Lecture 4

CSE11 Spring 2015

While Loops, For Loops, Break, Continue, Switch

Some shorthand operators in java

increment/decrement

++, --

j++ ➔ j = j + 1;

j -- ➔ j = j – 1;

These can before or after the variable name. And it matters in assignment statements;

k = j++; ➔ k = j; j = j + 1;

k = ++j; ➔ j = j + 1; k = j;

Fused Assignment operators

```
a  += b;  ➔ a = a + b;
a *= b;   ➔ a = a * b;
a /= b;   ➔ a = a/b;
a %= b;   ➔ a = a % b;
```

Special note for += Operator

+= :: primitive types and Strings can use this

An Introduction to Objectdraw

- Objectdraw is a java package from Williams College
- It’s a library for teaching and learning about Java concepts.
  - We haven’t formally Introduced Objects yet

Hello2 (Graphical Version of Hello World)

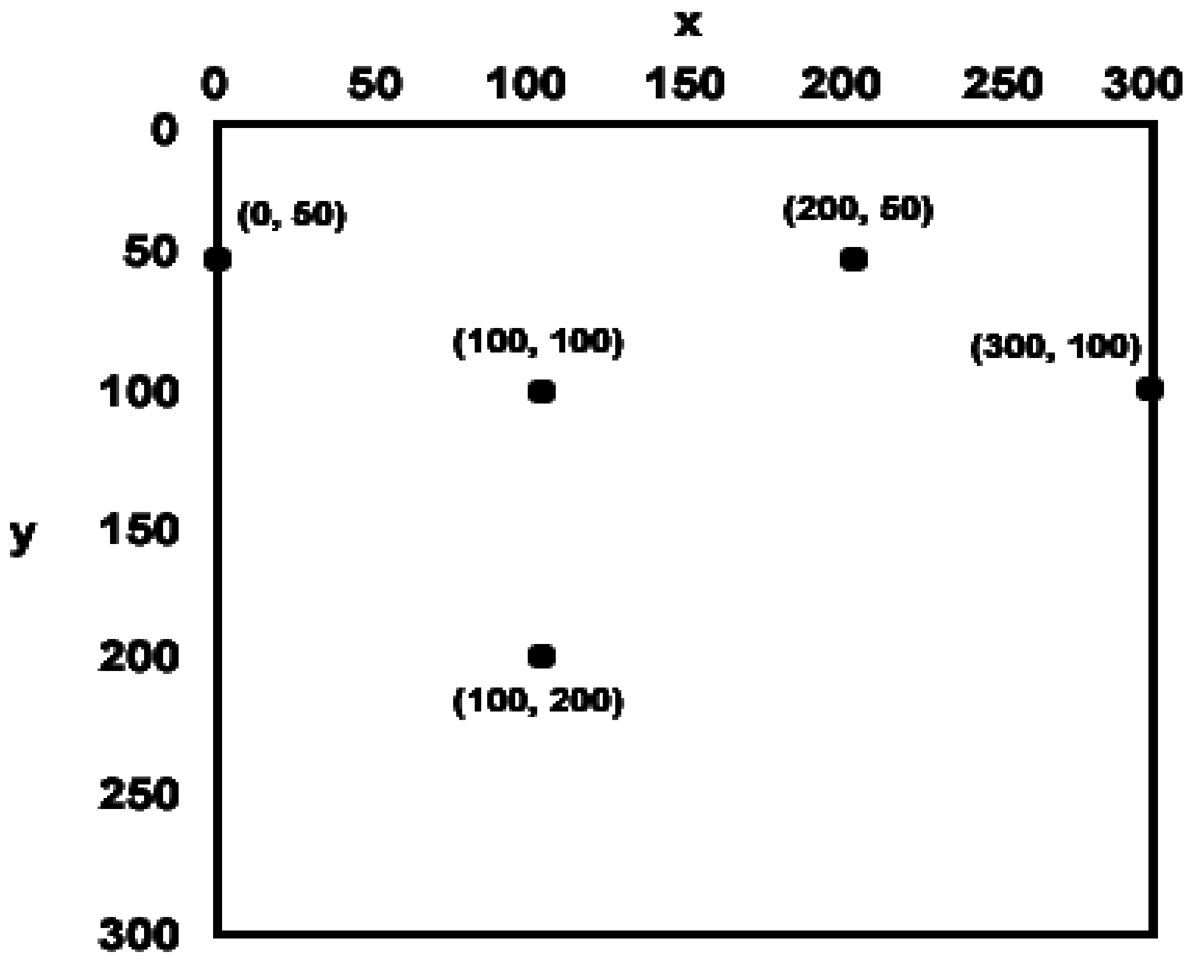
```
/****************************************************************
 *  Compilation:  javac -classpath '*' Hello2.java
 *  Execution:    java -classpath objectdraw.jar:'.' Hello2
 *
 ****************************************************************/
import objectdraw.*;
import java.awt.*;

public class Hello2 extends WindowController
{
        /* These are methods called when certain events occur */
        /* it is the objectdraw library that handles events   */
        public void onMousePress(Location point) {
                new Text("Hello World!", 40, 50, canvas );
        }

        public void onMouseRelease(Location point) {
                canvas.clear();
        }

        public static void main(String[] args) {
                new Hello2().startController(400,400);
}
```

Different Parts of Hello2

- `import` – allows use of classes written elsewhere
- `class` – a type definition. An object template.
- Instance variables, constructors, methods …
    - `extends` – builds upon an already-defined class
- Methods – sequence of statements to do something (code)
- `OnMousePress(), onMouseRelease(), …`
    - **Event handling** routines respond to specific input
- canvas – graphical “sketch paper”
    - ; - ends a java statement
- `new Text ( “Hello, World!”, 40, 50 canvas)`
- “new” creates a new <u>instance</u> of a class (Text, in this case)

Graphics Coordinates

Upper left corner is (0,0), +Y is <u>down</u>



What does “Location point” mean?

- Location is a “class”.
- One can also think of it is a data *type*

– *point is a <u>variable</u> (object) of <u>type</u> Location*

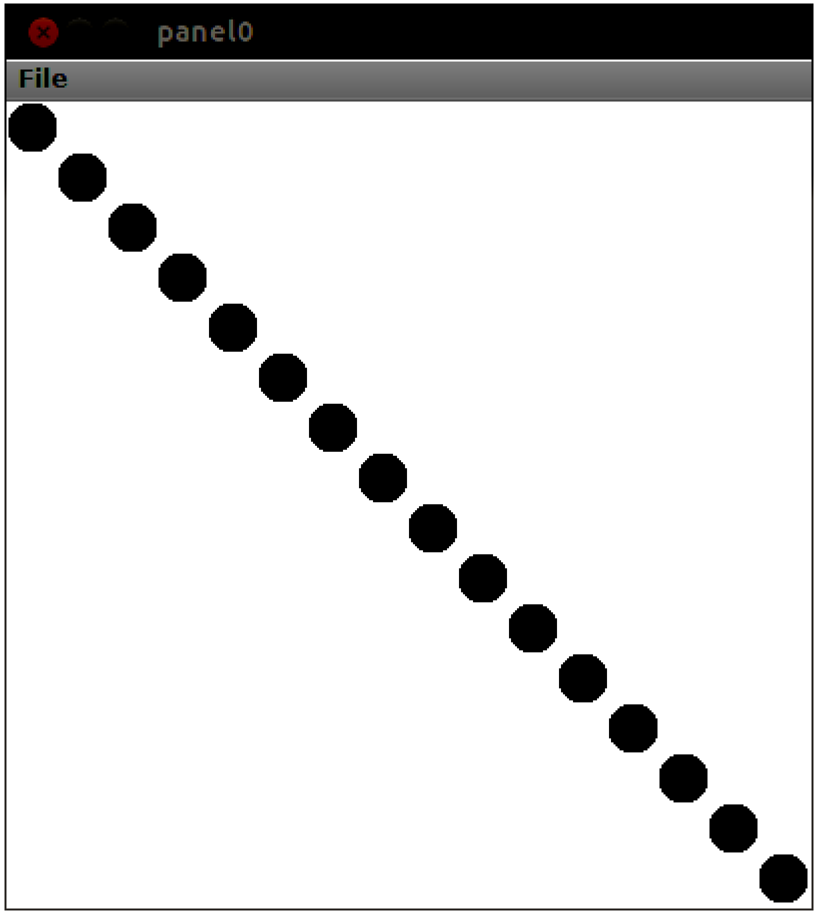
- It represents a coordinate location on the canvas.
- Abstractly, it is a compact way to represent an (x,y) coordinate
- Code segment:

```
private Location here;
        public void begin() {
                here = new Location(70,300);
        }
```

What are Computers Really Good at?

- Complex calculations
- Repetitive tasks
  - Identifying repetition is key to many programming tasks
  - There are many different ways to code repetition

Suppose you want to draw circles down the diagonal of a window



- Simply draw them, see Diagonal.java example

Diagonal.java

```
public class Diagonal extends WindowController
{
        private static final int WIN_SIZE = 400;
        private static final int DIAMETER = 25;
        private Text instructions;
        public void begin()
        {
                instructions = new Text("Click mouse to draw Circles", 50,WIN_SIZE-100, canvas);
        }
        public void onMouseClick(Location point) {
                instructions.hide();
                new FilledOval(0,0,DIAMETER,DIAMETER, canvas);
                new FilledOval(25,25,DIAMETER,DIAMETER, canvas);
                new FilledOval(50,50,DIAMETER,DIAMETER, canvas);
                new FilledOval(75,75,DIAMETER,DIAMETER, canvas);

                new FilledOval(100,100,DIAMETER,DIAMETER, canvas);
                new FilledOval(125,125,DIAMETER,DIAMETER, canvas);
                new FilledOval(150,150,DIAMETER,DIAMETER, canvas);
                new FilledOval(175,175,DIAMETER,DIAMETER, canvas);

                new FilledOval(200,200,DIAMETER,DIAMETER, canvas);
                 .
                 .
                 .
```

What's Wrong with this picture? (code)

- Nothing, It Works! Go Home. Have a Coffee.
- Except
  - What if you want to change the canvas size?
  - Seems very repetitive (many code statements that are <u>almost</u> identical)
  - Can't fit even a simple method definition on one screen
- What if you wanted to vary the color of each circle? How would you do that?

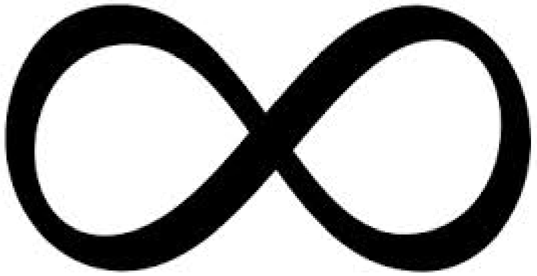
While loop

- Java has several ways to express a repetitive task. while() is just one

```
while (condition)
{
        java statement1;
        java statement2;
}
```

Keep repeating the statements in the while block (again and again and again) until condition is FALSE

The condition is how you test for termination (or when to STOP repeating the loop)



Infinite Loops

- These are almost always bad.
- Something <u>must</u> change inside the loop so that the while *condition* will eventually be FALSE
  - That change is an **<u>update</u>** inside the loop
  - While loops require you (the programmer) to properly code the update
- If that never happens, the computer executes the same statements over and over, forever
  - Often say, “The program is stuck in an infinite loop”

Does a while loop always execute?

- No.
- Why?
  - The condition may be false before any of the statements inside of the while loop are executed.

```
int j=10;
while (j < 10) {
    j *= 2;
}
System.out.println(”j = “ + j);
```

DiagonalLoop.java

```
public class DiagonalLoop extends WindowController
{
        private static final int WIN_SIZE = 400;
        private static final int DIAMETER = 25;
        private Text instructions;

        public void begin()
        {
                instructions = new Text("Click mouse to draw Circles", 50,WIN_SIZE-100, canvas);
        }
        public void onMouseClick(Location point)
        {
                instructions.hide();
                int corner = 0;                         // where to draw

                while ( corner < WIN_SIZE)      // Repeat until we run out of window
                {
                        new FilledOval(corner,corner,DIAMETER,DIAMETER, canvas);
                        corner += DIAMETER;
                }
        }
}
```

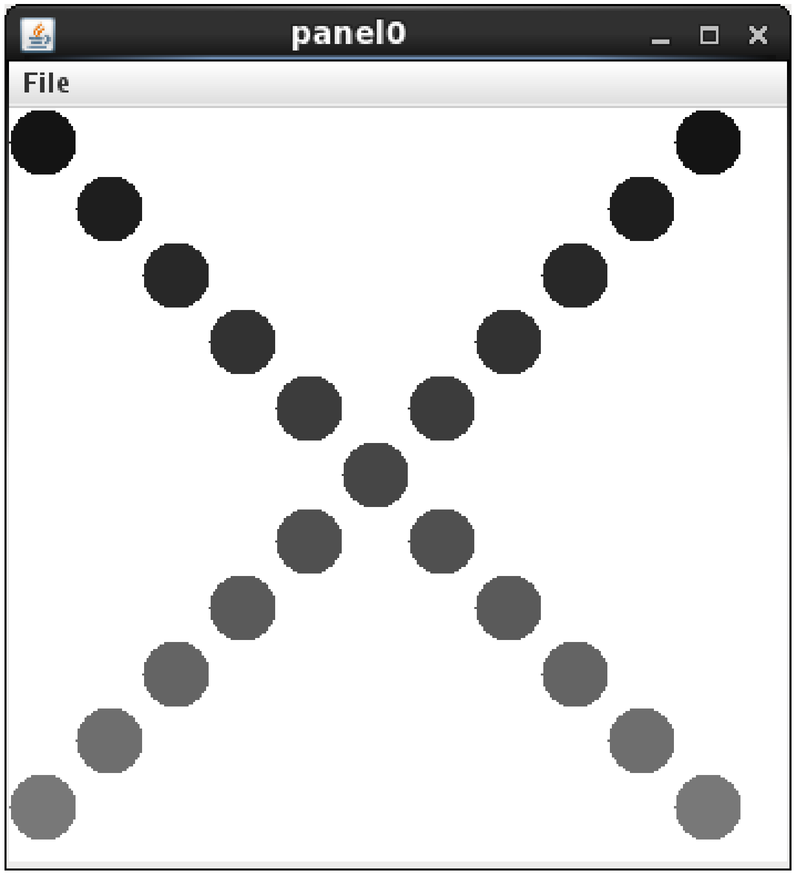
What's Different?

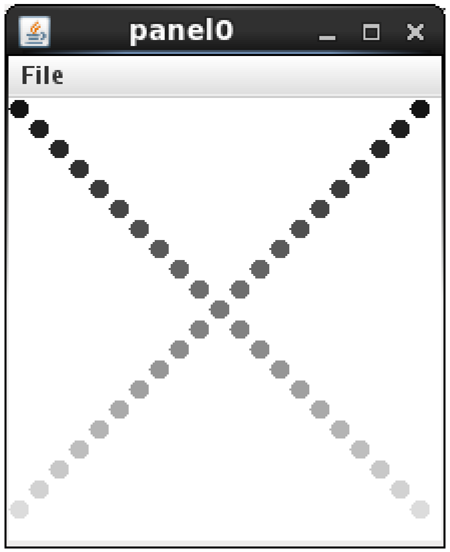
- Able to see the whole program on one screen
- Redefine WIN_SIZE and the number of circles drawn automatically adjusts to compensate
- What changes (updates) each time you execute the loop?
  - What is the termination condition?
- What would happen if you never updated corner?

Why does this work?

- We identified a repetitive task
  - Drawing a circle
- The only difference between different circles are their locations on the canvas
- One very good way to think about this
  - Draw the circle at (0,0)
  - Step down and to the right
  - Draw another circle.
  - Keep stepping down and to the right, drawing a new circle each time <u>until you run out of canvas</u>

How much extra code to create these pictures?





200x200 Circle Diameters of 10

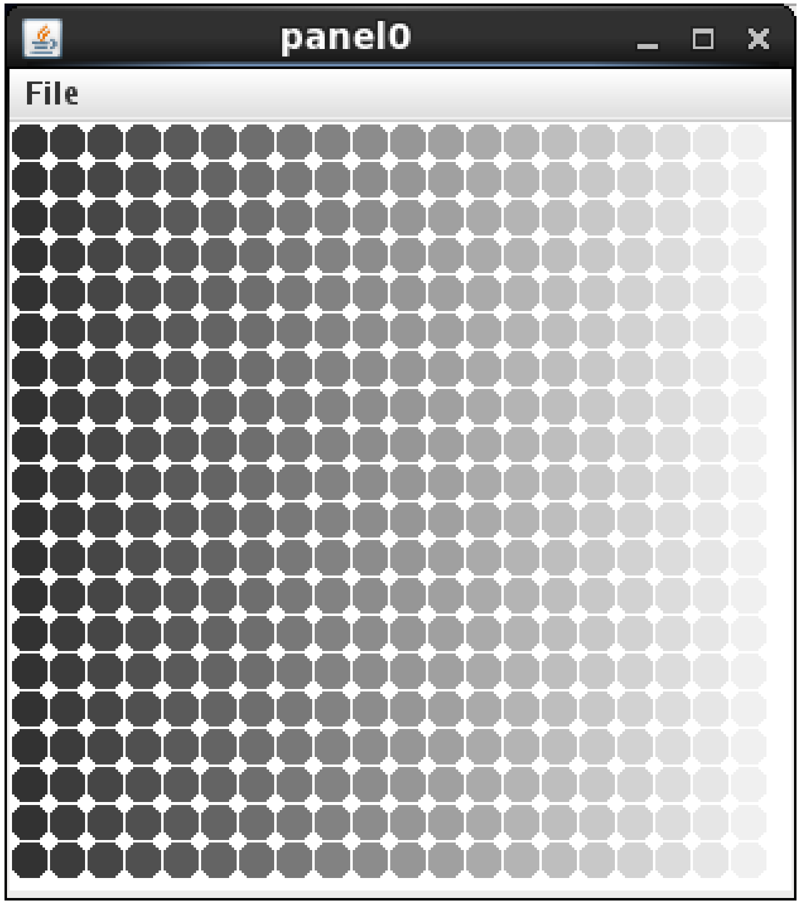
300x300, Circle Diameters of 25

DiagonalLoopGray.java

```
public void onMouseClick(Location point)
{
    instructions.hide();
    int xcoord = 0;         // where to draw on \-diagonal
    int ycoord = 0;         // ycoord same for both diagonals
    int x2coord = WIN_SIZE-DIAMETER; // where to draw on /-diagonal

    int hue = GRAY; int COLORCHANGE=10;
    FilledOval circle;

    // Draw circles down the diagonals. Start at upper right, upper left of canvas
    // and move downwards
    while ( xcoord < WIN_SIZE)  // Repeat until we run out of window
    {
        circle = new FilledOval(xcoord,ycoord,DIAMETER,DIAMETER, canvas);
        circle.setColor(new Color(hue,hue,hue));

        circle = new FilledOval(x2coord,ycoord,DIAMETER,DIAMETER, canvas);
        circle.setColor(new Color(hue,hue,hue));

        xcoord += DIAMETER; ycoord=xcoord;    // Move X-coord to right, Y-coord down
        x2coord -= DIAMETER;
        hue += COLORCHANGE;
    }
}
```

March Across – March Down

- Nested Loops
- Every row is identical, Gray color changes each time you go across the canvas



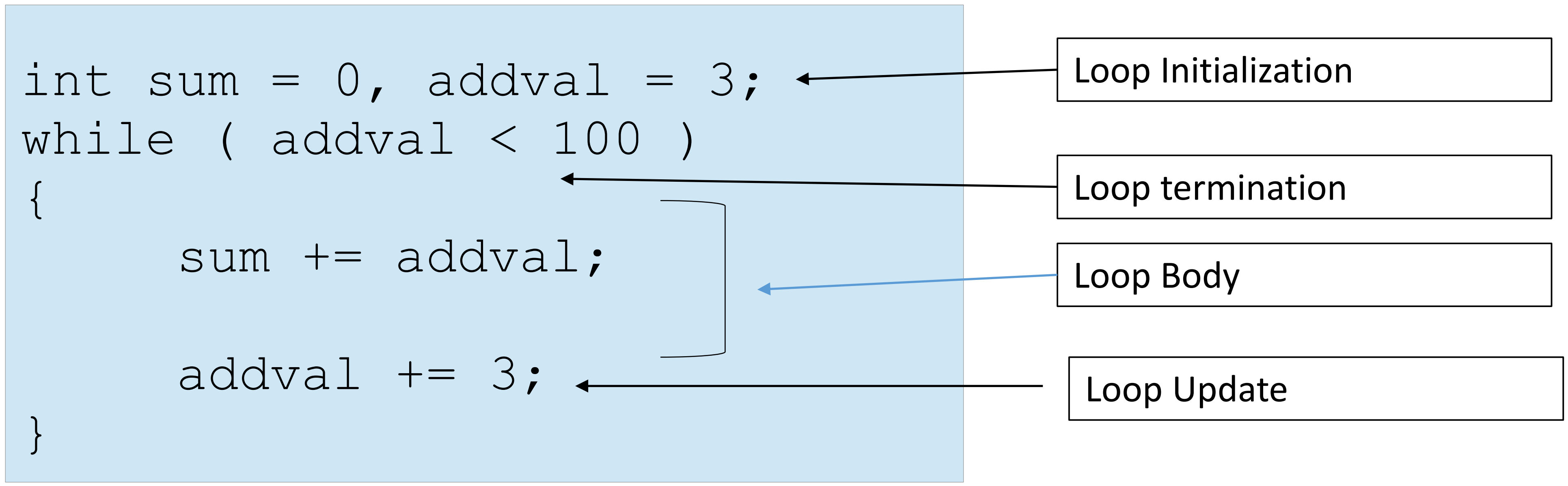
NestedLoopGray.java

```
public void onMouseClick(Location point)
{
        instructions.hide();
        int row = 0;            // which row are we on
        while ( row < WIN_SIZE )
        {
                int hue = GRAY; int COLORCHANGE=10;
                int column = 0; // Start at the left
                while (column < WIN_SIZE)
                {
                        FilledOval circle;
                        circle = new FilledOval(column, row,
                                        DIAMETER,DIAMETER, canvas);
                        circle.setColor(new Color(hue,hue,hue));
                        column += DIAMETER;
                        hue += COLORCHANGE ;
                }
                row += DIAMETER; // Go to the next row
        }
}
```

the `for` loop

- Counting up or counting down in loops is very common that Java (C, Python,C++, Perl, FORTRAN, …) provides a specific construct
- Simple Example
  - Add up all number between 1 and 100 that are divisible by 3

Doing this with a while (and for) loop



```
int sum = 0, addval = 3;
while ( addval < 100 )
{
     sum += addval;

     addval += 3;
}
```

A `for` loop statement rearranges the statements

`for` (<loop initialization>; <loop termination>; <loop update>)
     <loop body>

```
int sum = 0, addval;
for ( addval = 3; addval < 100; addval += 3)
     sum += addval;
```

Some Notes on for loops

- Usually, the loop body is actually a statement block
- <loop initialization>,<loop termination>, <loop update> can all be empty
    - `for (;;)` is a legal infinite loop
- Be careful with indented code that is *not a statement block*

```
int sum = 0, addval, mulval;
for (addval = 3, mulval=1; addval < 100; addval+=3)
     sum += addval;
     mulval *= 3;
```

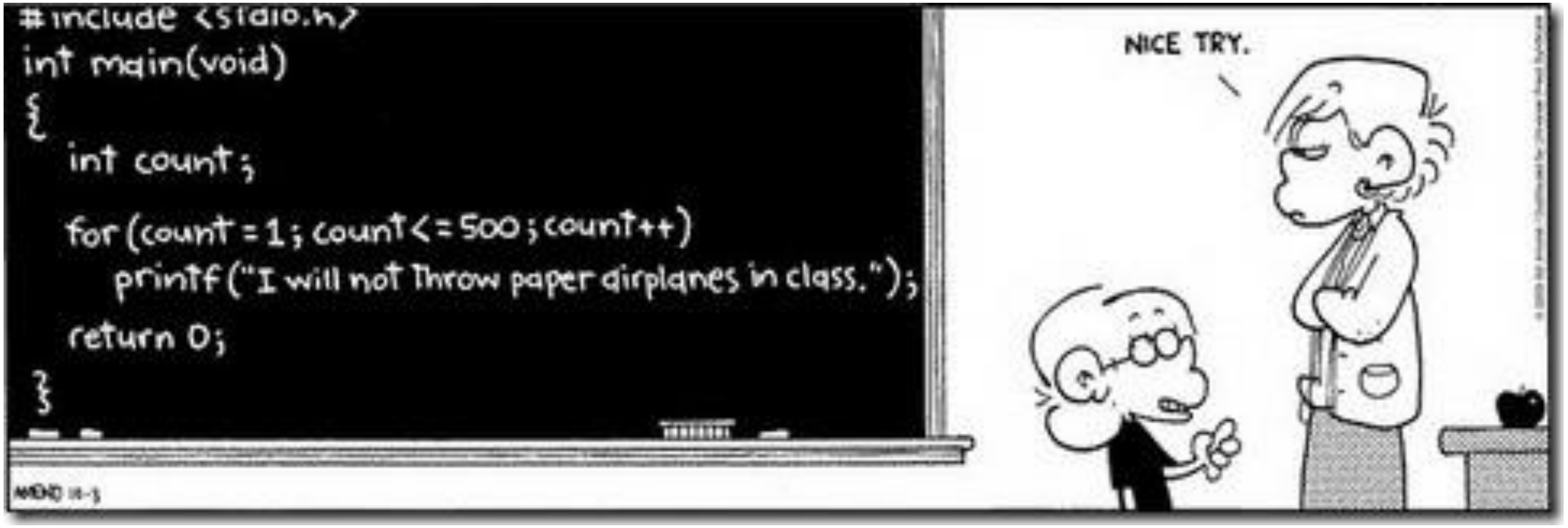
What is mulval at the end of this code block?

Other for loop issues and fun

- every for loop can be turned into a while loop
- How many times do the following loops execute?
  - `for (i = 0; i < 100; i++)`
  - `for (i = 1; i <= 100; i++)`
  - `for (i = 1; i < 100; i++)`
  - `for (i = 1; i <= 100; i++)`
- one ‘;’ can ruin your whole day

```
int sum = 0, addval;
for ( addval = 3; addval < 100; addval += 3);
     sum += addval;
```

For loops can make life better ☺



```
#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;
}
```

Do...While

```
do
{
      // processing statements
}
while (condition);
```

- Do loop guaranteed to be executed *at least once*

break and continue

- `break` and `continue` are special keywords used inside of loops
  - `break` – immediately stop processing the loop, go to first statement that follows the entire loop.
  - `continue` – stop processing this iteration of the loop. If a for loop, execute the “update statement”, then go to the top of the loop. While/do While, go to the top of the loop
- Why write code with break and/or continue?
  - special handling of particular cases become apparent

A common use of break

- Code is searching for the **first** occurrence of a particular condition, but will only search for so long.

```
for (int i = 0; i < MAX; i++)
{
       if (function(i) < 0)
              break;
}
if (i < MAX)
        System.out.println(i);
else
        System.out.println(”Function is >= 0”);
```

Common Use of Continue

- You only want to process occurrences that have (or have not) met a condition

```
Student pupil;
Course myClass;
while ((pupil = myClass.nextStudent()) != null)
{
        if (!pupil.tookMidterm())
                continue;
        // only proces if midterm was taken.
        grade = pupil.computeGrade();
        myClass.record(pupil.getName(),grade);
        pupil.emailGrade(grade);
}
```

Java's switch statement

```
switch ( variable_to_test )
{
      case value1:
             code_to_execute;
             break;
      case value2:
             code_to_execute2;
             break;
      case value3:
             code_to_execute3;       // No break stmt. This and next block
      case value4:                   // Will be executed
             code_to_execute4;
             break;
                  default:
             When nothing else matches, do this;

}
```

- Variable to test must be char, short, integer, or long primitive types.

Newer Java allows String types

- Values must be literal constants

Java's switch statement

```
int month = 8;
        String quarter;
        switch (month) {
            case 1:
            case 2:
            case 3:  quarter = "Winter";
                     break;
            case 4:
            case 5:
            case 6:  quarter = "Spring";
                     break;
            case 7:
            case 8:
            case 9:  quarter = "Summer";
                     break;
            case 10:
            case 11:
            case 12: quarter = "Fall";
                     break;
            default: quarter = "Invalid month";
                     break;
        }
        System.out.println(quarter);
```