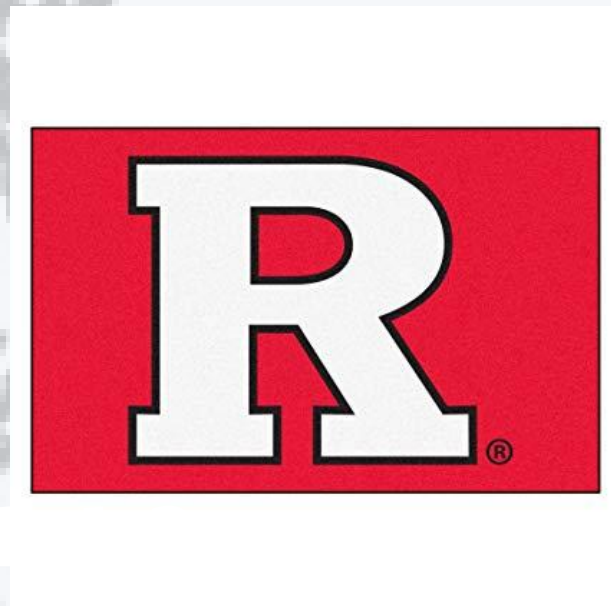


4. Conditionals and Loops



- 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?"
- **FROM TUESDAY**
 - Textbook readings????
 - Why exactly does a double of $(3/5)$ result in 0?
 - Math.random() review
 - DigitsSum review

4. Conditionals & Loops

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

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



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) {
```

operations

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

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

n operations

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

n operations

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

```
    }
```

```
    else{
```

n SOP operations

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

```
    }
```

```
    count++;
```

n operations

```
}
```

Total operations = $5(n) + 3 = 5n + 3$

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:

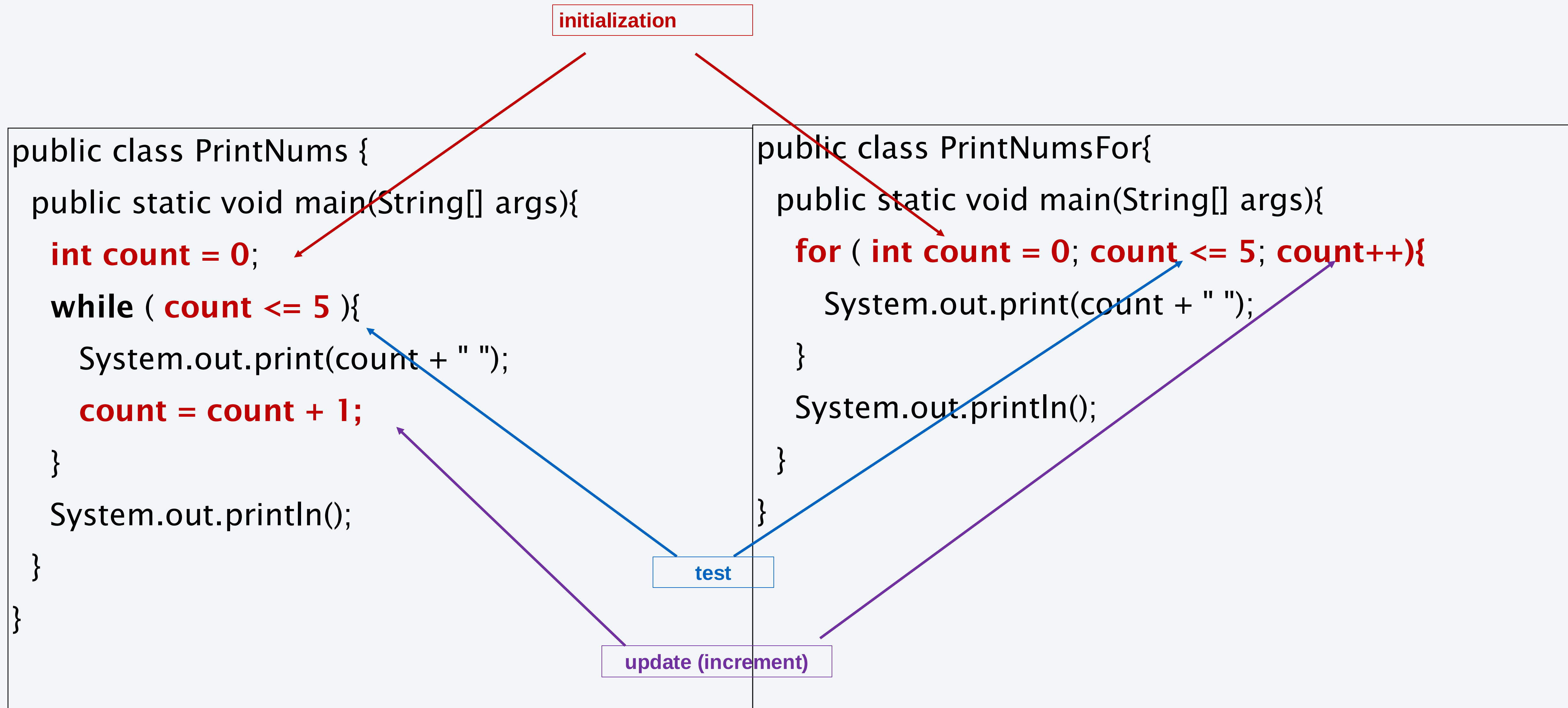
```
int v = 1;
for (int i = 0; i <= n; i++)
{
    System.out.println( i + " " + v );
    v = 2*v;
}
```

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

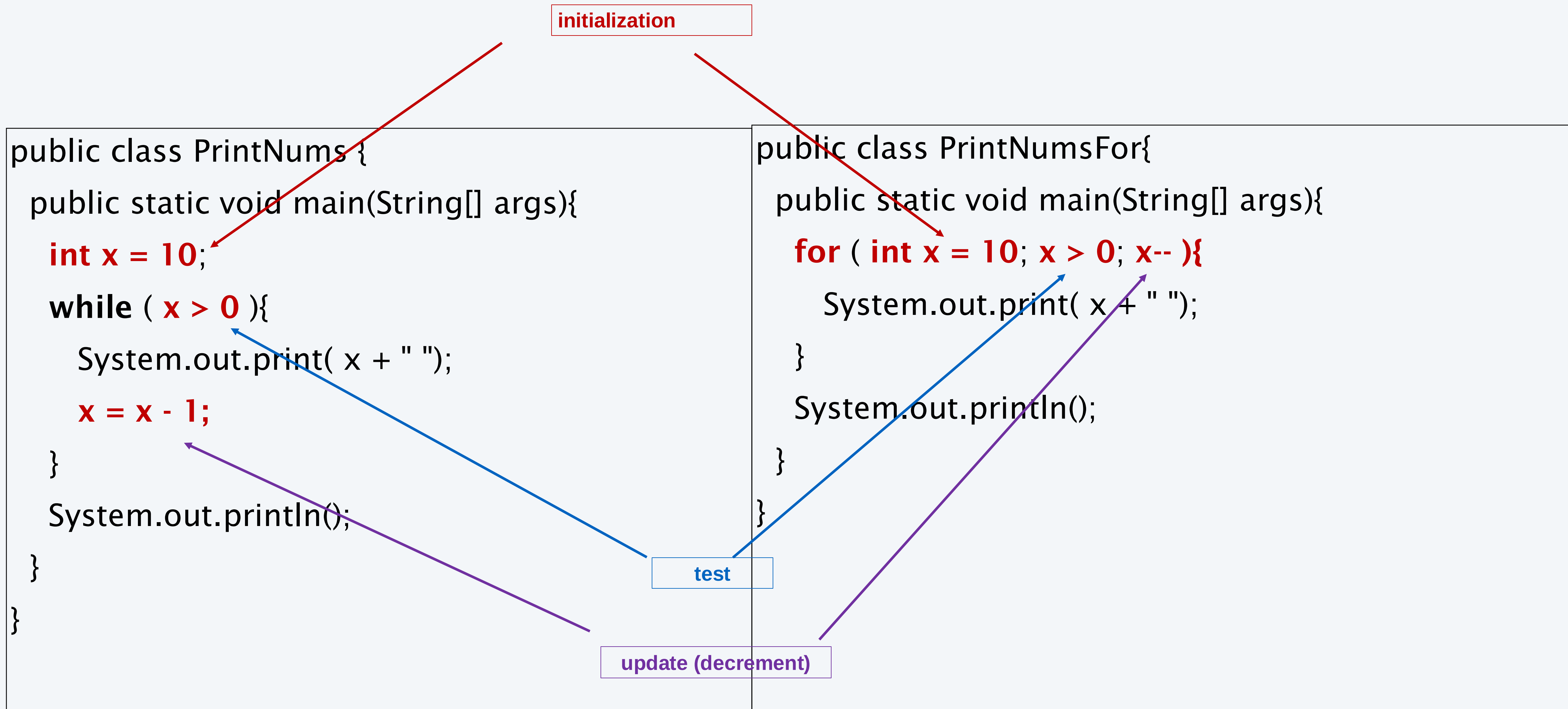
Every **for** loop has an equivalent **while** loop:

```
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)



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=y3rfunryi4ilszevy5ieombwbdjm6rv5>

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%

```
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;
```

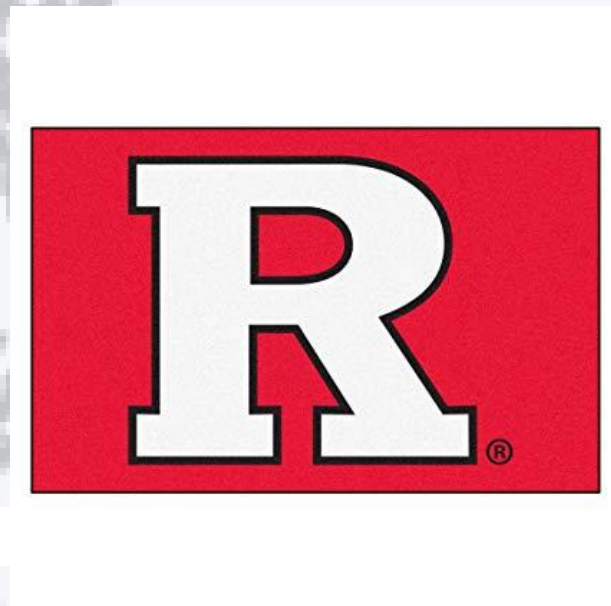
Note. Braces are not needed in this case but BE CAREFUL when nesting if-else statements because of potential ambiguity.

else ifs – Taxes2.java

4. Conditionals & Loops TO BE CONTINUED...

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

4. Conditionals and Loops



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