5) {
    continue;
  }
  System.out.println("in body of for loop");
}

Exercise C: Which Loop is Better

Are the following 3 loops equivalent?
Which of the following loops is preferable? Why?

```java
int count = 0;
while (count < 10) {
  System.out.println("count:" + count);
  count = count + 1;
}

for ( int count = 0; count < 10; ++count) {
  System.out.println("count:" + count);
}

int count = 0;
do {
  System.out.println("count:" + count);
  count = count + 1;
} while( count < 10);
```

Exercise D: Which Loop is Better

Are the following loops equivalent?
Which of the following loops would be better?

```java
Scanner input = new Scanner( System.in); //assume for each

int bigNum;
for (int i = 0; i < 2; i++) {
  System.out.print("Enter biggest int you can: ");
  bigNum = input.nextInt();
  if (bigNum < Integer.MAX_VALUE) {
    System.out.println("Go bigger!!!");
  }
}
```
int bigNum;
while (bigNum < Integer.MAX_VALUE) {
    System.out.print("Enter biggest int you can: ");
    bigNum = input.nextInt();
    if (bigNum < Integer.MAX_VALUE) {
        System.out.println("Go bigger!!!");
    }
}

Exercise E: Checking User Input

Describe the purpose of each line of code, paying particular attention to the Scanner method calls.

String name;
double weight = 0;
Scanner scnr = new Scanner(System.in);
System.out.print("Enter the name of a rock:");
name = scnr.nextLine();

boolean haveValidWeight = false;
do {
    System.out.print("Enter the weight of a rock:");
    if (scnr.hasNextDouble()) {
        weight = scnr.nextDouble();
        scnr.nextLine();
        if (weight > 0) {
            haveValidWeight = true;
        } else {
            System.out.println("Weight must be greater than 0.");
        }
    } else {
        String invalid = scnr.nextLine();
        System.out.println("Invalid input: " + invalid);
    }
} while (!haveValidWeight);

System.out.println("Name: " + name + "\nWeight: " + weight);
scnr.close();

Exercise F: Debugging with Print Statements

Debug the Factorial and Pascal Triangle programs with techniques described in zyBooks Chapter 6, such as print statements.
Factorial Expected Output
Calculate factorial of: 3
Factorial of 3 is 6

Calculate factorial of: 5
Factorial of 5 is 120

Calculate factorial of: 7
Factorial of 7 is 5040

Factorial.java

Pascal's Triangle Expected Output
Number of Rows in Pascal's Triangle: 4
1
1 1
1 2 1
1 3 3 1

Number of Rows in Pascal's Triangle: 5
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1

PascalTriangle.java  Wiki

Exercise G: More Maze
If you would like you may continue work on the Maze Lab from last week.

Exercise H: Debugging with Eclipse
Eclipse has a powerful debugger. Work through the Debugging Tutorial here:
Also available via the CS 200 Resources page.

Additional Learning Materials
When you have mastered everything in this lab, then you are welcome to learn from additional learning resources available on the web and beyond this course:
https://cs200-www.cs.wisc.edu/wp/learn-to-program-resources/
Note: Due to programs and zyBooks being individual work, it is Not appropriate to work on them during the Team Lab.

Lab designed by Jim Williams