COMP-202
Unit 4: Programming With Iterations

CONTENTS:

The `while` and `for` statements

Repetition Statements

- *Repetition statements* allow us to execute a (set of) statement(s) multiple times, repetitively; these statements are often called *loops*

Repetition Statements in Java

- Java has two main kinds of repetition statements: the *`while` loop* and the *`for` loop*
- "While" and "for" loops are equivalent
  - However, in certain circumstances, choosing one of these statements over the other results in a program that is easier to understand.

Example of a “while” loop

```
int x = 0;

while (x < 4) {
   System.out.println(“I must write“
   + “this over and over.”);
   x++;
}
```

Recall: increment/decrement operators

Remember that if you have an int variable, you can do

```
x++

++x

x--

--x
```

Example of a for loop

```
for(int x = 0; x < 4; x++) {
  System.out.println("I must"
  + "write this over and over");
}
```

Recall: "if" statement

if (condition)

One statement

Example:

```
if(x != 0)
    width = x + 5;
```

if (condition)

One **block** of statements

Example:

```
if(x == 0){
    z = x + 3;
}
```

“while” loops are similar in structure to “if” statements

```
while (condition)
    One statement
```

```
while (condition)
    One block of
statements
```

The difference is that when we reach the end of the one statement or block, we “go back” to the start of the while loop and re-test the condition

Example of a “while” loop

```
int x = 0;

while (x < 4) {
   System.out.println(“I must write“
   + “this over and over.”);
   x++;
}
```

```
x = 0;
if (x < 4) {
    System.out.println(...);
    x++;
    if (x < 4) {
        System.out.println(...);
        x++;
        if (x < 4) {
            System.out.....
```

Cheating at rock paper scissors

```
Scanner s = new Scanner(System.in);

bool keepPlaying = true;

while (keepPlaying) {

   System.out.println(“Enter Rock(r)”

    +“Scissor (s), Paper(p),or quit(q)”);

   char next = c.nextChar();

   if (next == 'r')
System.out.println(“I play paper!”);

if (next == 's')System.out.println(“I play rock!”);

if (next == 'p') System.out.println(“I play
scissor”);

if (next == 'q') {

      System.out.println(“Giving up! LAME!);
```

- `keepPlaying = false; } }`

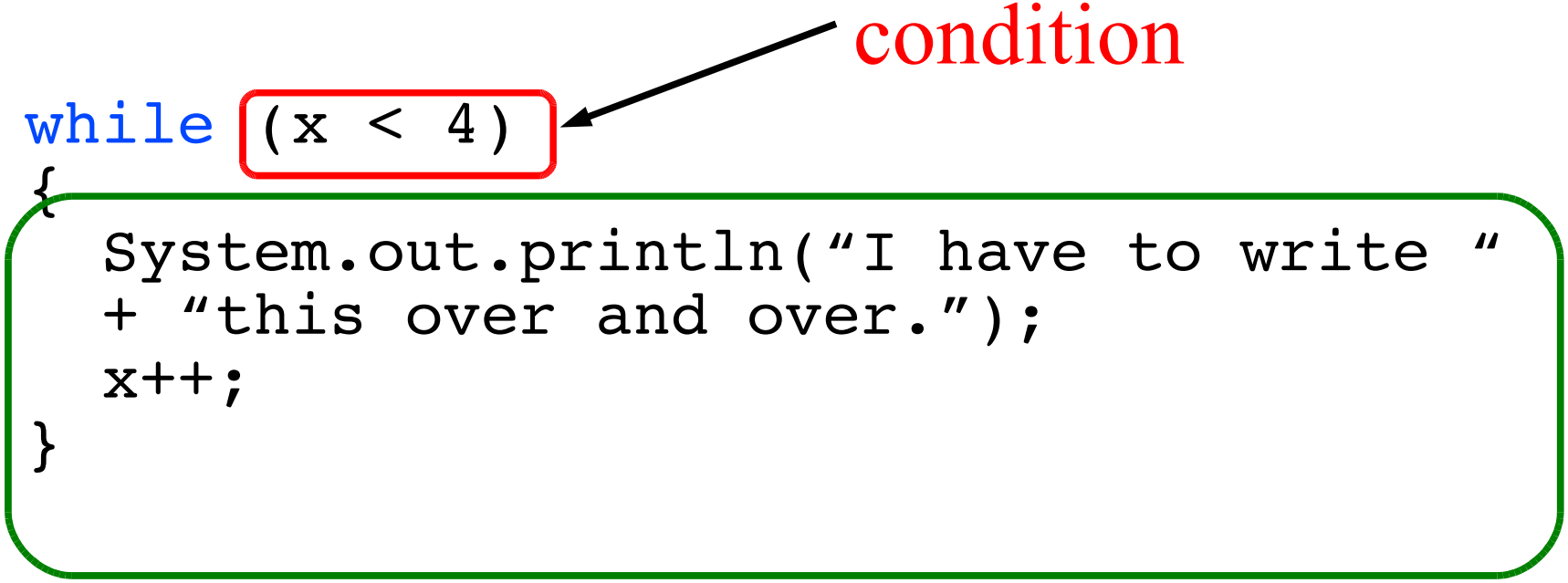
What does this display?

```
public class Counter {
  public static void main(String[] args) {
    final int LIMIT = 3;
    int count = 1;

    while (count <= LIMIT)
      count = count + 1;
      System.out.println(count);


    System.out.println("Done.");
  }
}
```

What does this display?

```
public class Counter {
  public static void main(String[] args) {
    final int LIMIT = 3;
    int count = 1;

    while (count <= LIMIT){
       System.out.println(count);
       count = count + 1;
     }

    System.out.println("Done.");
  }
}
```

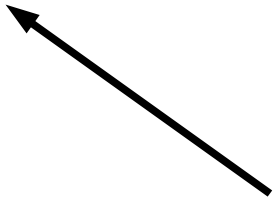
Components of a "while" loop



Components of a “while” loop

```
while (x < 4)
{
   System.out.println(“I have to write “
   + “this over and over.”);
   x++;
}
```

loop body



Definition: “iteration”

An **iteration** is a single execution of the instructions in the loop body.

This loop consists of 4 iterations.

```
int x = 0;

while (x < 4) {
   System.out.println("I have to write "
   + "this over and over.");
   x++;
}
```

I have to write this over and over.

I have to write this over and over.

I have to write this over and over.

I have to write this over and over.

How many iterations does the following loop consist of?

```
int x = 4;

while (x > 4) {
   System.out.println("I have to write "
   + "this over and over.");
   x++;
}
```

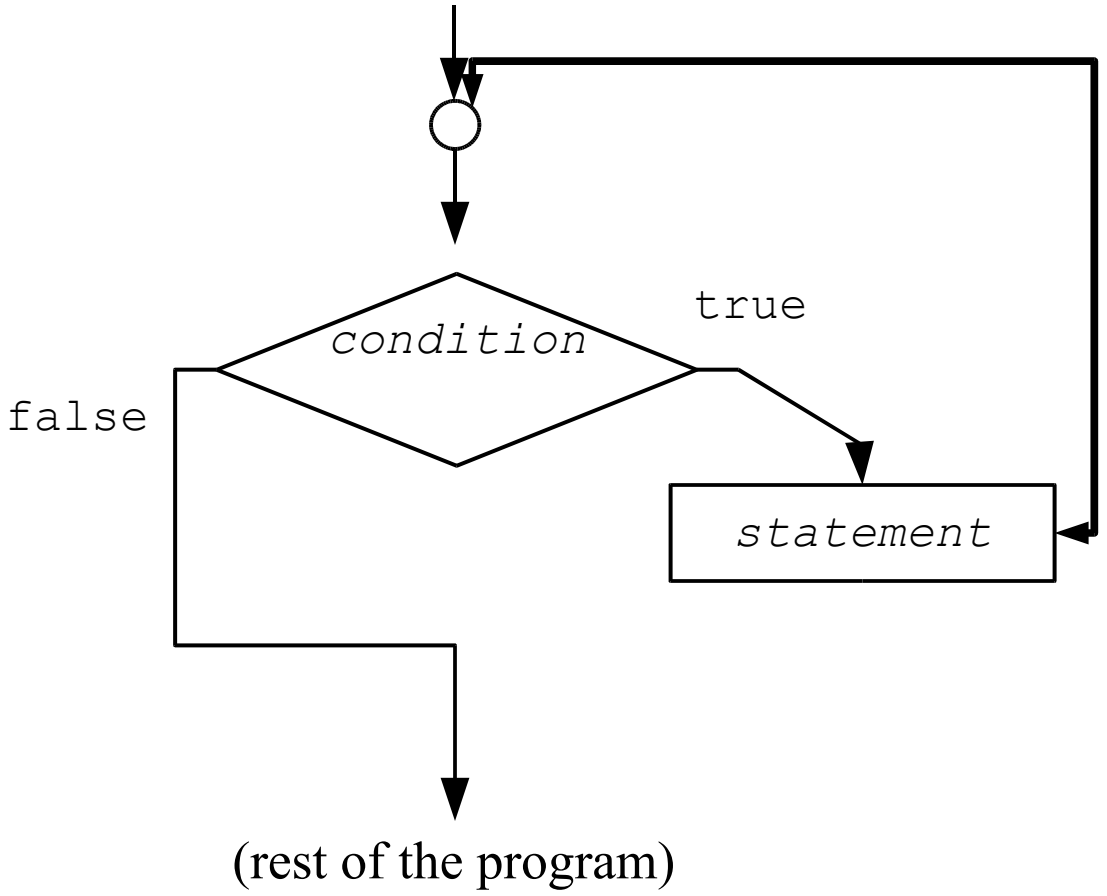
How many iterations does the following loop consist of?

```
int x = 6;

while (x > 4) {
   System.out.println("I have to write "
   + "this over and over.");
   x--;
}
```

How many iterations does the following loop consist of?

```
int x = 6;

while (x < 4) {
   System.out.println("I have to write "
   + "this over and over.");
   x++;
}
```

Logic of a `while` Statement



What is wrong here?

```
public class Abyss {
  public static void main(String[] args) {
    int count = 1;

    System.out.println("I'm going in...");

    while (count <= Integer.MAX_VALUE) {
      System.out.println(count);
      count = count - 1;
    }

    System.out.println("Found the bottom of the abyss!");
  }
}
```

Infinite Loops: A Common Logical Error

- An *infinite loop* executes until the user interrupts (terminates) the program.
- This is what happens when the statements in the body of a `while` loop never update the loop condition to `false`

What will be displayed? Why is it the wrong result? Correct the program so that it displays the right sum.

```
public class Sum {
  public static void main(String[] args) {
    final int MAX = 5;
    int sum = 0;
    int i = 1;

    while(i < MAX) {
      sum = sum + i;
      System.out.println("Sum: " + sum);
      i = i + 1;
      System.out.println("i: " + i);
      System.out.println();
    }

    System.out.println("The sum of integers from 1 to " +
      MAX + " is: " + sum);
  }
}
```

```
Sum
Sum: 1
i: 2

Sum: 3
i: 3

Sum: 6
i: 4

Sum: 10
i: 5

The sum of the integers from 1 to 5 is: 10
```

Off-By-One Errors

- It is a common logical error to write loop conditions that result in the loop body being executed one time too few, or one time too many
- You should always test your code to check that your loops conditions do not cause such errors

What will be displayed? Why is it the wrong result? Correct the program so that it displays the right sum.

```
public class AnotherSum {
  public static void main(String[] argv) {
    final int MAX = 5;
    int i = 0;
    int sum = 0;

    while(i <= MAX) {
      i = i + 1;
      System.out.println("i: " + i);
      sum = sum + i;
      System.out.println("sum: " + sum);
      System.out.println();
    }

    System.out.println("The sum of integers from 1 to " +
      MAX + " is: " + sum);
  }
}
```

```
i: 1
sum: 1

i: 2
sum: 3

i: 3
sum: 6

i: 4
sum: 10

i: 5
sum: 15

i: 6
sum: 21

The sum of the integers from 1 to 5 is: 21
```

General loop tips

- Try to think of exactly what it is that you want to do over and over again
- Figure out the range of values you want to try again and again. Is the loop counter also used in the computation?
- Make sure that your loop **starts** or initializes with the right result
- Make sure that your loop terminates with the correct result

Nested Loops

- Like `if` and `if-else` statements, loops can be nested as well
- Every time the body of the outer loop is executed, the inner loop will go through its entire set of iterations

What does this display?

```
public class NestedLoop {
  public static void main(String[] args) {
    final int MAX = 4;
    int outerCount, innerCount;

    outerCount = 1;
    while(outerCount <= MAX) {
      innerCount = 1;
      while (innerCount <= MAX) {
        System.out.print("(" + outerCount + "," + innerCount +
          ") ");
        innerCount++;
      }
      System.out.println();
      outerCount++;
    }
    System.out.println("Done.");
  }
}
```

```
(1,1) (1,2) (1,3) (1,4)
(2,1) (2,2) (2,3) (2,4)
(3,1) (3,2) (3,3) (3,4)
(4,1) (4,2) (4,3) (4,4)
Done.
```

What does this display?

```
public class AnotherNestedLoop {
  public static void main(String[] args) {
    final int MAX = 4;
    int outerCount, innerCount;

    outerCount = 1;
    while(outerCount <= MAX) {
      innerCount = 1;
      while (innerCount <= outerCount) {
        System.out.print("(" + outerCount + "," + innerCount +
          ") ");
        innerCount++;
      }
      System.out.println();
      outerCount++;
    }
    System.out.println("Done.");
  }
}
```

```
(1,1)
(2,1) (2,2)
(3,1) (3,2) (3,3)
(4,1) (4,2) (4,3) (4,4)
Done.
```

Part 2: The `for` Statement

Example: a “for” loop

```
for(int x = 0; x < 4; x++) {
   System.out.println(“I have to write “
   + “this over and over.”);
}
```

I have to write this over and over.

I have to write this over and over.

I have to write this over and over.

I have to write this over and over.

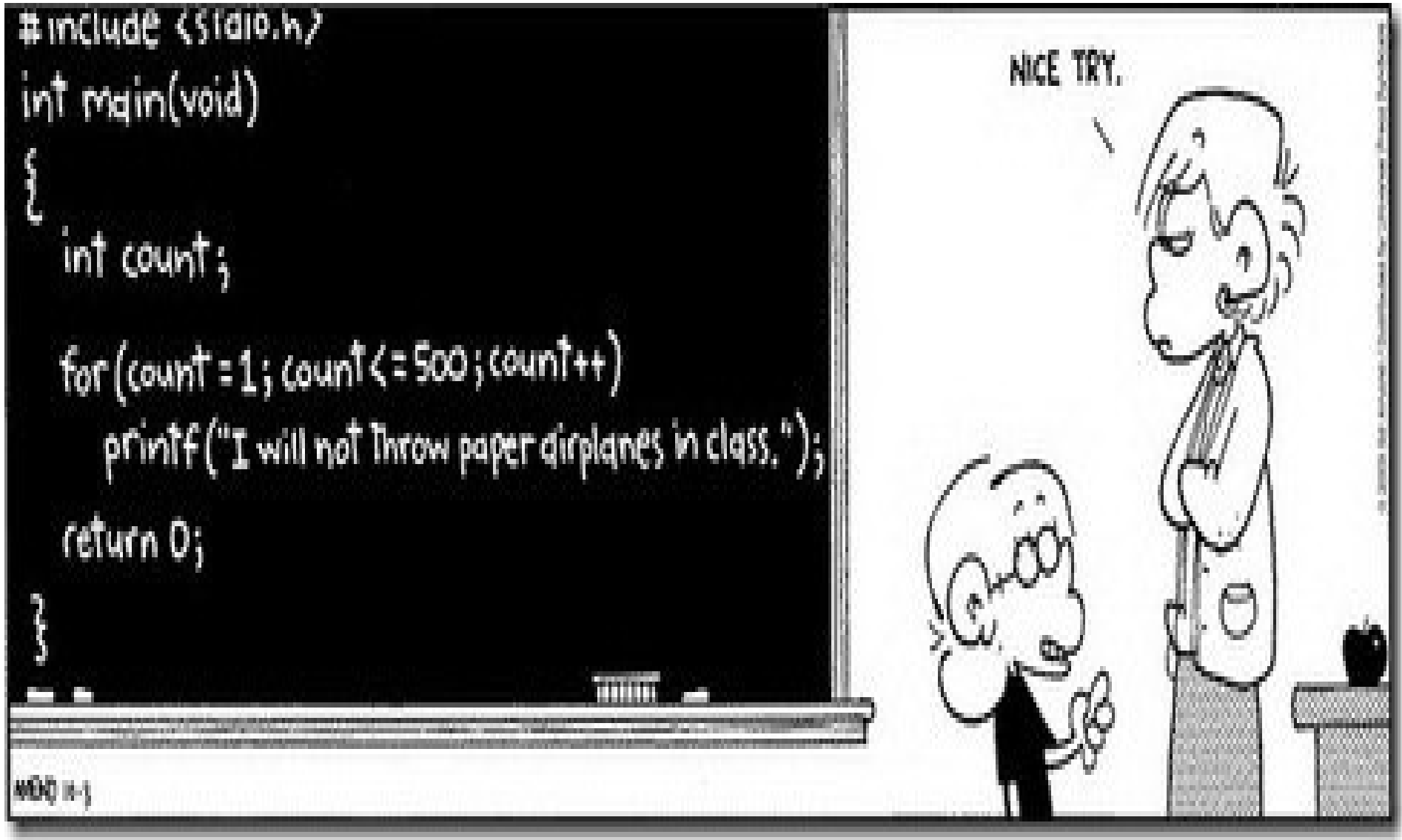
“for” loops are a little different, but still quite similar

- *for* (initialization; condition; follow-up action)
  - One statement

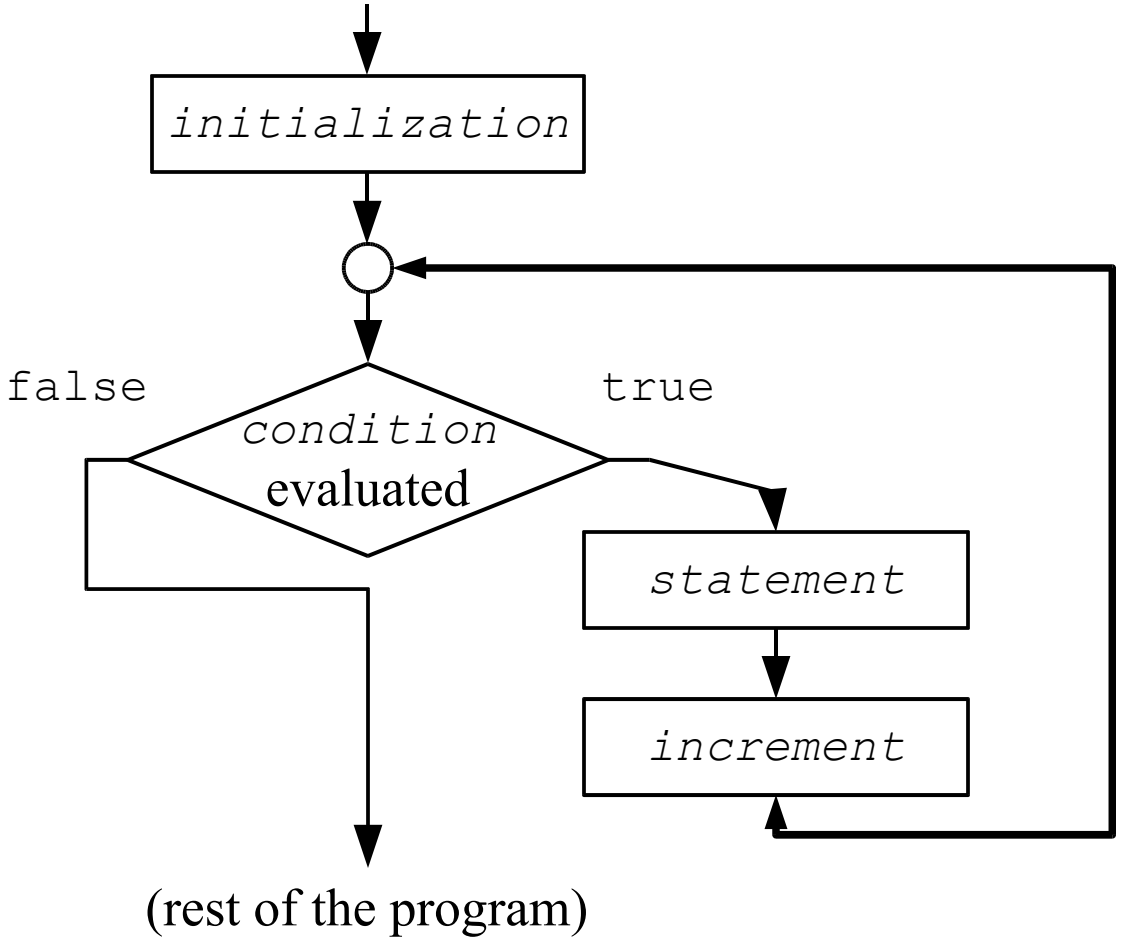
- *for* (initialization; condition; follow-up action)
  - One **block** of statements

Components of a “for” loop

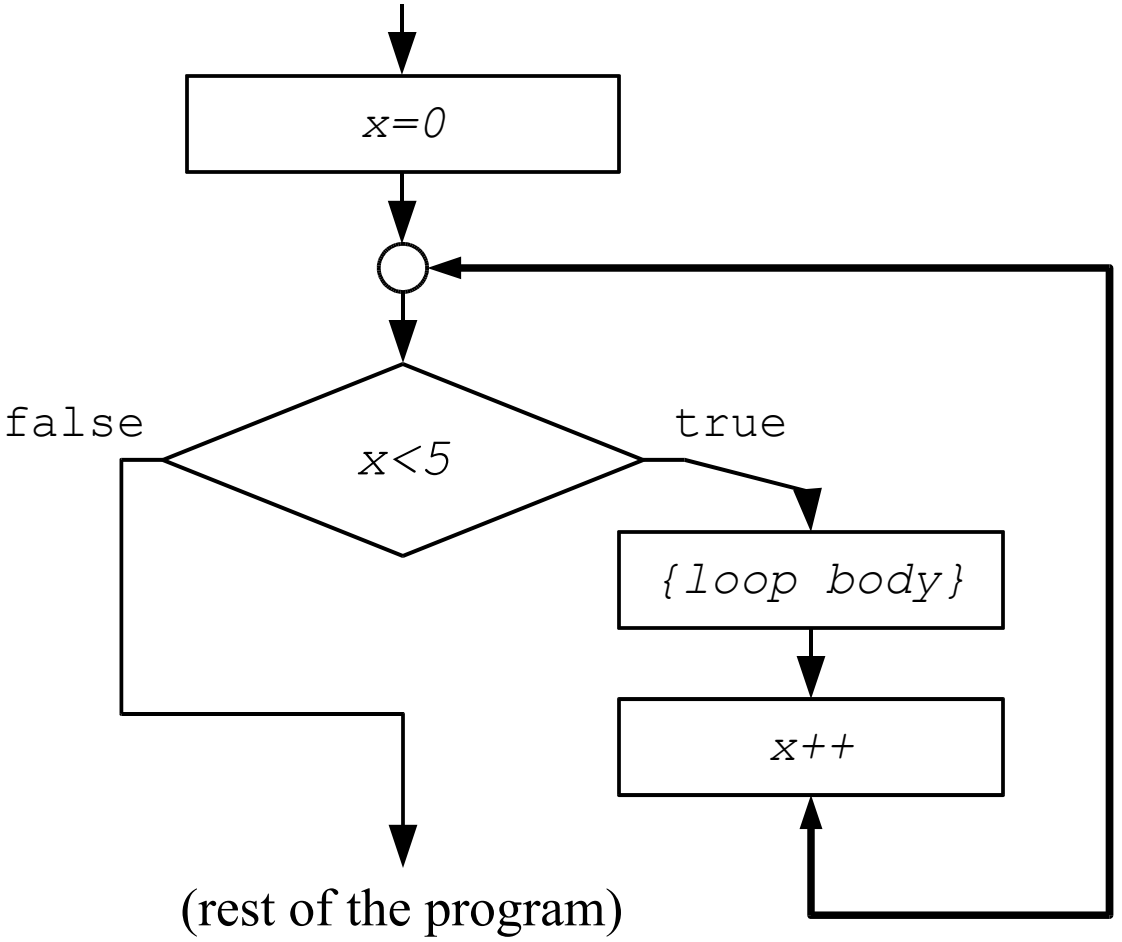
```
for(int x = 0; x < 4; x++)
{
  System.out.println(“I have to write “
  + “this over and over.”);
}
```


#include <stdio.h>
int main(void)
{
int count;
for (count = 1; count <= 500; count++)
printf ("I will not Throw paper airplanes in class.");
return 0;
}
NICE TRY.

Logic of a `for` Statement



Logic of a `for` Statement



`for` Loops as `while` Loops

- A `for` loop is equivalent to the following `while` loop structure:

```
initialization;
while ( condition ) {
   statement;
   increment;
}
```

`for` Loops as `while` Loops

- A `for` loop is equivalent to the following `while` loop structure:

```
initialization;
while ( condition ) {
    statement;
    increment;
}
```

- Excellent exercise to do at home!
- Find a for-loop example and:
- Rewrite it as a while loop
- Rewrite it as a series of if-else statements.

Execution of a `for` Loop

- Like in a `while` loop, the condition of a `for` loop is tested prior to entering the loop body
- If the condition of a `for` loop evaluates to `false` initially (that is, it evaluates to `false` the first time it is evaluated), the statement is never executed
- Therefore, the body of a `for` loop will be executed zero or more times
- A `for` loop is well suited for executing a specific number of times that can be determined in advance

AnotherCounter.java

```
public class AnotherCounter {
  public static void main(String[] args) {
    final int LIMIT = 3;

    for (int count=1; count <= LIMIT; count++) {
       System.out.println(count);
    }

    System.out.println("Done.");
  }
}
```

AnotherCounter.java

```
public class AnotherCounter {
  public static void main(String[] args) {
    final int LIMIT = 3;

    for (int count=1; count <= LIMIT; count++) {
       System.out.println(count);
    }

    System.out.println("Done.");
  }
}
```

Console:

```
1

2

3

Done.
```

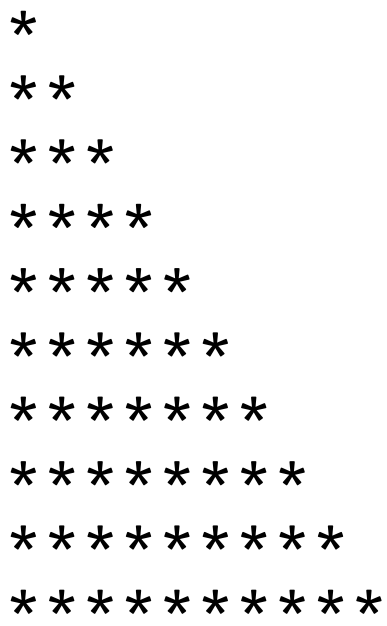
LIMIT: 3

count: 1 2 3 4

What does this display?

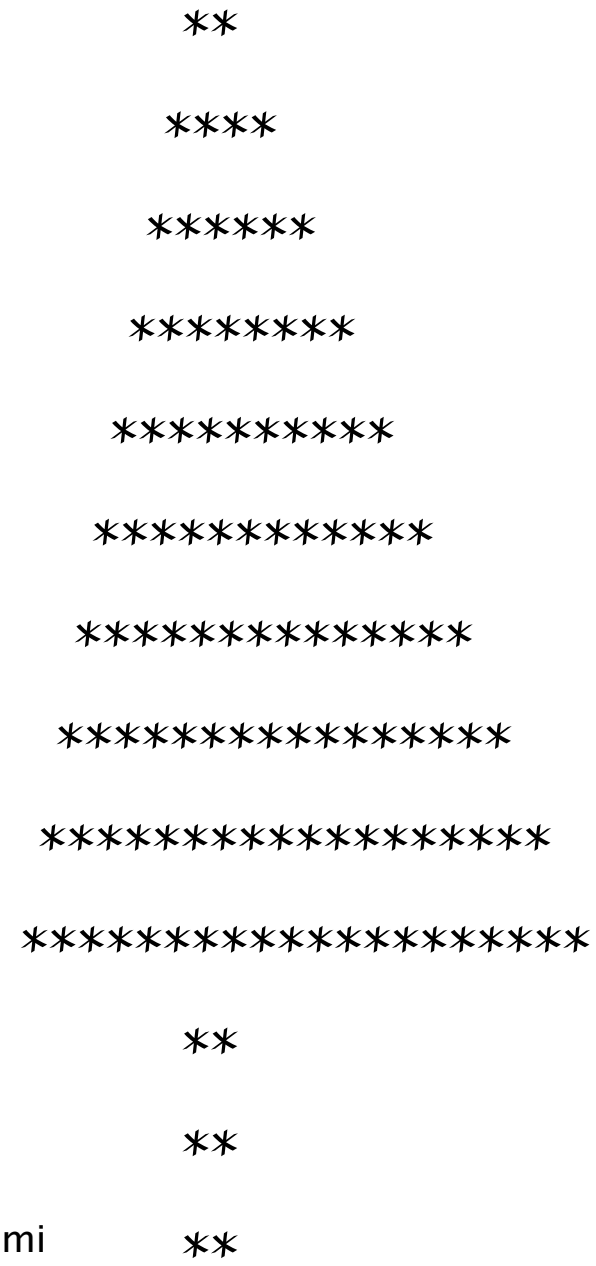
What shape does this display?

```
public class Stars {
  public static void main (String[] args) {
    final int MAX_ROWS = 10;

    for (int row = 1; row <= MAX_ROWS; row++)
  {
      for (int star = 1; star <= row; star++)
  {
        System.out.print("*");
      }
      System.out.println();
    }
  }
}
```



What Shape does this Display?

```
public class WhatIsThis {
  public static void main(String[] args) {
    final int MAX_ROWS = 10;
    for (int row = 1; row <= MAX_ROWS; row++) {
      for (int space = 1; space <= MAX_ROWS - row; space++) {
        System.out.print(" ");
      }
      for (int star = 1; star <= row * 2; star++) {
        System.out.print("*");
      }
      System.out.println();
    }
    for (int base = 3; base > 0; base--) {
      for (int space = 1; space <= MAX_ROWS-1; space++) {
        System.out.print(" ");
      }
      System.out.println("**");
    }
  }
}
```



Details on the `for` Statement

- Each expression in the header of a `for` loop is optional
  - If the *`initialization`* portion is left out, no initialization is performed
  - If the *`condition`* portion is left out, it is always considered to evaluate to `true`, and therefore creates an infinite loop
  - If the *`increment`* portion is left out, no increment operation is performed
- Both semi-colons are always required in the `for` loop header

`for` Loop: Exercise 1

- Complete the `main()` method of the `Multiples` class by adding code that displays all the multiples of a number entered by the user that exist between 1 and an upper limit also entered by the user
  - The completed program **MUST** display 5 multiples of the number entered by the user per line, except for the last line, which may contain less
  - In addition, the completed program **MUST** display

Multiples.java (1 / 2)

```
import java.util.Scanner;

public class Multiples {
  public static void main(String[] args) {
    Scanner keyboard = new Scanner(System.in);
    final int PER_LINE = 5;
    int value;
    int limit;

    System.out.print ("Enter a positive value: ");
    value = keyboard.nextInt();
    System.out.print ("Enter an upper limit: ");
    limit = keyboard.nextInt();

    System.out.println ("The multiples of " + value +
      " between " + value + " and " + limit +
      " (inclusive) are:");
    // Continued on next slide
```

Multiples.java (2 / 2)

```
      // Continued from previous slide

      // Add your code here
    }
  }
```

`for` Loop: Exercise 2

- Complete the `main()` method of the `Approximation` class by adding code that approximates the number *e* (the basis of the natural logarithm). The number *e* can be approximated by the following sum: *e* = (1 / 0!) + (1 / 1!) + (1 / 2!) + (1 / 3!) + ...
  - The `main()` method of the `Approximation` class asks the user to enter the number of terms of be computed; your task is to add code which computes each of these terms and adds them

Approximation.java

```
import java.util.Scanner;

public class Approximation {
  public static void main(String[] args) {
    Scanner keyboard = new Scanner(System.in);
    int n;

    System.out.print("Please enter the number of terms: ");
    n = keyboard.nextInt();

    // Add your code here
  }
}
```

Stars.java

```
public class Stars {
  public static void main (String[] args) {
    final int MAX_ROWS = 10;

    for (int row = 1; row <= MAX_ROWS; row++) {
      for (int star = 1; star <= row; star++) {
        System.out.print("*");
      }
      System.out.println();
    }
  }
}
```

What shape does this program display?

WhatIsThis.java (1 / 2)

```
public class WhatIsThis {
  public static void main(String[] args) {
    final int MAX_ROWS = 10;

    for (int row = 1; row <= MAX_ROWS; row++) {
      for (int space = 1; space <= MAX_ROWS - row; space++) {
        System.out.print(" ");
      }
      for (int star = 1; star <= row * 2; star++) {
        System.out.print("*");
      }
      System.out.println();
    }

    for (int base = 3; base > 0; base--) {
      for (int space = 1; space <= MAX_ROWS-1; space++) {
        System.out.print(" ");
      }
      // Continued on next slide
```

WhatIsThis.java (2 / 2)

```
        // Continued from previous slide
        System.out.println("**");
      }
    }
  }
```

What shape does this program display?

Part 4: Exercises

Exercises (1)

1. Write a program which consists of a single class called `Factor` that asks the user to enter a positive integer (including zero). The program then displays that number and its greatest prime factor. The program repetitively does this until the user inputs a negative number.

Exercises (2)

2.Write a program which consists of single class called `Boxes` that asks the user for a positive integer number `n`. The program then displays a solid square with side length `n` next to a hollow square with side length `n`. The program does nothing if the user inputs `0` or a negative value. If example, if the user enters 4, the following will be displayed:

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

Exercises (3)

3. One can approximate the square root of any number *a* using the following sequence:

$$x_0 = a / 2$$

$$x_{i+1} = (x_i + a / x_i) / 2$$

Write a program which consists of a single class called `SquareRoot`. This class defines a `main()` method that asks the user to enter a number *a*, and a number of iterations *n*, and reads these values from the keyboard. The program then computes the n^{th} term in the above sequence, thus approximating the value of of the square root of *a*, and displays it to the screen.