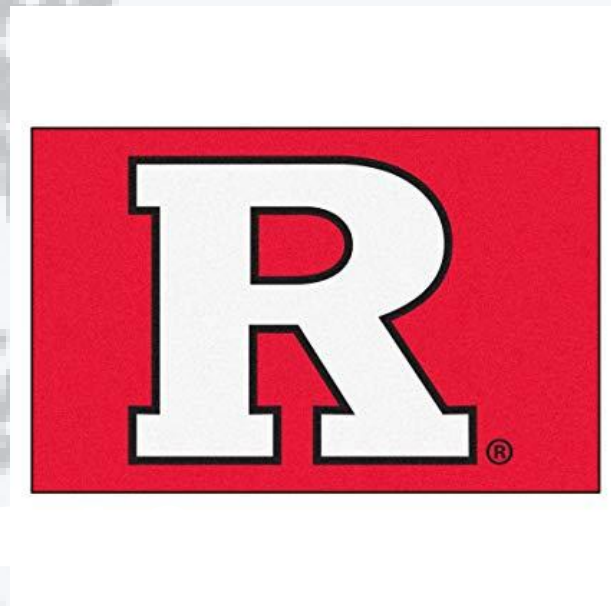


4. Conditionals and Loops

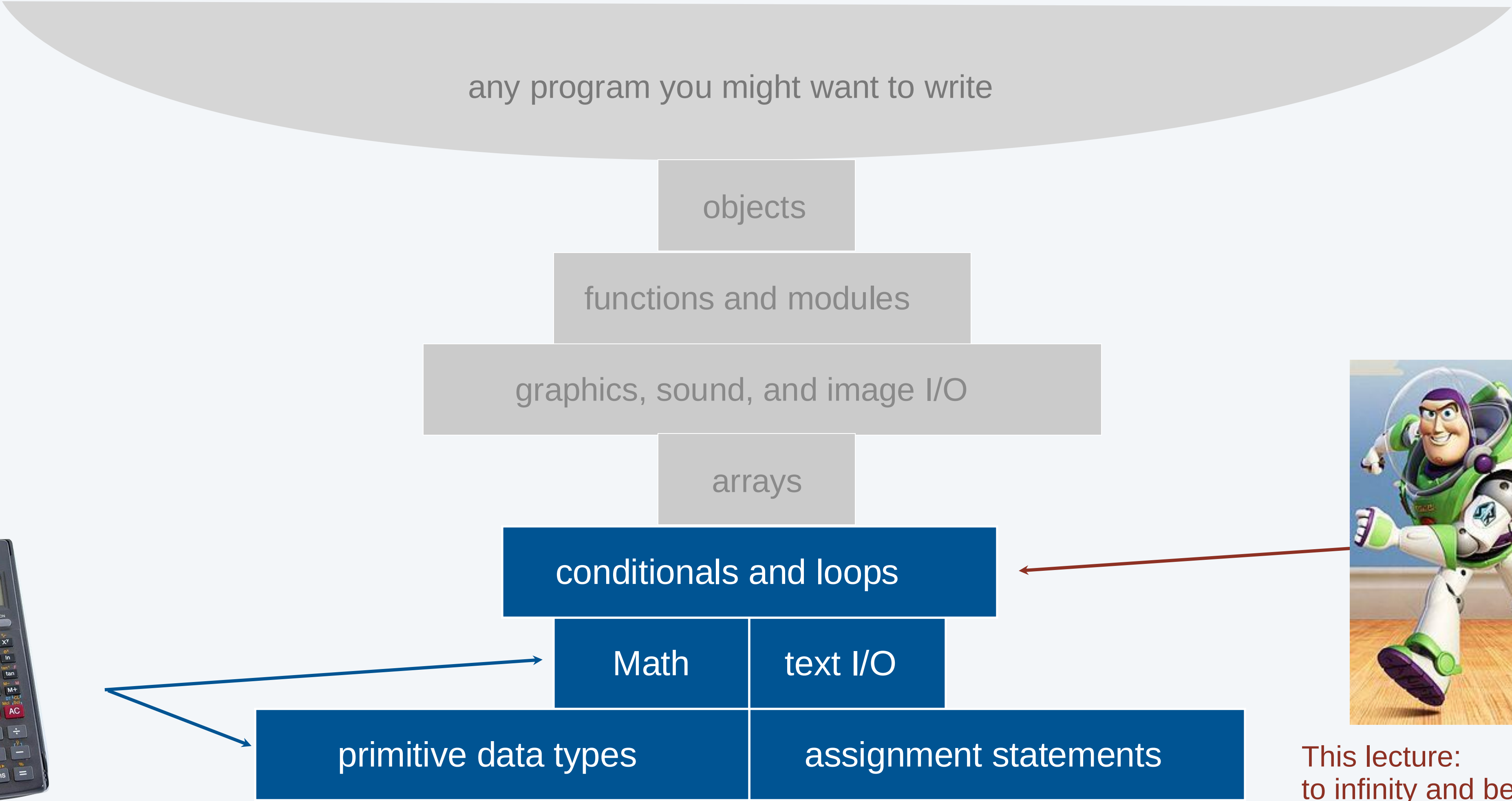


- REMEMBER: Socrative is OPEN (b.Socrative.com) Room: TREESCS111
- Teacher's Question: What went well? what did not go so well? What do I need to reteach? Do you want anything "recorded?"

4. Conditionals & Loops

- Conditionals: the if statement
- Loops: the while statement
- An alternative: the for loop
- Nesting
- Debugging

Context: basic building blocks for programming



This lecture:
to infinity and beyond!

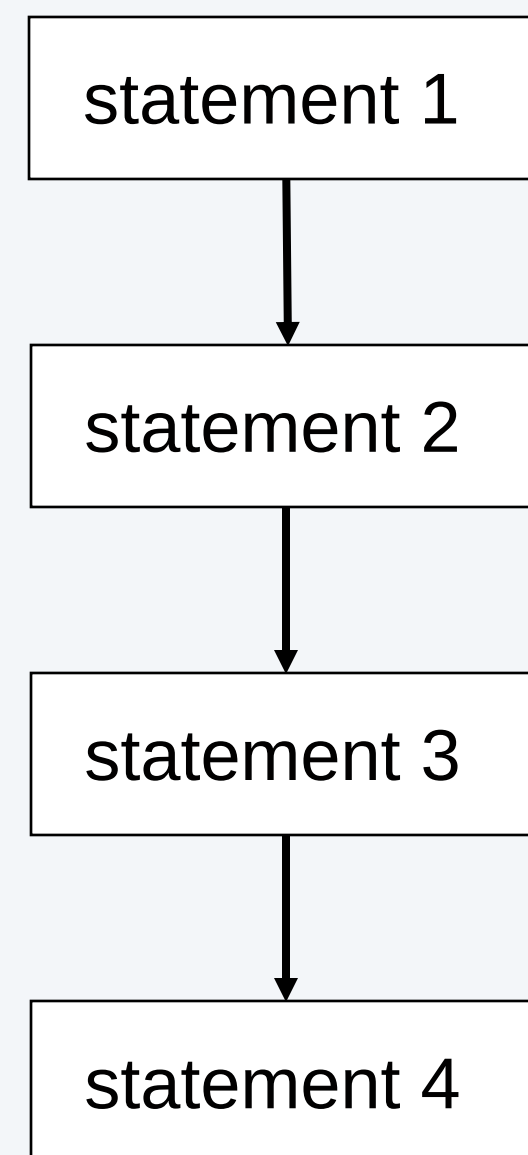


Previous lecture:
equivalent to a calculator

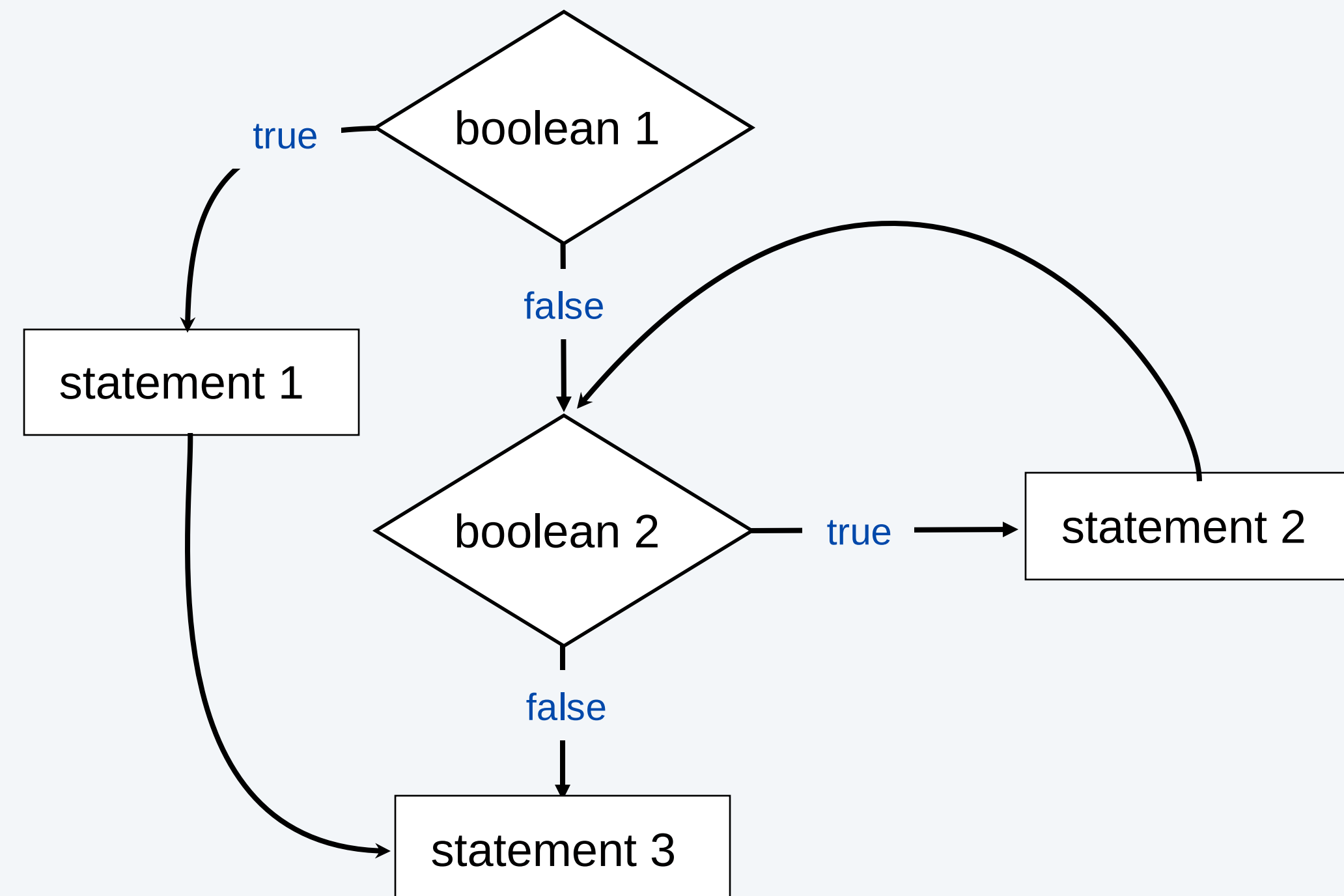
Conditionals and Loops – **WE DID THIS IS PSEUDOCODE**

Control flow

- The sequence of statements that are actually executed in a program.
- **Conditionals and loops** enable us to choreograph control flow.



straight-line control flow
[previous lecture]



control flow with conditionals and a loop
[this lecture]

Math class, Conditionals, Loops

BY EXAMPLES

Let's look at some problems using conditionals and Math.random()

1. Write a Java program to simulate the flipping of a fair coin. Display the result.

INPUT: N/a

OUTPUT: "heads" or "tails"

ERROR CONDITIONS: N/A

Name your program Coin1.java

YOUR TURN: IN VSC

Math.random()	[0,1)
(int)(Math.random() * 2 + 1)	1 or 2
(int) (Math.random() *(b – a + 1) + a <i>(b – a + 1) is the number of numbers on the range</i> <i>a is the smallest number in the range</i>	[a, b]

Example of if statement use: simulate a coin flip

```
public class Flip
{
    public static void main(String[] args)
    {
        if (Math.random() < 0.5)
            System.out.println("Heads");
        else
            System.out.println("Tails");
    }
}
```

Notice – no braces

% java Flip
Heads

% java Flip
Heads

% java Flip
Tails

% java Flip
Heads

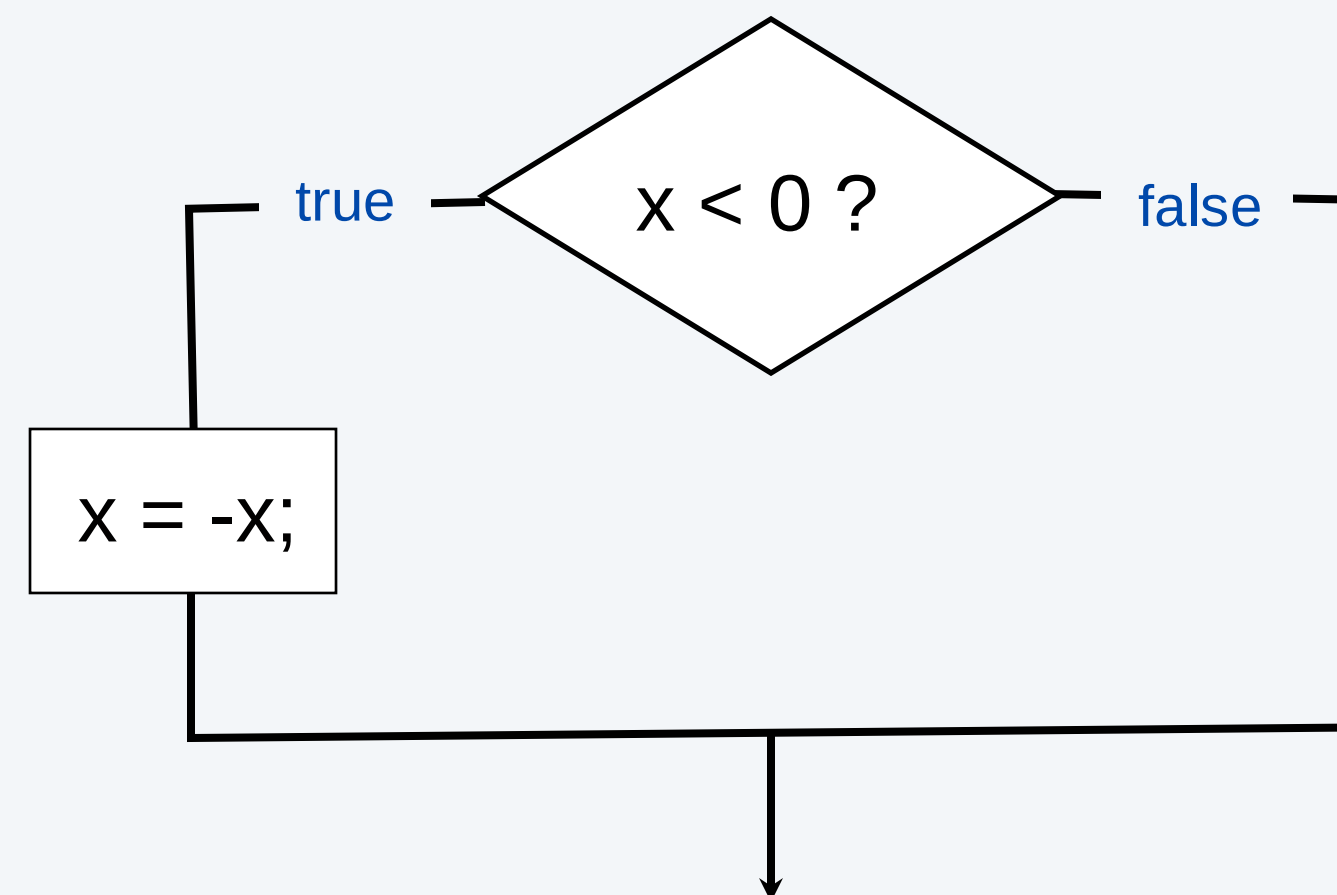


The if statement - **WE DID THIS IS PSEUDOCODE**

Execute certain statements depending on the values of certain variables.

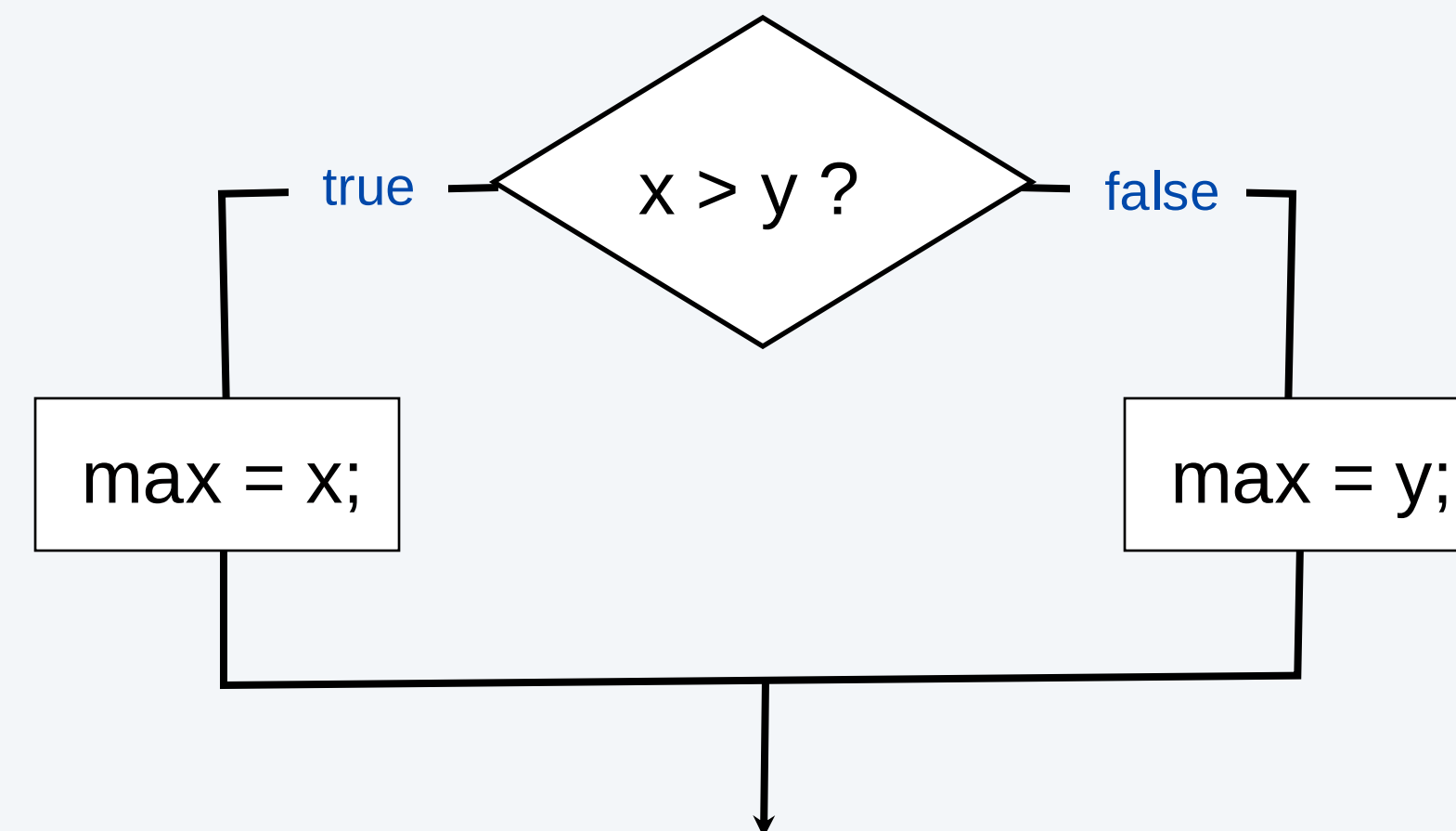
- Evaluate a boolean expression.
- If true, execute a statement.
- The **else option**: If false, execute a different statement.

Example: if ($x < 0$) $x = -x$;



Replaces x with the absolute value of x

Example: if ($x > y$) $\text{max} = x$;
else $\text{max} = y$;



Computes the maximum of x and y

4. Conditionals & Loops

- Conditionals: the if statement
- Loops: the while statement
- An alternative: the for loop
- Nesting
- Debugging

The while loop **WE DID THIS IS PSEUDOCODE**

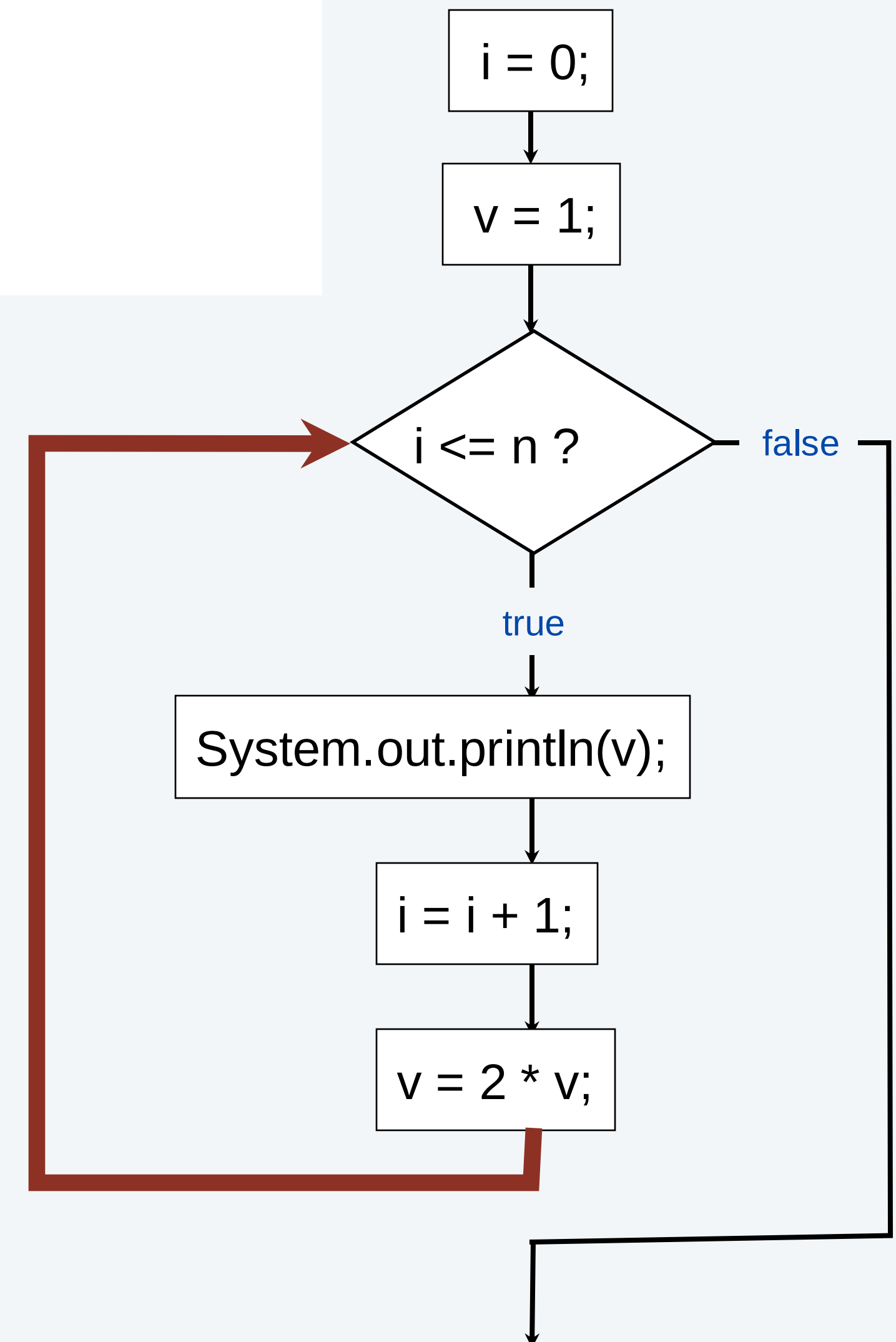
Execute certain statements repeatedly until certain conditions are met.

- Evaluate a boolean expression.
- If true, execute a sequence of statements.
- **Repeat.**

Example:

```
int i = 0;
int v = 1;
while (i <= n)
{
    System.out.println(v);
    i = i + 1;
    v = 2 * v;
}
```

IMPLEMENT!
Show variable values at each step
Number of operations?



Prints the powers of two from 2^0 to 2^n .

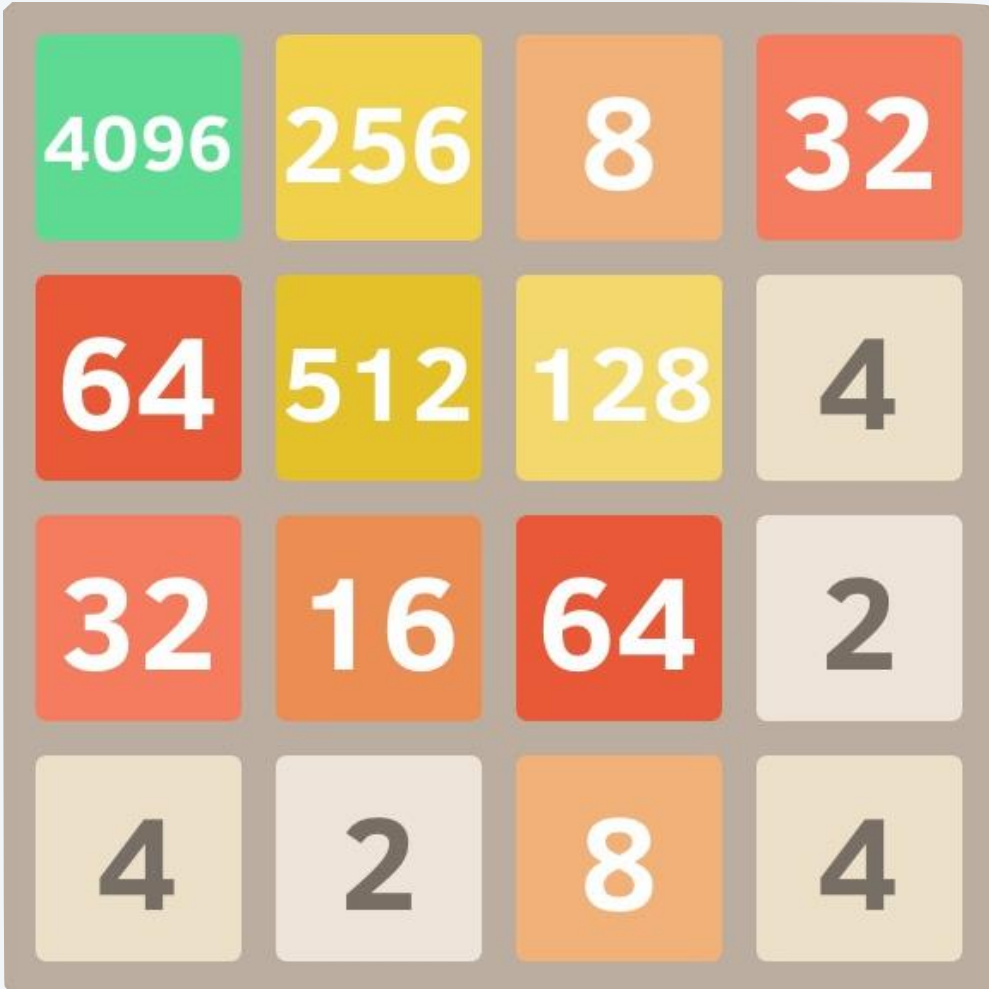
A **trace** is a table of variable values after each statement.

```
public class PowersOfTwo
{
    public static void main(String[] args)
    {
        int n = Integer.parseInt(args[0]);
        int i = 0;
        int v = 1;

        while (i <= n)
        {
            System.out.println(v);
            i = i + 1;
            v = 2 * v;
        }
    }
}
```

Prints the powers of two from 2^0 to 2^n .

i	v	i <= n
0	1	true
1	2	true
2	4	true
3	8	true
4	16	true
5	32	true
6	64	true
7	128	false



Game: 2048

```
% java PowersOfTwo 6
1
2
4
8
16
32
64
```

Counting operations is a good way to know how well your program works.

```
public class PowersOfTwo
```

```
{
```

```
    public static void main(String[] args)
```

```
    {
```

```
        int n = Integer.parseInt(args[0]); ← 1 initialize operation
```

```
        int i = 0; ← 1 initialize operation
```

```
        int v = 1; ← 1 initialize operation
```

```
        while (i <= n) ← Loop condition checked for i= 0..n+1 (n+2 times)  
                        Each instruction in loop body is executed for i = 0..n (n+1 times) : 3 instructions
```

```
        {
```

```
            System.out.println(v); ← 1 print operation
```

```
            i = i + 1; ← 1 increment/assignment operation
```

```
            v = 2 * v; ← 1 multiply/assignment operation
```

```
        }
```

```
    } Prints the powers of two from  $2^0$  to  $2^n$ .
```

Total operations
 $3 + (n+2) + 3(n+1)$

Back to our COIN: conditionals and Math.random() and a WHILE loop

1. Write a Java program to simulate the flipping of **a fair coin 10 times**. Display the flips.
Use a **WHILE** loop. COUNT the operations. *Name your program Coin2.java*

```
int count = 0;
while (count < 10){
    code goes here
```

```
    count ++;          // count++ uses Java's increment operator: count = count + 1
}
```

YOUR TURN: IN VSC

NOTE: You need to know how to stop an "endless (infinite) loop"

MAC: Control z

WINDOWS: Control c?

Back to our COIN: conditionals and Math.random() and a WHILE loop

1. Write a Java program to simulate the flipping of **a fair coin 10 times**. Display the flips.
Use a WHILE loop. COUNT the operations. **Coin2.java**

int count = 0;	<i>1 operation</i>
while (count < 10) {	<i>9 - 0 + 1 + 1 operations = 11 operations</i>
int flip = (int)(Math.random() * 2 + 1);	<i>10 operations</i>
if (flip == 1){	<i>10 operations</i>
System.out.println("HEADS");	
}	
else{	<i>10 SOP operations</i>
System.out.println("TAILS");	
}	
count++;	<i>10 operations</i>
}	<i>Total operations = 1 + 11 + 40 = 52</i>

Back to our COIN: conditionals and Math.random() and a WHILE loop

Write a Java program to simulate the flipping of **a fair coin n times, where n is input by the user.** Display the flips. **Modify Coin2.java (Coin2Modified.java)**

Use a WHILE loop. COUNT the operations.

```
int n = Integer.parseInt(args[0]);
int count = 0;
while (count < n) {
    int flip = (int)(Math.random() * 2 + 1);
    if (flip == 1){
        System.out.println("HEADS");
    }
    else{
        System.out.println("TAILS");
    }
    count++;
}
```

YOUR TURN: IN VSC

Back to our COIN: conditionals and Math.random() and a WHILE loop

Write a Java program to simulate the flipping of **a fair coin n times, where n is input by the user.** Display the flips. **Coin2.java MODIFIED**

Use a WHILE loop. COUNT the operations.

```
int n = Integer.parseInt(args[0]);
```

1 operation

```
int count = 0;
```

1 operation

```
while (count < n) {
```

$n - 0 + 1 + 1 \text{ operations} = n + 2 \text{ operations}$

```
    int flip = (int)(Math.random() * 2 + 1);
```

$n + 1 \text{ operations}$

```
    if (flip == 1){
```

$n + 1 \text{ operations}$

```
        System.out.println("HEADS");
```

```
    }
```

```
    else{
```

$n + 1 \text{ SOP operations}$

```
        System.out.println("TAILS");
```

```
    }
```

```
    count++;
```

$n + 1 \text{ operations}$

```
}
```

$\text{Total operations} = 5(n + 1) + 3 = 5n + 8$

Back to our COIN: conditionals and Math.random() and a WHILE loop

ANOTHER MODIFICATION: **Coin3.java**

Write a Java program to simulate the flipping of **a fair coin n times**, where n is input by the **user**. Display the **number of heads** that were flipped. *Name your program Coin3.java*

YOUR TURN: IN VSC

4. Conditionals & Loops

- Conditionals: the if statement
- Loops: the while statement
- An alternative: the for loop
- Nesting
- Debugging

The **while** loop

The **for** loop

```
int k = 0;
while (k < TIMES)
{
    // do stuff;
    k += 1;
}
```

```
for(int k=0; k < TIMES; k++)
{
    // do stuff
}
```

The for loop has the *initialization, test, update* in one place.

Operations - WHILE

1
 $((\text{TIMES} - 1) - 0 + 1) + 1 = \text{TIMES} + 1$
 TIMES (for each instruction in *do stuff*)
 TIMES
 TOTAL = 3 X TIMES + 2

Operations -- FOR

1 for init
 TIMES + 1 for test
 TIMES for increment
 TIMES for each instruction in *do stuff*
 TOTAL = 3 X TIMES + 2

An alternative repetition structure.

← Why? Can provide code that is more compact and understandable.

- Evaluate an *initialization statement*.
- Evaluate a *boolean expression*.
- If true, execute a *sequence of statements*, then execute an *increment statement* (**UPDATE statement**).
- **Repeat.**

Example:

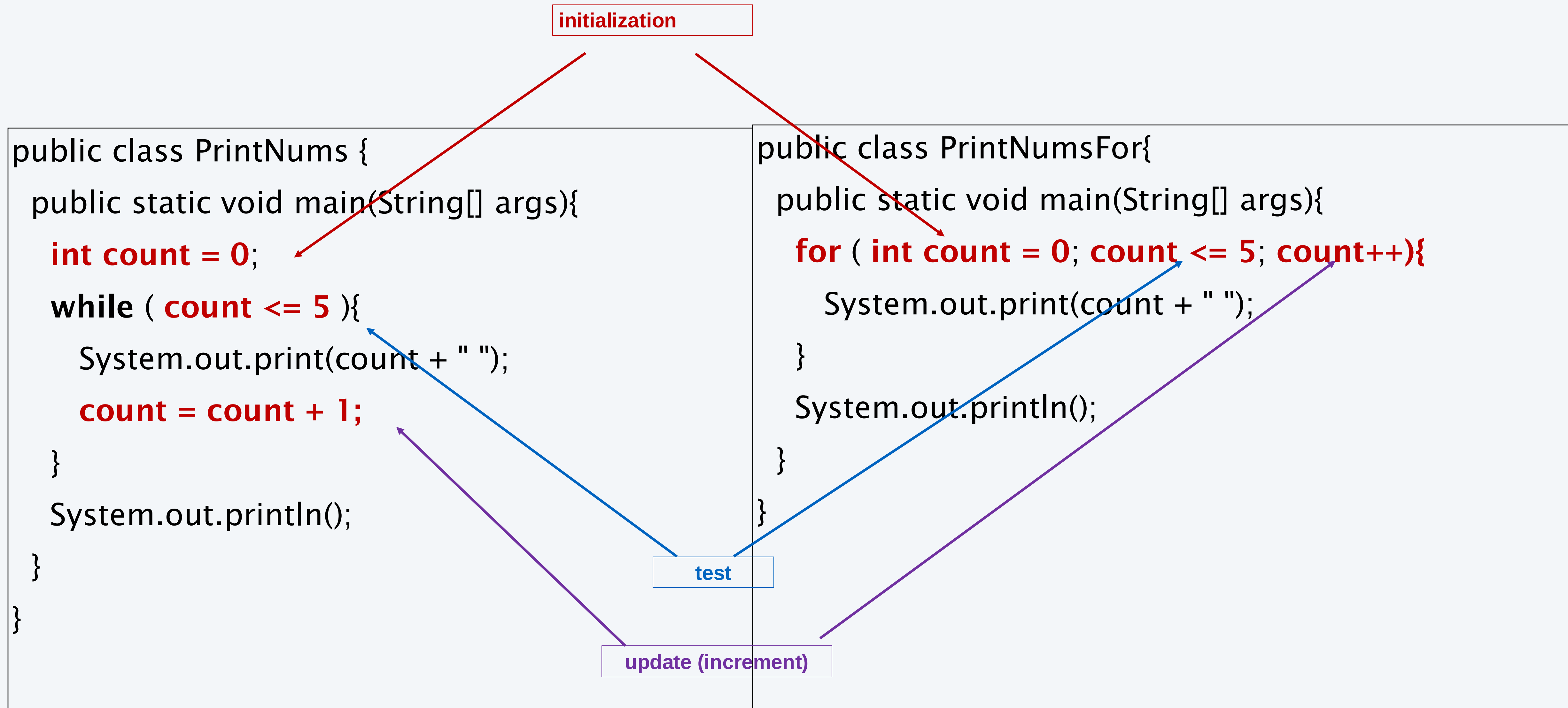
```
int v = 1;
for (int i = 0; i <= n; i++)
{
    System.out.println( i + " " + v );
    v = 2*v;
}
```

Prints the powers of two from 2^0 to 2^n

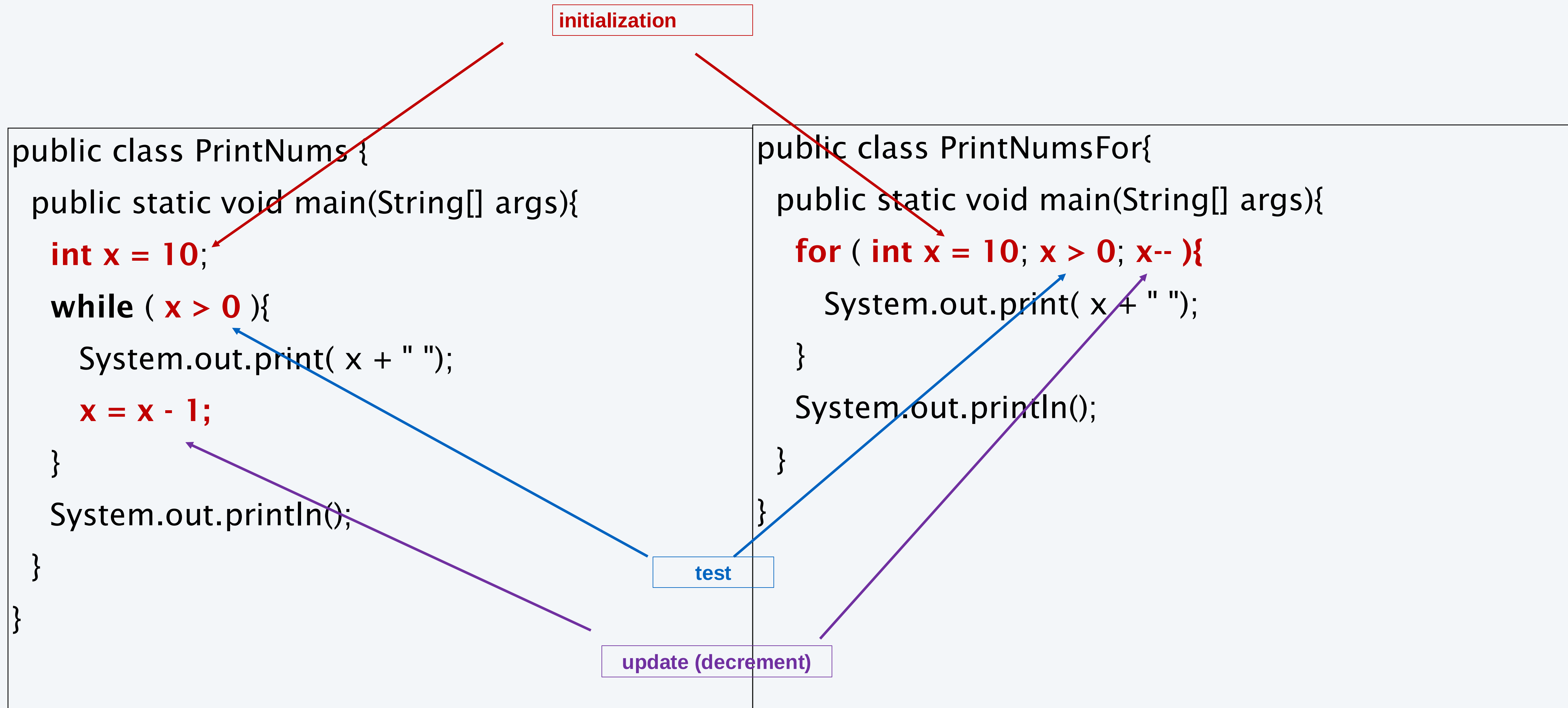
Every **for** loop has an equivalent **while** loop:

```
int v = 1;
int i = 0;
while ( i <= n )
{
    System.out.println( i + " " + v );
    v = 2*v;
    i++;
}
```

Java program (while vs. for)



Java program (while vs. for)



```
int sum = 0;
for (int i = 1; i <= N; i++)
    sum += i;
System.out.println(sum);
```

Compute sum $(1 + 2 + 3 + \dots + N)$

sum	i
1	1
3	2
6	3
10	4

trace at end of loop for $N = 4$

product	i
1	1
2	2
6	3
24	4

```
long product = 1;
for (int i = 1; i <= N; i++)
    product *= i;
System.out.println(product);
```

Compute $N! = 1 * 2 * 3 * \dots * N$

```
for (int k = 0; k <= N; k++)
    System.out.println(k + " " + 2*Math.PI*k/N);
```

Print a table of function values

k	$\frac{2\pi k}{N}$
0	0
1	1.57079632...
2	3.14159265...
3	4.71238898...
4	6.28318530...

```
int v = 1;
while (v <= N/2)
    v = 2*v;
System.out.println(v);
```

Print largest power of 2 less than or equal to N

v
2
4
8
16

trace at end of loop for $N = 23$

Scope of `for` loop variables

If a variable is declared in the `for` loop header, its scope extends to the end of the `for` loop.

```
for(int x = 1; x <= n; x++) {  
    // x can be accessed here  
}  
  
// x is no longer defined.
```

You can declare before the `for` loop and initialize in the `for` loop.

```
int x;  
for(x = 1; x <= n; x++) {  
    // x can be accessed here  
}  
  
// x is still in scope here
```

Back to our COIN: conditionals and Math.random() and a FOR loop

ANOTHER MODIFICATION: COIN4

Write a Java program to simulate the flipping of **n fair coins, where n is input by the user.**
Display the **number of heads** that were flipped. *Name your program Coin4.java*

```
count = 0;
while (count < n){ for (int count = 0; count < n; count++){
    int flip = (int)(Math.random() * 2 + 1);
    if (flip == 1){
        System.out.println("HEADS");
        numHeads++;
    }
    else{
        System.out.println("TAILS");
    }
count++;
}
System.out.println("The number of HEADS flipped is: " + numHeads);
}
```

That was an OVERVIEW!!!!

We'll look at each topic in detail!

We need to be able to TRACE algorithms and WRITE algorithms!

- Conditionals
- While loop
- For loop
- Counting operations

if statement

Java Reference Guide

<https://rutgers.app.box.com/file/533536224328?s=y3rfunnyri4ilszevy5ieombwbdjm6rv5>

Decisions
<pre>if (<i>condition</i>) { }</pre>
<pre>if (<i>condition</i>) { } else { }</pre>
<pre>if (<i>condition</i>) { } else if (<i>condition</i>) { } ... else { }</pre>

Example of **if** statement use2-sort

Q. What does this program do?

```
public class TwoSort
{
    public static void main(String[] args)
    {
        int a = Integer.parseInt(args[0]);
        int b = Integer.parseInt(args[1]);
        if (b < a)
        {
            int t = a;
            a = b;
            b = t;
        }
        System.out.println(a);
        System.out.println(b);
    }
}
```

alternatives for if and else
can be a *sequence* of
statements, enclosed in braces

Let's trace this

```
% java TwoSort 1234 99
99
1234
```

a	b	t
1234	99	
1234	99	1234
99	99	1234
99	1234	1234

```
% java TwoSort 99 1234
99
1234
```

a	b	t
99	1234	
99	1234	

A. Reads two integers from the command line, then prints them out in numerical order.

Pop quiz on **if** statements

Add code to this program that puts a, b, and c in numerical order.

```
public class ThreeSort
{
    public static void main(String[] args)
    {
        int a = Integer.parseInt(args[0]);
        int b = Integer.parseInt(args[1]);
        int c = Integer.parseInt(args[2]);

        System.out.println(a);
        System.out.println(b);
        System.out.println(c);
    }
}
```

```
% java ThreeSort 1234 99 1
1
99
1234
```

```
% java ThreeSort 99 1 1234
1
99
1234
```

Pop quiz on **if** statements

Add code to this program that puts a, b, and c in numerical order.

```
public class ThreeSort
{
    public static void main(String[] args)
    {
        int a = Integer.parseInt(args[0]);
        int b = Integer.parseInt(args[1]);
        int c = Integer.parseInt(args[2]);
        if (b < a)
        { int t = a; a = b; b = t; }
        if (c < a)
        { int t = a; a = c; c = t; }
        if (c < b)
        { int t = b; b = c; c = t; }
        System.out.println(a);
        System.out.println(b);
        System.out.println(c);
    }
}
```

← makes *a* smaller than *b*

← makes *a* smaller than both *b* and *c*

← makes *b* smaller than *c*

```
% java ThreeSort 1234 99 1
1
99
1234
```

a	b	c	t
1234	99	1	
99	1234	1	1234
1	1234	99	99
1	99	1234	1234

```
% java ThreeSort 99 1 1234
1
99
1234
```

a	b	c	t
99	1	1234	
1	99	1234	

Example of **if** statement use: error checks (IntOps.java MODIFIED TO IntOps2.java)

```
public class IntOps2
{
    public static void main(String[] args)
    {
        int a = Integer.parseInt(args[0]);
        int b = Integer.parseInt(args[1]);
        int sum = a + b;
        int prod = a * b;
        System.out.println(a + " + " + b + " = " + sum);
        System.out.println(a + " * " + b + " = " + prod);
        if (b == 0) System.out.println("Division by zero");
        else      System.out.println(a + " / " + b + " = " + a / b);
        if (b == 0) System.out.println("Division by zero");
        else      System.out.println(a + " % " + b + " = " + a % b);
    }
}
```

% java IntOps2 5 2

5 + 2 = 7

5 * 2 = 10

5 / 2 = 2

5 % 2 = 1

% java IntOps2 5 0

5 + 0 = 5

5 * 0 = 0

Division by zero

Division by zero

Good programming practice. Use conditionals to check for *and avoid* runtime errors.

Example of nesting conditionals: Tax rate calculation

Goal. Given income, calculate proper tax rate.

income	rate
0 – \$47,450	22%
\$47,450 – \$114,649	25%
\$114,650 – \$174,699	28%
\$174,700 – \$311,949	33%
\$311,950 +	35%

Taxes.java

```
if (income < 47450) rate = 0.22;
else
{
    if (income < 114650) rate = 0.25;
    else
    {
        if (income < 174700) rate = 0.28;
        else
        {
            if (income < 311950) rate = 0.33;
            else
                rate = 0.35;
        }
    }
}
```

if statement
within an if statement

if statement
within an if statement
within an if statement

if statement
within an if statement
within an if statement
within an if statement

Q. Anything wrong with the following code?

```
public class PQif
{
    public static void main(String[] args)
    {
        double income = Double.parseDouble(args[0]);
        double rate = 0.35;
        if (income < 47450) rate = 0.22;
        if (income < 114650) rate = 0.25;
        if (income < 174700) rate = 0.28;
        if (income < 311950) rate = 0.33;
        System.out.println(rate);
    }
}
```

Pop quiz on nested if statements

Q. Anything wrong with the following code?

```
public class PQif
{
    public static void main(String[] args)
    {
        double income = Double.parseDouble(args[0]);
        double rate = 0.35;
        if (income < 47450) rate = 0.22;
        if (income < 114650) rate = 0.25;
        if (income < 174700) rate = 0.28;
        if (income < 311950) rate = 0.33;
        System.out.println(rate);
    }
}
```

A. Yes! Need else clauses.
Without them, code is equivalent to:

```
if (income < 311950) rate = 0.33;
else                rate = 0.35;
```

Example of nesting conditionals: Tax rate calculation

Goal. Given income, calculate proper tax rate.

income	rate
0 – \$47,450	22%
\$47,450 – \$114,649	25%
\$114,650 – \$174,699	28%
\$174,700 – \$311,949	33%
\$311,950 +	35%

Note. Braces are not needed in this case but BE CAREFUL when nesting if-else statements because of potential ambiguity.

```
if (income < 47450)
    rate = 0.22;
else if (income < 114650)
    rate = 0.25;
else if (income < 174700)
    rate = 0.28;
else if (income < 311950)
    rate = 0.33;
else
    rate = 0.35;
```

else ifs – Taxes2.java

Write the Java code; Test for your Test Cases; Count operations in your algorithm.

Problem. A cake up to 6" cost \$10, 6"-12" cost \$15, greater than 12" cost \$20. Given the diameter of the cake, find the cost.

Solution Design. Input: diameter of the cake (integer)
Output: the price
Error: diameters less than or equal to 0

One possibility

```
public class Cake{
    Run | Debug
    public static void main(String[] args){
        double diameter = Double.parseDouble(args[0]);
        double price = 0.0;
        if (diameter <= 0){
            System.out.println("Error");
        }
        else if (diameter < 6){
            price = 10;
        }
        else if (diameter <= 12){
            price = 15.0;
        }
        else{
            price = 20.0;
        }
        System.out.println("Price of cake is $" + price);
    }
}
```

Number of operations?

Min =

Max =

WHILE loop

Java Reference Guide

<https://rutgers.app.box.com/file/533536224328?s=y3rfunnyri4ilnzevy5ieombwbdjm6rv5>

Loops

```
while (condition)  
{  
    ....  
}
```

Coding with while loops must include:

1. Initialization of variable used in test
2. Test
3. Update of variable

Q. Anything wrong with the following code?

```
public class PQwhile
{
    public static void main(String[] args)
    {
        int n = Integer.parseInt(args[0]);
        int i = 0;
        int v = 1;
        while (i <= n)
            System.out.println(v);
            i = i + 1;
            v = 2 * v;
        }
    }
```

Q. Anything wrong with the following code?

```
public class PQwhile
{
    public static void main(String[] args)
    {
        int n = Integer.parseInt(args[0]);
        int i = 0;
        int v = 1;
        while (i <= n)
        {
            System.out.println(v);
            i = i + 1;
            v = 2 * v;
        }
    }
}
```

A. Yes! Needs braces.

Q. What does it do (without the braces)?

A. Goes into an *infinite loop*.

```
% java PQwhile 6
1
1
1
1
1
1
1
1
1
1
1
```

KNOW HOW TO STOP AN ENDLESS LOOP ON YOUR COMPUTER! (control Z on mine!)

← challenge: figure out
how to stop it
on your computer



Write Java code to solve the following problem. Use a WHILE loop in your solution.

Find the largest digit in an integer input by the user. Assume the input >0.

Write Java code to solve the following problem. Use a WHILE loop in your solution.

Find the largest digit in an integer input by the user. Assume the input >0.

```
public class LargestDig{
    Run | Debug
    public static void main(String[] args){
        int num = Integer.parseInt(args[0]);
        int largest = 0;
        while (num > 0){
            int dig = num % 10;
            if (dig > largest){
                largest = dig;
            }
            num = num / 10;
        }
        System.out.println("The largest digit is: " + largest);
    }
}
```

Conditions

- What's wrong?
- What one change in the code below would cause in an infinite loop?

Example:

```
int i = 0;
while ( i > 1) {
    System.out.println(i);
    i++;
}
```

Conditions

- What's wrong?

Example:

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

← The value of i is always greater than 1

FOR loop

Java Reference Guide

<https://rutgers.app.box.com/file/533536224328?s=y3rfunnyri4ilnzevy5ieombwbdjm6rv5>

```
for (initialization ; condition ; update)  
{  
    ....  
}
```

Coding with for loops SHOULD include:

1. Initialization of variable used in test
2. Test condition
3. Update of variable

Q. What does the following program print?

```
public class PQfor
{
    public static void main(String[] args)
    {
        int f = 0, g = 1;
        for (int i = 0; i <= 10; i++)
        {
            System.out.println(f);
            f = f + g;
            g = f - g;
        }
    }
}
```

Pop quiz on **FOR** loops

Q. What does the following program print?

```
public class PQfor
{
    public static void main(String[] args)
    {
        int f = 0, g = 1;
        for (int i = 0; i <= 10; i++)
        {
            System.out.println(f);
            f = f + g;
            g = f - g;
        }
    }
}
```

A.

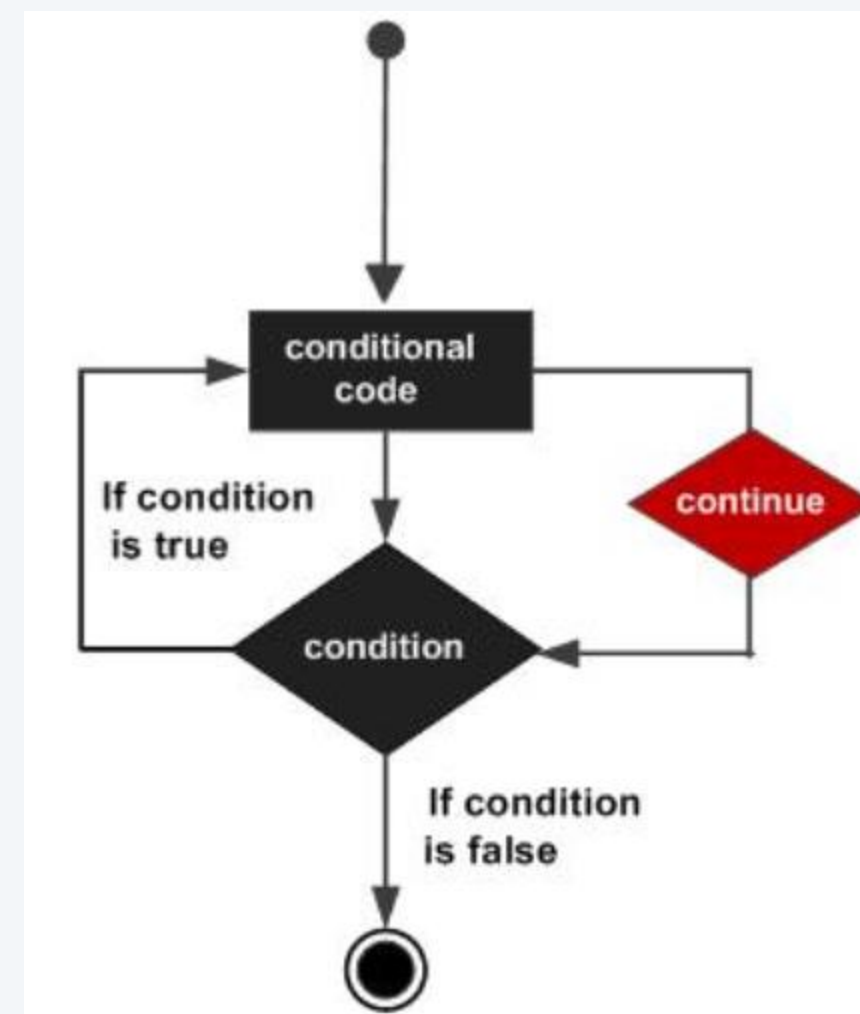
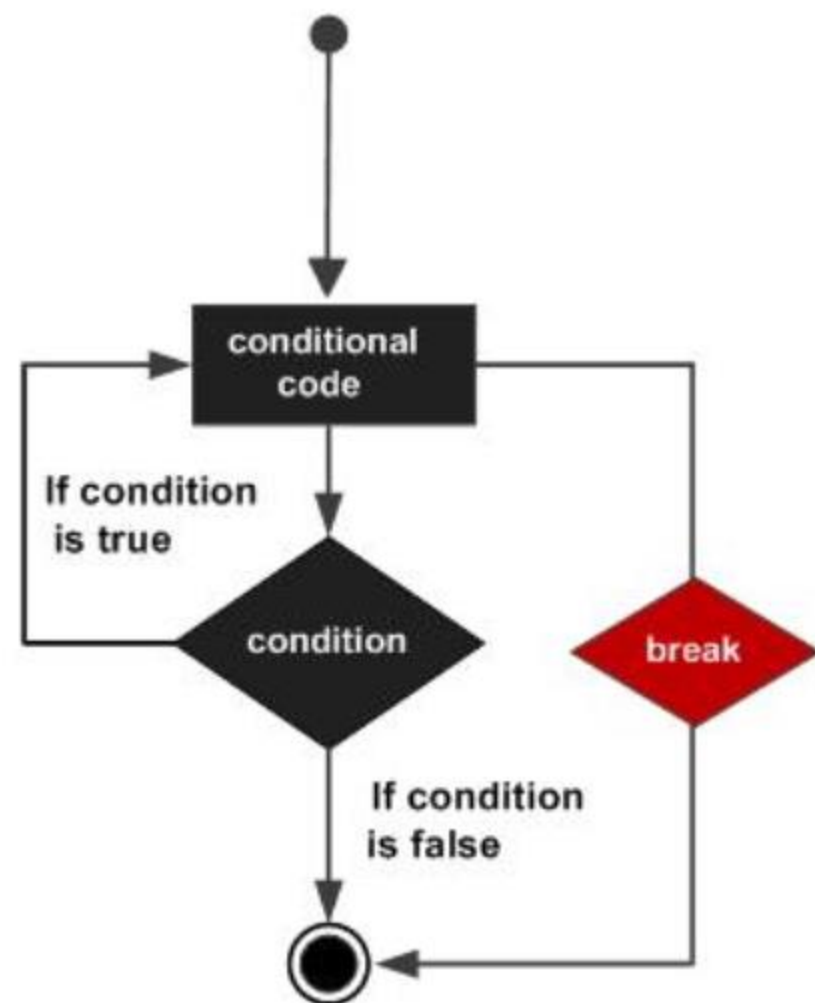
Beginning-of-loop trace

i	f	g
0	0	1
1	1	0
2	1	1
3	2	1
4	3	2
5	5	3
6	8	5
7	13	8
8	21	13
9	34	21
10	55	34

Printed values of f

break: exit the loop

continue: skip to end of the loop and continue to the next iteration



```

for (int i=0; i < n ; i++) {
    if ( i == 11)
        continue;
    System.out.println(i);
}
    
```

```

for (int i=0; i < n ; i++) {
    if ( i == 11)
        break;
    System.out.println(i);
}
    
```

break is executed when i is 11
which forces the end of the loop
(exit the loop)

Loop skips printing 11

PROBLEM with BREAK and CONTINUE

Write a Java a program that displays all numbers between 1 and n that are divisible by 3 but not by 2. Use a **for** loop with **continue**.

Modify your program stop checking numbers when it finds a number divisible by 11 (using **break**).

```
$ javac WithContBreak.java
$ java WithContBreak 50
3 9 15 21 27
```

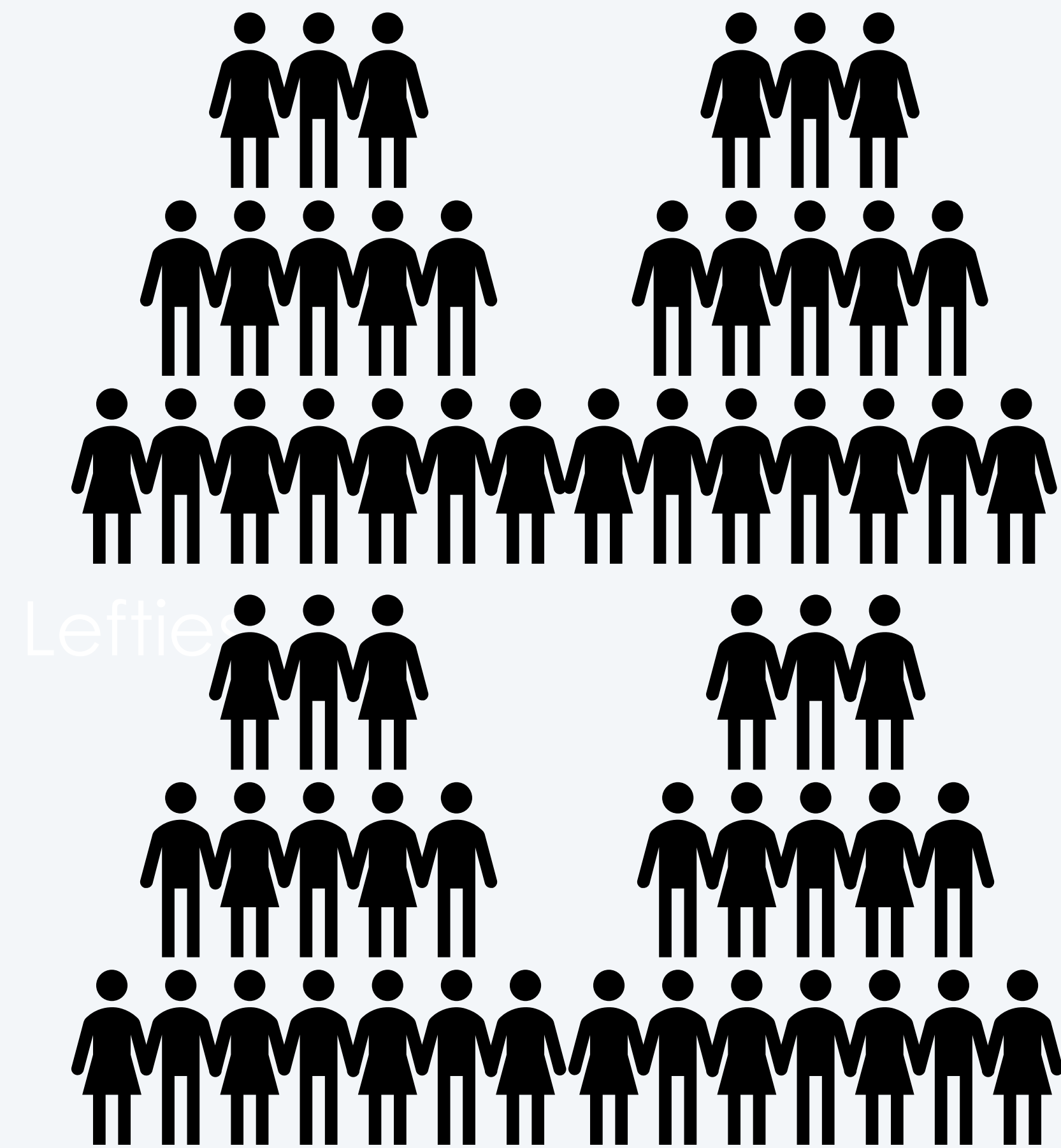
```
public class WithContBreak{
    Run | Debug
    public static void main(String[] args){
        int n = Integer.parseInt(args[0]);
        for(int i = 1; i <= n; i++){
            if(i % 3 == 0){
                if (i % 11 == 0){
                    break;
                }
                if (i % 2 == 0){
                    continue;
                }

                System.out.print(i + " ");
            }
        }
        System.out.println();
    }
}
```

4. Conditionals & Loops

- Conditionals: the if statement
- Loops: the while statement
- An alternative: the for loop
- **Nesting**
- Debugging

Nesting loops



- I know that 10% of the population is left-handed.
- To test this, I walk around campus and ask folks if they are left handed or not. I conduct **100 trials** with each trial consisting of 20 people. I am interested in knowing **how many trials** resulted in 10% or more left-handed people in the group of 20.

JUST DO IT!! (*Lefties.java*)

Q. In each of these cases, find an appropriate control structure

1. Given 3 integers, find the max value
2. Given n integers, find the max value
3. Print a n x n grid of integers

← nested if statements **maxNum.java**

← nested while-if statements (or for-if statements) **maxNumOfN.java**

← nested while-while (or for-for) statements **Grid.java**

Write a segment of code to produce the following: *Use nested Loops – Nest 1*

1	2	3	4
2	4	6	8
3	6	9	12
4	8	12	16
5	10	15	20

Questions to ask:

- How many rows?
- How many columns?
- Is the value in each position related to the row number or the column number?

Write a segment of code to produce the following: *Use nested Loops- Nest2*

```
1      1      1      1
2      2      2      2
3      3      3      3
4      4      4      4
5      5      5      5
```

Questions to ask:

- How many rows?
- How many columns?
- Is the value in each position related to the row number or the column number?

Write a segment of code to produce the following: *Use nested Loops – nest3*

```
1      2      3      4
1      2      3      4
1      2      3      4
```

Questions to ask:

- How many rows?
- How many columns?
- Is the value in each position related to the row number or the column number?

Write a segment of code to produce the following: *Use nested Loops – nest4*

```
1
2   2
3   3   3
4   4   4   4
5   5   5   5   5
```

Questions to ask:

- How many rows?
- How many columns?
- Is the value in each position related to the row number or the column number?

Write a segment of code to produce the following: *Use nested Loops – nest5*

```
5      5      5      5      5
4      4      4      4
3      3      3
2      2
1
```

Questions to ask:

- How many rows?
- How many columns?
- Is the value in each position related to the row number or the column number?

Determine if a positive integer is a palindrome.

Does the number read the same in reverse?

num = 123
rev = 0

Hints (NUMERICAL algorithm)

- What if the number is only one digit?
- How do I reverse the digits in a number?

dig = Num%10 = 3
rev = 10 * rev + dig = 0 + 3 = 3
num = num/10 = 12

dig = num%10 = 2
rev = 10 * rev + dig = 30 + 2 = 32
num = num/10 = 1

dig = num%10 = 1
rev = 10 * rev + dig = 320 + 1 = 321
num = num/10 = 0

rev = 321

4. Conditionals & Loops

- Conditionals: the if statement
- Loops: the while statement
- An alternative: the for loop
- Nesting
- Debugging

is 99% of program development in any programming language, *even for experts*.



Q. For each of the variables in the code, identify the scope of each variable

Scope of n is the main program

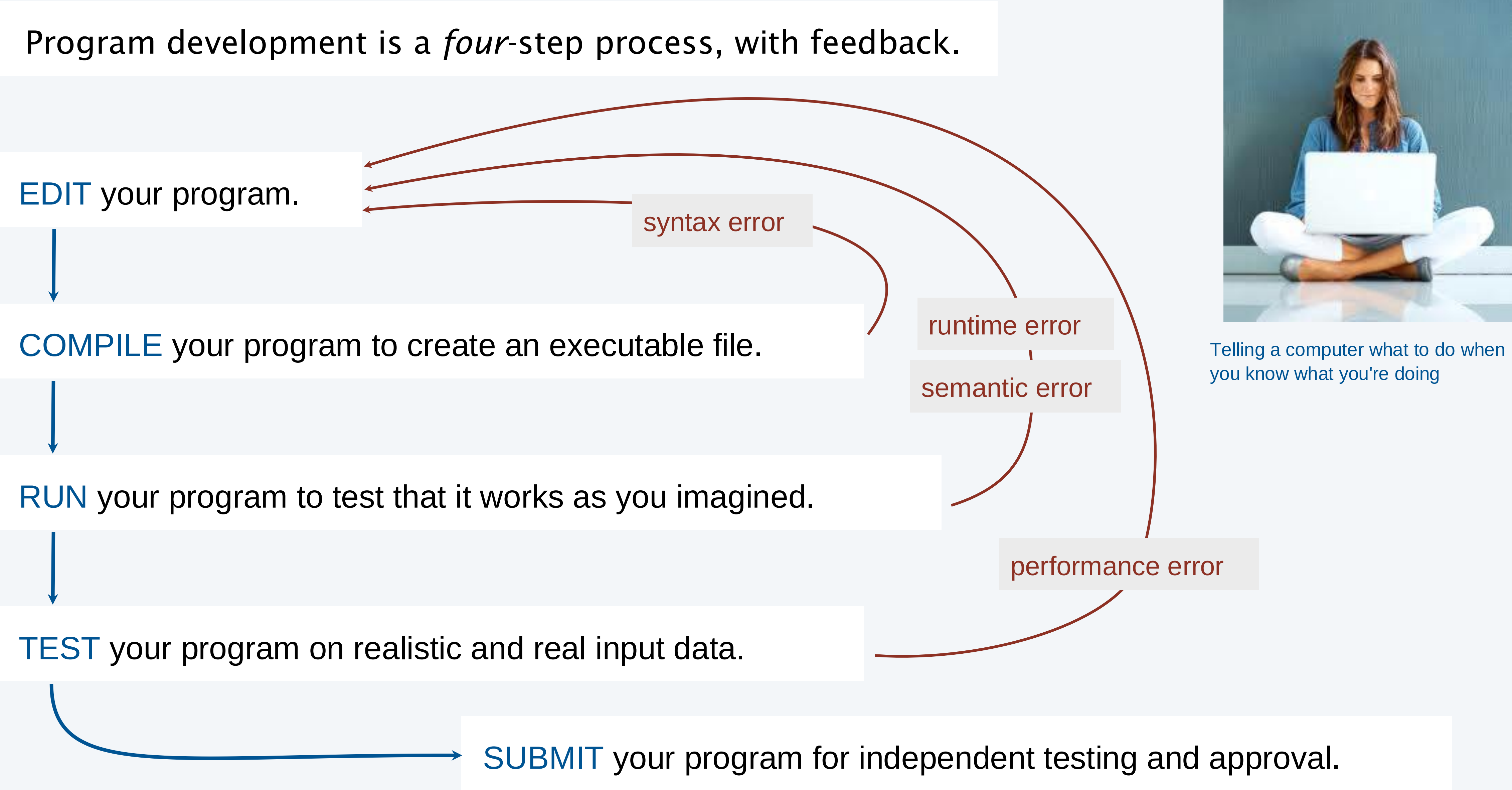
Scope of i is the for-loop body

Scope of num is the while-loop body

```
public class Factors
{
    public static void main(String[] args)
    {
        long n = Long.parseLong(args[0]);
        for ( int i = 2; i <= n/i; i++)
        {
            while (n % i == 0)
            {
                int num = i;
                System.out.print(num + " ");
                n = n / i;
            }
        }
        if (n > 1) System.out.println(n);
        else      System.out.println();
    }
}
```

Caution. Not understanding scope of a variable might lead to many wasted debugging hours

Debugging your program: summary



Craps.... *A Final Challenge*

Involves a while loop and conditionals

Rules for the game: In the game of *Craps*, players roll a pair of dice. Only the sum of the spots matters: a number between 2 and 12. The *roller* does the rolling. Below, *win* or *loss* refers to the roller.

Roller rolls two dice to get the **first sum**

If the **first sum** is 2, 3, or 12, roller loses

Otherwise, if the **first sum** is 7 or 11, roller wins

Otherwise, the **first sum** becomes the **point**

Now keep rolling again and again:

Roller rolls two dice to get the **new sum**

If the **new sum** equals the **point**, roller wins

Otherwise, if the **new sum** equals 7, roller loses

Otherwise, roller rolls again

The above is the simplest form of *street craps*. – *Craps.java*

Image sources

http://playatlantic.com/sites/default/files/good%20looking%20girl%20on%20laptop_0.jpg

http://en.wikipedia.org/wiki/Edsger_W._Dijkstra#mediaviewer/File:Edsger_Wybe_Dijkstra.jpg

<http://pixabay.com/en/hourglass-clock-timer-sand-time-152090/>

Image sources

http://commons.wikimedia.org/wiki/File:Calculator_casio.jpg

<http://en.wikipedia.org/wiki/File:Buzz-lightyear-toy-story-3-wallpaper.jpg>

<http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2789164/#!po=30.0000> [181e306f1.jpg]

Image sources

<http://www.sciencecartoonsplus.com>

http://en.wikipedia.org/wiki/Isaac_Newton

<http://www.onlinemathtutor.org/help/wp-content/uploads/math-cartoon-28112009.jpg>

Image sources

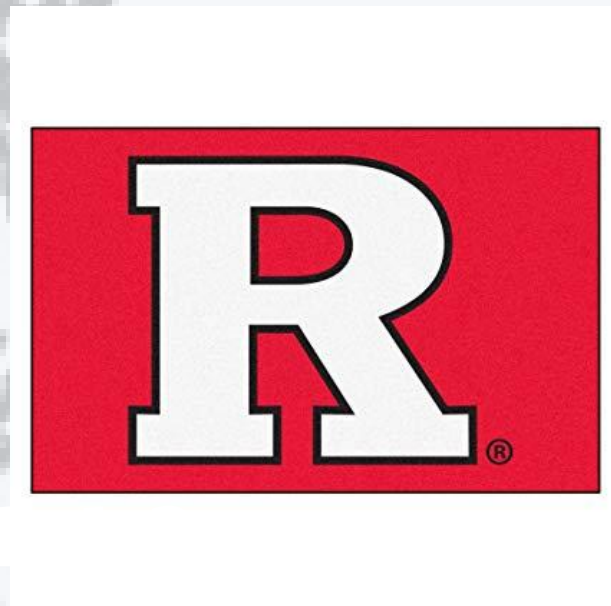
<http://pixabay.com/en/atlantic-city-ocean-holiday-316301/>

http://en.wikipedia.org/wiki/Christiaan_Huygens#mediaviewer/File:Christiaan_Huygens.jpg

4. Conditionals & Loops

- Conditionals: the if statement
- Loops: the while statement
- An alternative: the for loop
- Nesting
- Debugging

4. Conditionals and Loops



<https://introcs.cs.rutgers.edu>