CMSC 150
INTRODUCTION TO COMPUTING

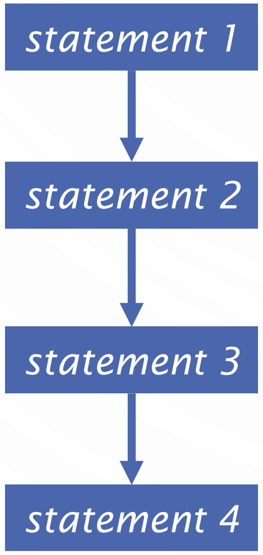
ACKNOWLEDGEMENT: THESE SLIDES ARE ADAPTED FROM SLIDES PROVIDED WITH INTRODUCTION TO PROGRAMMING IN JAVA: AN INTERDISCIPLINARY APPROACH, SEDGEWICK AND WAYNE (PEARSON ADDISON-WESLEY 2007)

LECTURE 3

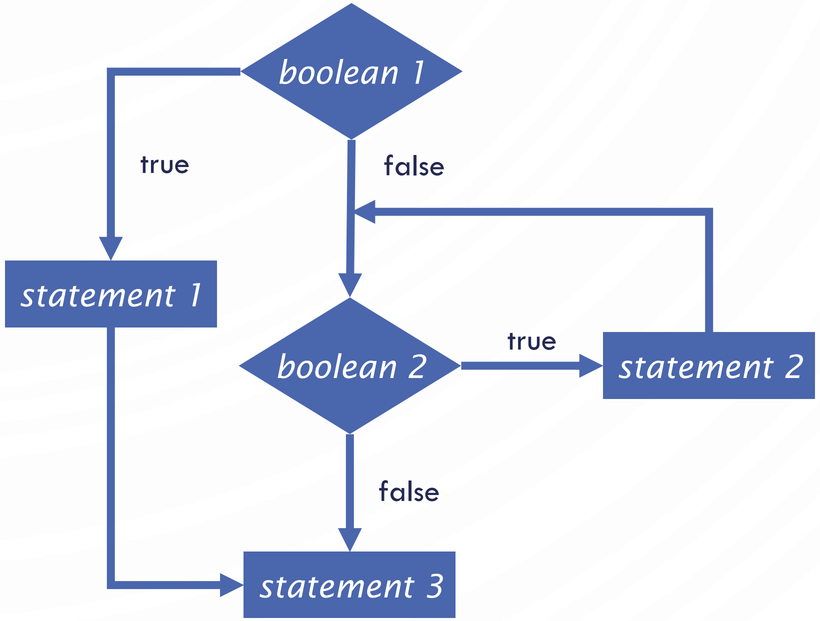
- IF, WHILE, FOR
- SCOPE
- NESTING
- OTHER CONTROL FLOW STATEMENTS

CONTROL FLOW

- Control flow.
  - Sequence of statements that are actually executed in a program.
  - Conditionals and loops: enable us to choreograph control flow.



straight-line control flow



control flow with conditionals and loops

CONDITIONALS

LETS SAY YOU WANT TO BE A POLL WORKER FOR A CAUCUS OR PRIMARY

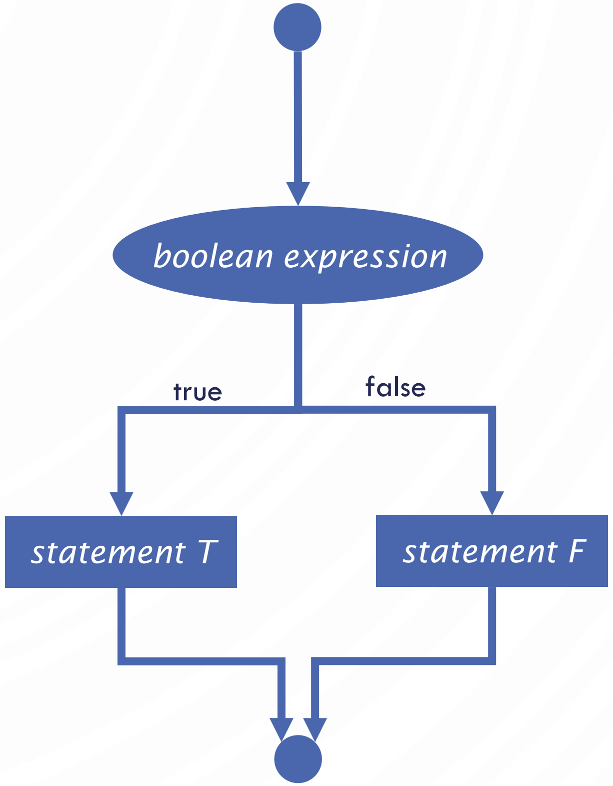
- You have to sort people by their political party
- If a person is republican they take one ballot, otherwise they are democrat and have a different ballot
- How could we automate this?

IF STATEMENT

- if statement. A common branching structure.
  - Evaluate a boolean expression.
  - If true, execute some statements.
  - If false, execute other statements.

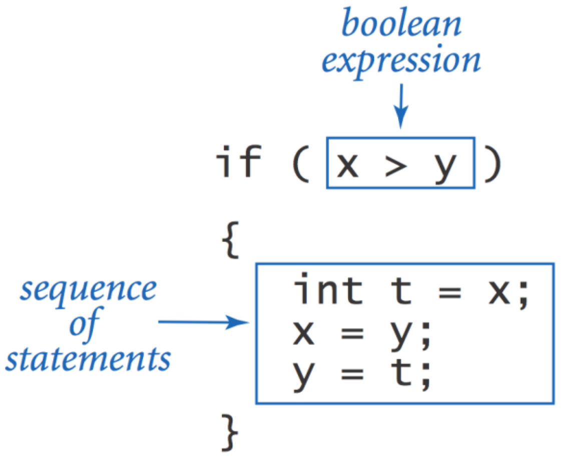
```
if (boolean expression) {
  statement T;
}
else {
  statement F;
}
```

can be any sequence of statements



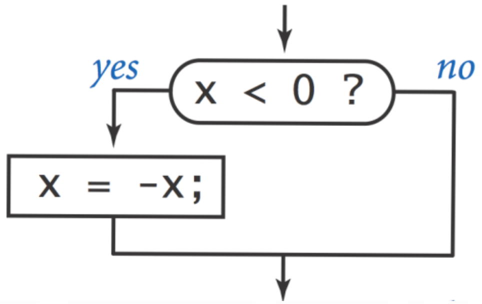
IF STATEMENT

- Example of control flow with if

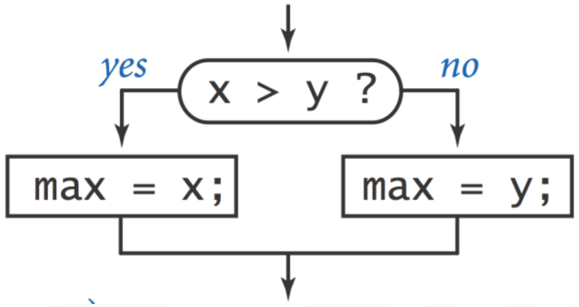


```
if ( x > y )
{
   int t = x;
   x = y;
   y = t;
}
```

```
if (x < 0) x = -x;
```



```
if (x > y) max = x;
else       max = y;
```



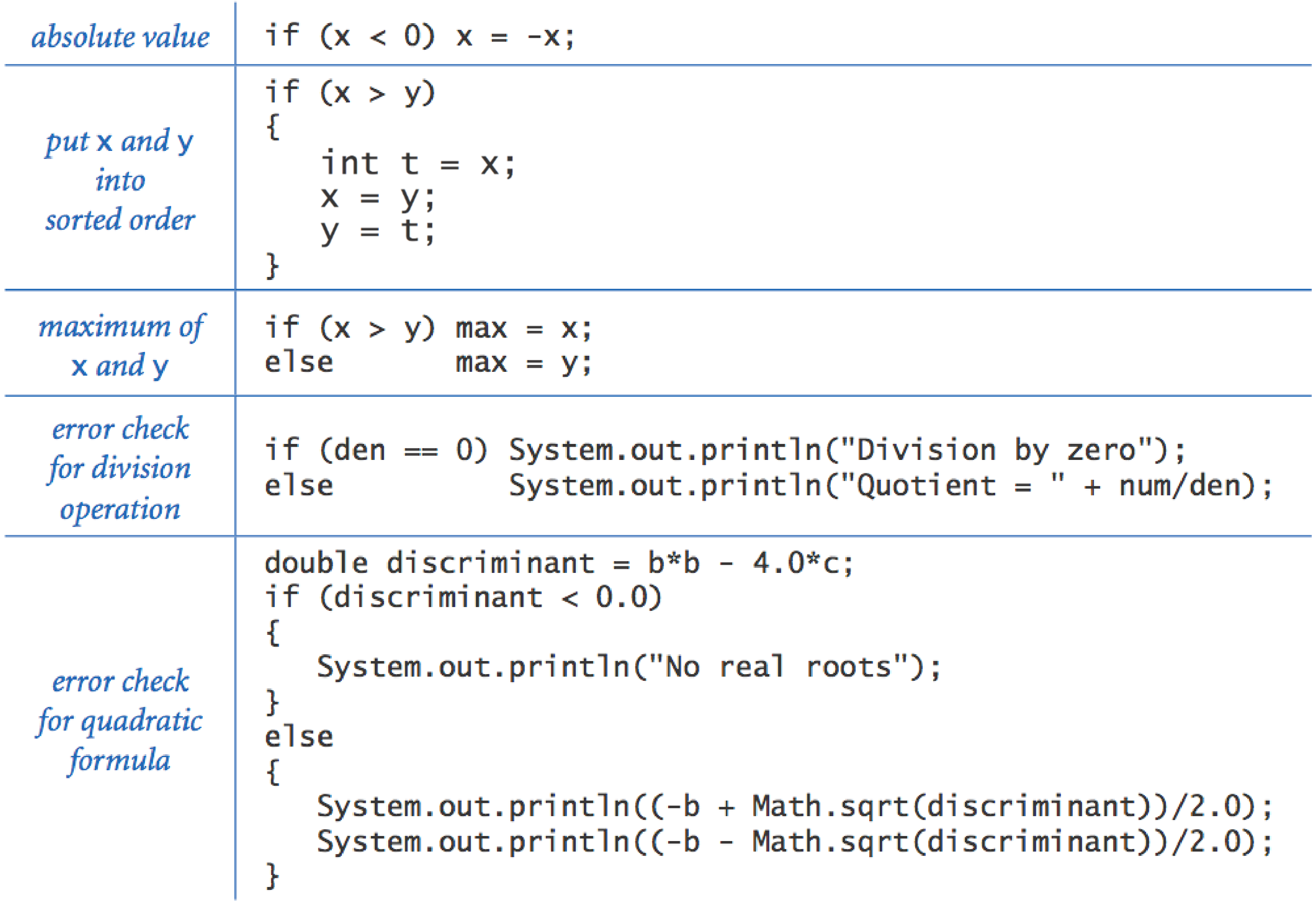
IF STATEMENT

- Ex. Take different action depending on value of variable.

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

```
% java Flip
Heads

% java Flip
Heads

% java Flip
Tails

% java Flip
Heads
```

IF STATEMENT EXAMPLES



absolute value	`if (x < 0) x = -x;`

put x *and* y *into sorted order*	`if (x > y)`
	`{`
	`   int t = x;`
	`   x = y;`
	`   y = t;`
	`}`

maximum of x *and* y	`if (x > y) max = x;`
	`else       max = y;`

error check for division operation	`if (den == 0) System.out.println("Division by zero");`
	`else          System.out.println("Quotient = " + num/den);`

error check for quadratic formula	`double discriminant = b*b - 4.0*c;`
	`if (discriminant < 0.0)`
	`{`
	`   System.out.println("No real roots");`
	`}`
	`else`
	`{`
	`   System.out.println((-b + Math.sqrt(discriminant))/2.0);`
	`   System.out.println((-b - Math.sqrt(discriminant))/2.0);`
	`}`

ACTIVITY – WITH A PARTNER

- Write an algorithm using if and else statements to output three numbers a, b, c in sorted order. You don’t have to write valid Java. This is just called pseudocode, i.e., code-like statements
  - Example pseudocode vs Java

    ```
    Output a
    ```

    vs

    ```
    System.out.println(a);
    ```

  - Or

    $$a \leftarrow 0$$

    vs

    ```
    int a = 0;
    ```

ELSE IF STATEMENTS

- Can allow more than two options with else-if statement
- Ex. Pay a certain tax rate depending on income level.

Income	Rate
0 – 47, 450	22%
47,450 – 114,650	25%
114,650 – 174,700	28%
174,700 – 311,950	33%
311,950 – ∞	35%

5 mutually exclusive alternatives…

```
double rate;
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 IF STATEMENTS

- Why didn't we use this program?

```
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;
```

5 mutually exclusive alternatives…

Income	Rate
0 – 47, 450	22%
47,450 – 114,650	25%
114,650 – 174,700	28%
174,700 – 311,950	33%
311,950 – ∞	35%

ACTIVITY

- Could we rework our algorithm to sort 3 numbers with else-if statements to make it more clear?

EXERCISE – WITH A PARTNER

- Write a program that takes three integer command-line arguments and prints equal if all three are equal, and not equal otherwise
- Add statements to your first program which ensure three and only three arguments were given to the program. Output a good error message so that "exception:ArrayIndexOutOfBounds" doesn't occur and you know what went wrong in your program. Hint: Use args.length to see how many arguments there are.
- Fix this java excerpt

```
if(x = b && x != a)
    DoSomething();
```

THE WHILE LOOP

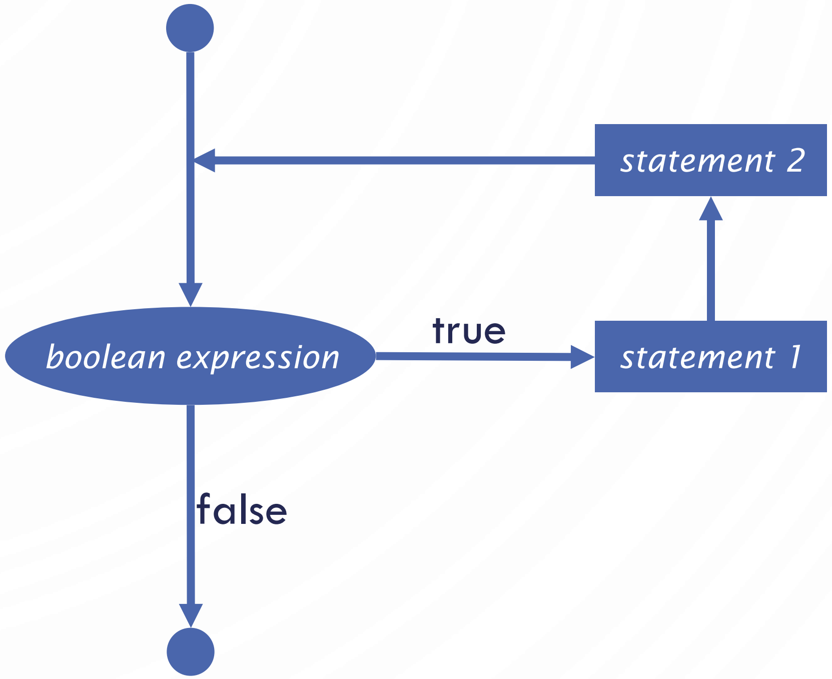
WHILE LOOP

- while loop. A common repetition structure.
  - Evaluate a boolean expression.
  - If true, execute some statements.
  - Repeat.

```
while (boolean expression) {
   statement 1;
   statement 2;
}
```

loop continuation condition

loop body



WHILE LOOP: POWERS OF TWO

- Ex. Print powers of 2 that are $\leq 2^N$.
  - Increment i from 0 to N.
  - Double v each time.

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

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

$\boldsymbol{N = 6}$

```
0 1
1 2
2 4
3 8
4 16
5 32
6 64
```

ACTIVITY – WHILE LOOP

- What is wrong with the following code?
- What happens?
- Fix it and explain what the code outputs

```
int i = 0;
while (i <= N)
    System.out.println(i);
    i = i + 5;
```

ACTIVITY – WHILE LOOP

- Write an algorithm (in pseudocode) to compute the number of digits an integer has.
  - Example: input – 34567 output – 5
- Bonus: modify your algorithm to compute the number of “digits” for any base, e.g., binary, octal, or hexadecimal

EXAMPLE: IMPLEMENTING MATH.SQRT()

- Newton-Raphson method to compute $\sqrt{c}$:
  - Initialize $t_0 = c$
  - Repeat-until $t_i = \mathrm{c/t_i}$, up to desired precision:

    set t_{i+1} to be the average of t_i and c/t_i

$$
\begin{array}{lcccc}
t_0 & & & = & 2.0 \\
t_1 & = & \frac{1}{2}(t_0 + \frac{2}{t_0}) & = & 1.5 \\
t_2 & = & \frac{1}{2}(t_1 + \frac{2}{t_1}) & = & 1.4166666666666665 \\
t_3 & = & \frac{1}{2}(t_2 + \frac{2}{t_2}) & = & 1.4142156862745097 \\
t_4 & = & \frac{1}{2}(t_3 + \frac{2}{t_3}) & = & 1.4142135623746899 \\
t_5 & = & \frac{1}{2}(t_4 + \frac{2}{t_4}) & = & 1.414213562373095
\end{array}
$$

computing the square root of 2

EXAMPLE: IMPLEMENTING MATH.SQRT()

- Newton-Raphson method to compute $\sqrt{c}$:
  - Initialize $t_0 = c$
  - Repeat-until $t_i = \mathrm{c}/\mathrm{t_i}$, up to desired precision: set t_{i+1} to be the average of t_i and c/t_i

```java
public class Sqrt {
    public static void main(String[] args) {
        double epsilon = 1e-15;
        double c = Double.parseDouble(args[0]);
        double t = c;
        while (Math.abs(t - c/t) > t*epsilon) {
            t = (c/t + t) / 2.0;
        }
        System.out.println(t);
    }
}
```

```
% java Sqrt 2.0
1.414213562373095
```

ACTIVITY – WHILE LOOP

- Reverse guessing game – Write a program which takes as input N and a number g. Generate random numbers in the range $[1, N]$ until g is generated. Output the number of guesses the computer took.
- Bonus
  - Protect your program input with if statements.
  - Allow the computer to repeat the guessing process for g 10 times. Average the number of guesses taken.

QUESTION DAY

- This is your chance to ask about all things java. Consider it a review and clarification time! I will explain anything you want to the best of my ability.

EXAMPLES – WITH A PARTNER

- What does are the values of n and m after this:

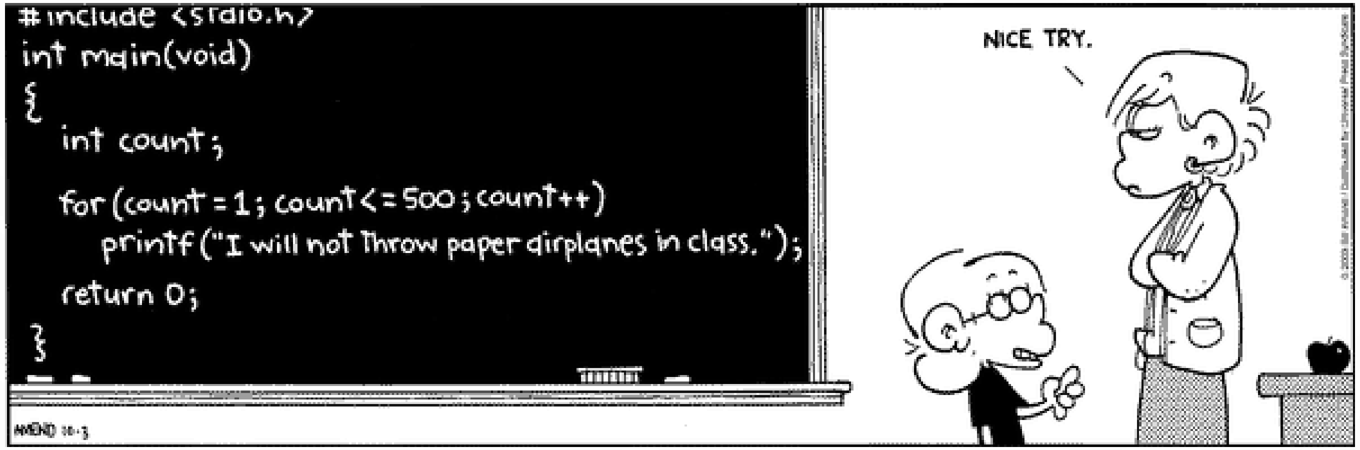
```
int n = 1234567;
int m = 0;
while(n != 0) {
    m = (10*m) + (n % 10);
    n /= 10;
}
```

- Show the trace of the program at each step

EXERCISE – WITH A PARTNER

- Random walk. You begin standing at the center of a disk of radius r. At each time-step you pick a random direction in with respect to the x-axis and take a step of 1 meter. How many steps did it take you to fall off?
    - Start at $(x, y) = (0, 0)$; *YES DECIMAL PLACES ARE ALLOWED*
    - Randomly generate theta $\theta \in [0,2\pi)$
    - Then your new position $(x, y) = (x + \cos(\theta)\,, y + \sin(\theta))$
    - Bonus: Bias the random walk so that the random direction isn't 100% random.
    - Bonus: Lets say after falling off your disk you fall on another disk, for N disks. Each time you fall, you land at a random position $\left(r * cos(\theta), r * sin(\theta)\right)$ where $\theta \in [0, 2\pi)$ and begin again. How many steps did it take?
- Start by planning you algorithm. Then implement it.
- This question has applications to simulating cellular and molecular systems.

THE FOR LOOP



```
#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;
}
```

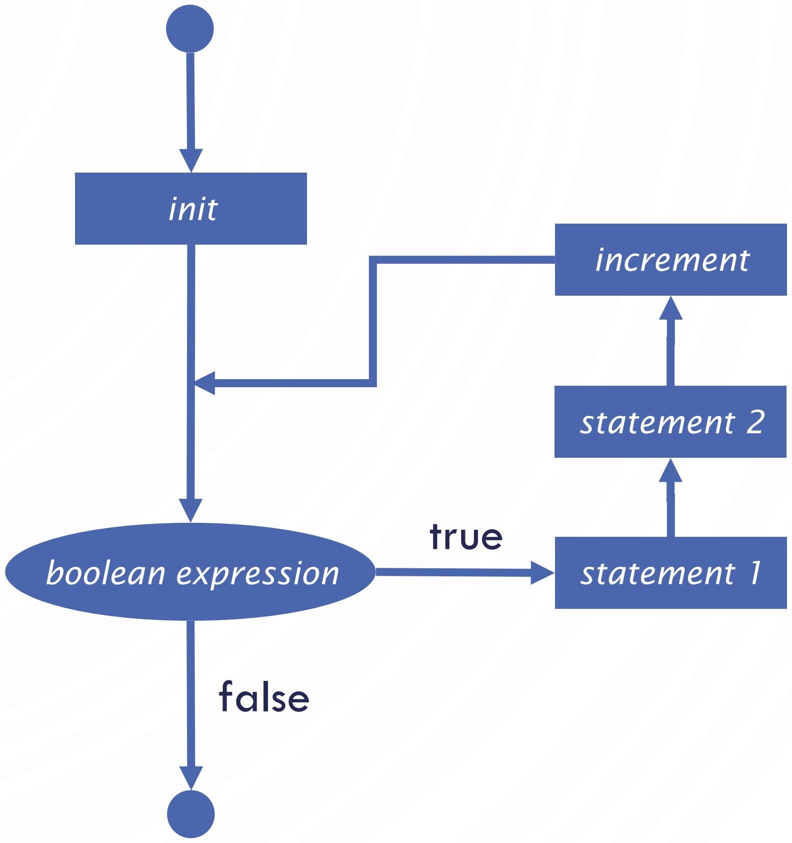
FOR LOOPS

- for loop. Another common repetition structure.
    - Execute initialization statement.
    - Evaluate a boolean expression.
    - If true, execute some statements.
    - And then the increment statement.
    - Repeat.

```
for (init; boolean expression; increment) {
   statement 1;
   statement 2;
}
```

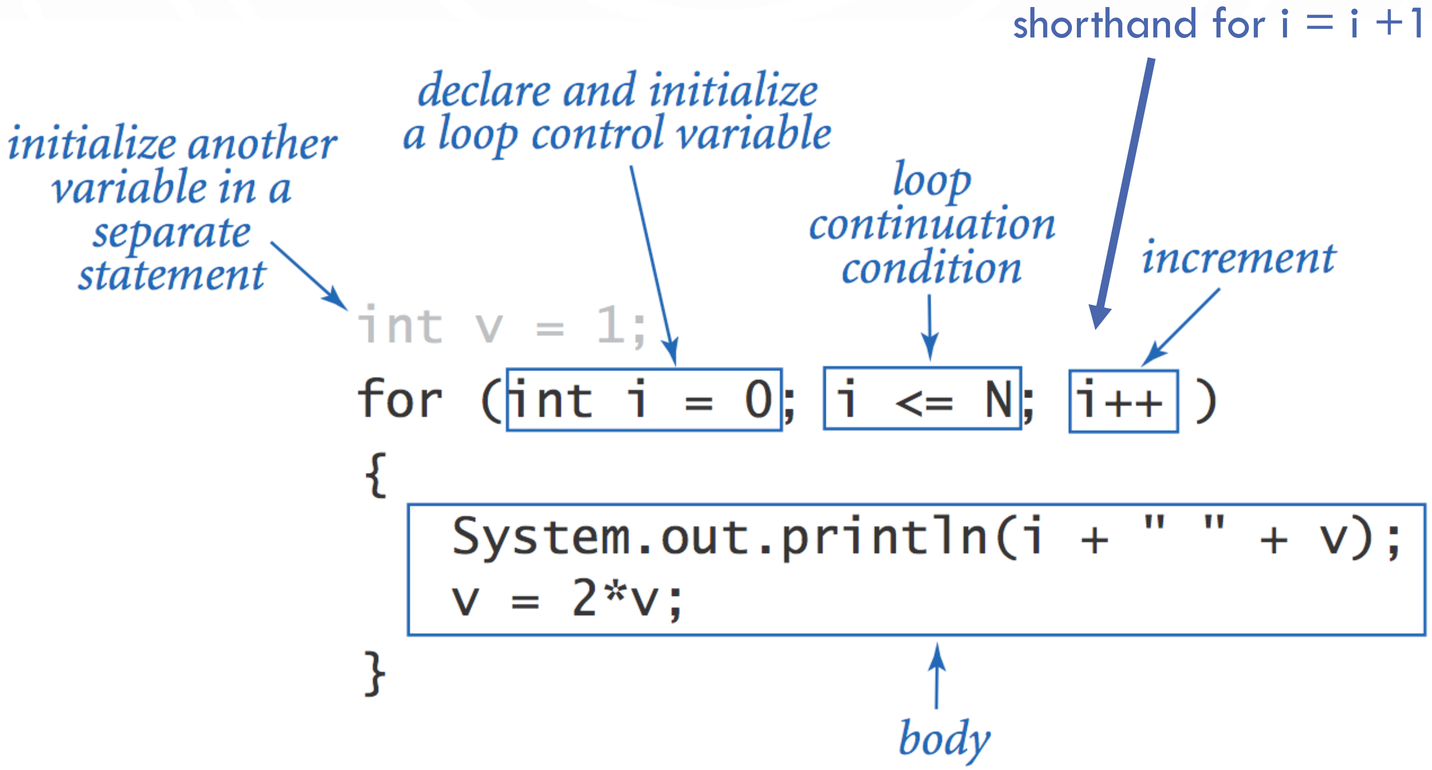
loop continuation condition

body



ANATOMY OF A FOR LOOP

- Q. What does it print?



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

shorthand for i = i +1

declare and initialize a loop control variable

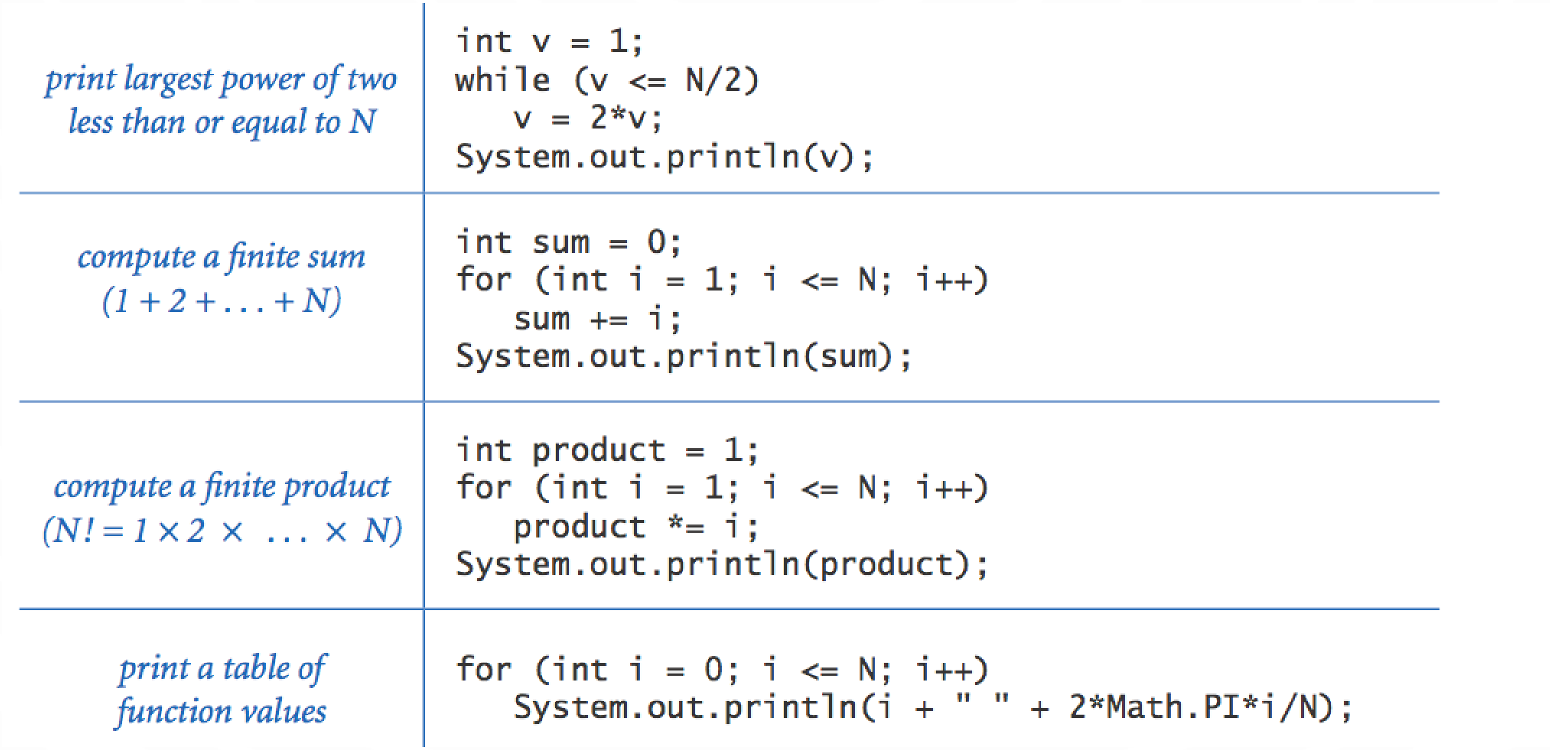
initialize another variable in a separate statement

loop continuation condition

increment

body

LOOP EXAMPLES



print largest power of two less than or equal to N	int v = 1;
	while (v <= N/2)
	   v = 2*v;
	System.out.println(v);

compute a finite sum (1 + 2 + . . . + N)	int sum = 0;
	for (int i = 1; i <= N; i++)
	   sum += i;
	System.out.println(sum);

compute a finite product (N! = 1 × 2 × . . . × N)	int product = 1;
	for (int i = 1; i <= N; i++)
	   product *= i;
	System.out.println(product);

print a table of function values	for (int i = 0; i <= N; i++)
	   System.out.println(i + " " + 2*Math.PI*i/N);

PRACTICE

- Table 1: Write a for loop to output all numbers between integers a and b
- Table 2: Write a for loop to output all command line arguments. Recall: args.length gives the number of command line arguments
- Table 3: Write a for loop to output the multiples of an integer a up to N
- Table 4: Write a for loop to output all the even numbers from 100 to 999 in reverse order.

NESTING

NESTING

- In control flow, nesting is where you place a control structure inside of another
- Example: 2 for loops to print a multiplication table

```
for(int i = 0; i < 10; ++i) {
    for(int j = 0; j < 10; ++j)
        System.out.printf(“%d*%d = %2d\t”, i, j, i*j);
    System.out.println();
}
```

NESTED IF STATEMENTS

- Use nested if statements to handle multiple alternatives.

```
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;
      }
   }
}
```

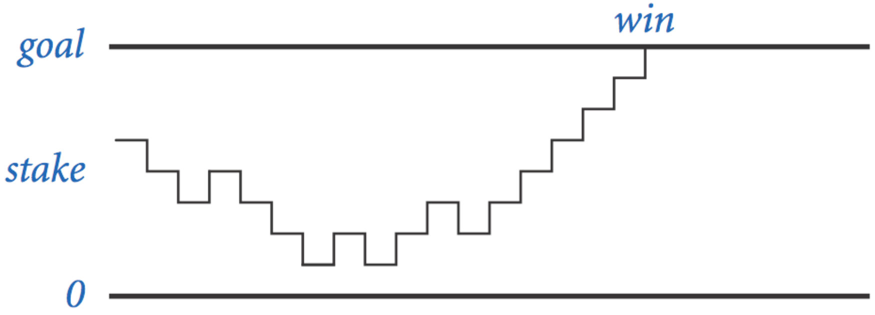
- Or use the shorthand:

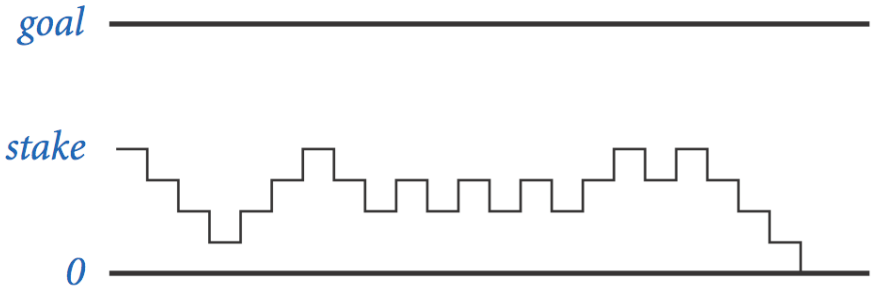
```
if      (income <  47450) rate = 0.22;
else if (income < 114650) rate = 0.25;
else if (income < 174700) rate = 0.28;
else if (income < 311950) rate = 0.33;
else                      rate = 0.35;
```

MONTE CARLO SIMULATION

GAMBLER'S RUIN

- Gambler's ruin. Gambler starts with $stake and places $1 fair bets until going broke or reaching $goal.
  - What are the chances of winning?
  - How many bets will it take?
- One approach. Monte Carlo simulation.
  - Flip digital coins and see what happens.
  - Repeat and compute statistics.





GAMBLER'S RUIN

```java
public class Gambler {
   public static void main(String[] args) {
      int stake = Integer.parseInt(args[0]), goal = Integer.parseInt(args[1]); T = Integer.parseInt(args[2]);
      int wins = 0;
      // repeat experiment T times
      for (int t = 0; t < T; t++) {
         // do one gambler's ruin experiment
         int cash = stake;
         while (cash > 0 && cash < goal) {
            // flip coin and update
            if (Math.random() < 0.5) cash++;
            else                     cash--;
         }
         if (cash == goal) wins++;
      }
      System.out.println(wins + " wins of " + T);
   }
}
```

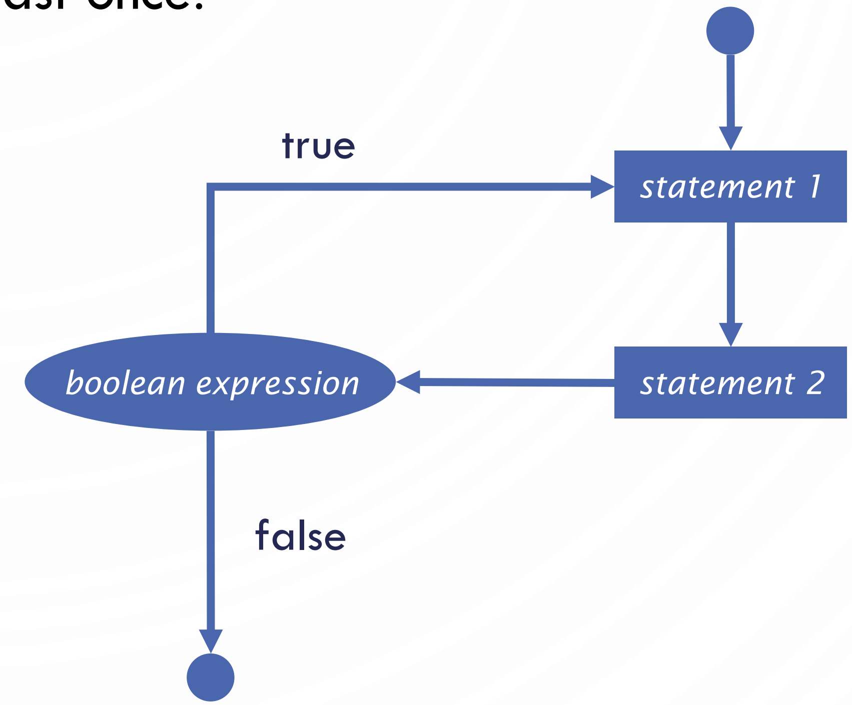
```
% java Gambler 5 25 1000
191 wins of 1000

% java Gambler 5 25 1000
203 wins of 1000

% java Gambler 500 2500 1000
197 wins of 1000
```

OTHER CONTROL FLOW STATEMENTS

DO-WHILE LOOP

- do-while loop. Guaranteed to execute at least once!
  - Execute sequence of statements.
  - Check loop-continuation condition.
  - Repeat.

```
do {
  statement 1;
  statement 2;
} while (boolean expression);
```



EXAMPLE: DO-WHILE

- Average a set of numbers

```
Scanner s = new Scanner(System.in);
double sum = 0, number = 0;
do {
    System.out.print(“Enter a number (0 to quit): ”);
    number = s.nextDouble();
    sum += number;
} while(number != 0);
System.out.println(“Sum: ” + sum);
```

COMPARISON OF LOOPS

- for loop – used when you know how many times to execute or each iteration has a natural increment
- while loop – used to execute 0 or more times. Pre-condition check.
- do-while loop – used to execute 1 or more time. Post-condition check.

OTHER HELPFUL STATEMENTS FOR LOOPS

- **break** – immediately exit the loop. Do not continue executing any more of the loop:

```
while(true) {
    if(q-key-is-pressed()) //quit the game
        break;
    Game-loop();
}
```

- **continue** – immediately skip to the end of the body of the loop, i.e., start next iteration (checking the condition):

```
for(int i = 0; i < 10; ++i) {
    if(isPrime(i)) //OCD against prime numbers
        continue;
    HandleNotPrimes();
}
```

MULTIPLE CONDITIONS WITH SWITCH

- Switch statement. Allows multiple alternatives just like with if-else.
    - Expression must be of type **char**, **byte**, **int**, String, etc. But no floating point values!
    - **default** is like **else**
    - **break** exits switch block

```
switch(expression) {
    case firstValue: statements; break;
    case secondValue: statements; break;
    default: statements;
}
```

Example

```
char keyPressed;
switch(keyPressed) {
    case ‘w’: MoveUp(); break;
    case ‘a’: MoveLeft(); break;
    case ‘s’: MoveDown(); break;
    case ‘d’: MoveRight(); break;
    default: StandStill();
}
```

CONTROL FLOW SUMMARY

- Control flow.
  - Sequence of statements that are actually executed in a program.
  - Conditionals and loops: enable us to choreograph the control flow.

Control Flow	Description	Examples
Straight-line programs	All statements are executed in the order given	
Conditionals	Certain statements are executed depending on the values of certain variables	**if; if-else; switch**
Loops	Certain statements are executed repeatedly until certain conditions are met	**while; for; do-while**