## Nested Loops

## Nested loops

- Just as a selection structure can be nested within another selection structure (or within a loop), a loop can also be nested
- When one loop is nested within another, each iteration of the "outer" loop contains several iterations of the "inner" loop

## Example – multiplication table

- Suppose you wanted to print a multiplication table of the sort your instructor was forced to memorize in second grade
- Each line and column of the table has a number between 2 and 15 as its heading; the entries at each row/column intersection are the results when the row heading is multiplied by the column heading

## Multiplication table program output – an excerpt

```
          2    3    4    5    6    7    8    9
-------------------------------------------------
      2|    4    6    8   10   12   14   16   18
      3|    6    9   12   15   18   21   24   27
      4|    8   12   16   20   24   28   32   36
      5|   10   15   20   25   30   35   40   45
      6|   12   18   24   30   36   42   48   54
      7|   14   21   28   35   42   49   56   63
      8|   16   24   32   40   48   56   64   72
      9|   18   27   36   45   54   63   72   81
```

## Multiplication table - headings

Print the numbers between 2 and 15, spaced evenly
Print a series of hyphens in a single line
Place an end of line character after each of the lines above

```
public void printHeadings () {
        System.out.printf ("%8s", "");
        for (int x=2; x<=15; x++)
            System.out.printf ("%5d", x);
        System.out.print("\n");
        for (int y=0; y<80; y++)
            System.out.print("-");
        System.out.print("\n");
}
```

## Multiplication table

- Outer loop controls the number of lines to be printed; contains:
  - Inner loop
  - Line to print a newline character
- Inner loop controls the contents of each line
  - Row heading
  - Product of current row & column headings

## Code to print table

```
public void drawTable () {
        for (int x = start; x <= size; x++)
        {
            for (int y = start; y <= size; y++)
            {
                if (y==start)
                    System.out.printf("%7d%s", x, "|");
                System.out.printf("%5d", (x * y));
            }
            System.out.printf("\n");
        }
    }
```

## Tracing nested loops

- Write down value of each loop counter as it changes during loop execution
- If any output or change in other variable occurs, write this down next to the tally of loop counters

## Example – multiplication table

| x | y | output |
|---|---|---|
| 2 | 2 3 4 … 15 16 | 4 6 8 … 30 |
| 3 | 2 3 4 … 15 16 | 6 9 12 … 45 |
| 4 | 2 … 15 16 | 8 … 60 |
| . | | |
| . | | |
| . | | |
| 15 | 2 … 15 16 | 30 … 225 |
| 16 | | |

## Pattern of a Nested Loop

```
initialize outer loop
while  ( outer loop condition )
{       . . .
        initialize inner loop
        while ( inner loop condition )
        {
                inner loop processing and update
        }
         . . .
}
```

## Example Problem

Suppose we have data in the form below, involving several ID strings. For each ID string, a variable number of readings have been recorded; the number of readings for each ID is shown in the howMany column

| ID | howMany | Readings | | | | |
|---|---|---|---|---|---|---|
| 4567 | 5 | 180 | 140 | 150 | 170 | 120 |
| 2318 | 2 | 170 | 210 | | | |
| 5232 | 3 | 150 | 151 | 151 | | |

## Our goal: read in the data and display a summary chart like the one shown below:

```
ID                    Average

4567                  152
2318                  190
5232                  151
  . . .
There were 15 data sets on file
```

## Algorithm

- initialize count to 0
- read first ID and howMany
- while not at end of data
  - increment count
  - display ID
  - use a count-controlled loop to read and sum up this ID's howMany readings
  - calculate and display average for ID
  - read next ID and howMany
- display count

```
import java.util.*;

public class NestLoop {
  public static void main (String [] args) {
    int total = 0;      // total for all IDs
    int thisID,         // current ID number
      howMany,    // number of readings for current ID
      reading,    // current reading
      idTotal,    // total for current ID number
      idCount,    // counter for inner loop
      again;      // outer loop control variable
    double average; // average for current ID
    Scanner kb = new Scanner(System.in);
```

```
do { // start of outer loop
       System.out.print("Enter ID number");
       thisID = kb.nextInt();
       System.out.print
            ("How many readings for this ID?");
       howMany = kb.nextInt();
       idTotal = 0;
       idCount = 0;
       total++;
     // inner loop starts here
// inner loop – process all readings for this ID
   while (idCount < howMany) {
        System.out.print ("Enter reading");
        reading = kb.nextInt();
        idTotal += reading;
        idCount++;
   }
// outer loop continues here
```

```
     // continuation of outer loop
      average = (double)idTotal / howMany;
      System.out.print(thisID);
      System.out.printf("%17.2f\n", average);
      System.out.print
           (“Enter 0 to quit, 1 to continue: ”);
      again = kb.nextInt();
    } while (again == 1);
    System.out.println ("Total of " + total +
                   " records were processed.");
  } // end of main
} // end of class
```

## Using nested loops to draw figures (ASCII art)

- Drawing figures can illustrate how nested loops work
- Keep in mind the principle: outer loop controls number of lines, inner loop controls  content of lines

## Trace the following loop

```
int x, y;
for(x=0; x<5; x++)
{
        for(y=5; y>0; y--)
                System.out.print(“* ”);
        System.out.print(“\n”);
}
```

## Trace the following loop

```
import java.util.*;

public class triangle {
   public static void main (String [] args) {
      int x, y, z, height;
      Scanner kb = new Scanner(System.in);
      System.out.print ("Enter height: ");
      height = kb.nextInt();
      for (x=0; x<height; x++)
      {
         for (y=height; y>x; y--)
            System.out.print(" ");
         for (z=0; z<=x; z++)
            System.out.print("* ");
         System.out.print("\n");
      }
   }
}
```

```
height = 4
x  y              z
0  4 3 2 1 0      0 1
1  4 3 2 1        0 1 2
2  4 3 2          0 1 2 3
3  4 3            0 1 2 3 4
```

Output:

```
      *
     * *
    * * *
   * * * *
```

- y loop prints spaces
- z loop prints stars

## Loop example with break statement

```
int x,y;
for (x=1; x<5; x++)
{
        for (y=1; y<5; y++)
        {
                if (y > x)
                        break;
                System.out.print(“* ”);
        }
        System.out.print(“\n”);
        }
}
```

**OUTPUT:**

```
*
*  *
*  *  *
*  *  *  *
```

## Continue statement

- **is valid only within loops**
- **terminates the current loop iteration, but not the entire loop**
- **in a For or While, continue causes the rest of the body statement to be skipped--in a For statement, the update is done**
- **in a Do-While, the exit condition is tested, and if true, the next loop iteration is begun**

## Loop example with continue

```
int x,y;
for (x=1; x<5; x++)
{
        for (y=1; y<5; y++)
        {
                if (y > x)
                        break;
                System.out.print("* ");
        }
        if (x % 2 != 0)
                continue;
        System.out.print("\n");
}
```

**OUTPUT**

```
* * *
* * * * * * *
```

## Loop Testing and Debugging

- **test data should test all sections of program**
- **beware of infinite loops -- program doesn't stop**
- **check loop termination condition, and watch for "off-by-1" problem**
- **trace execution of loop by hand with code walk-through**
- **use debugging output statements**