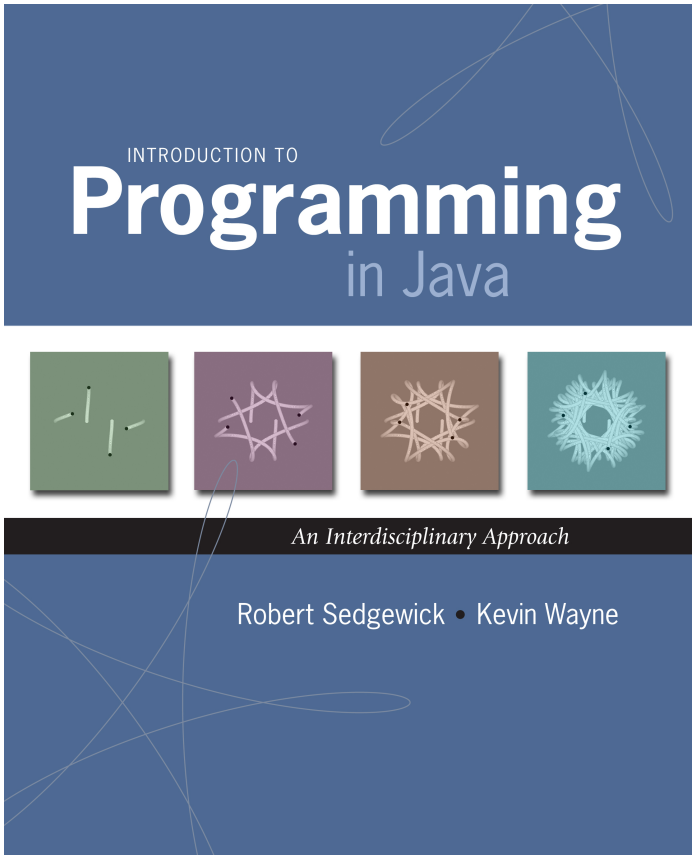
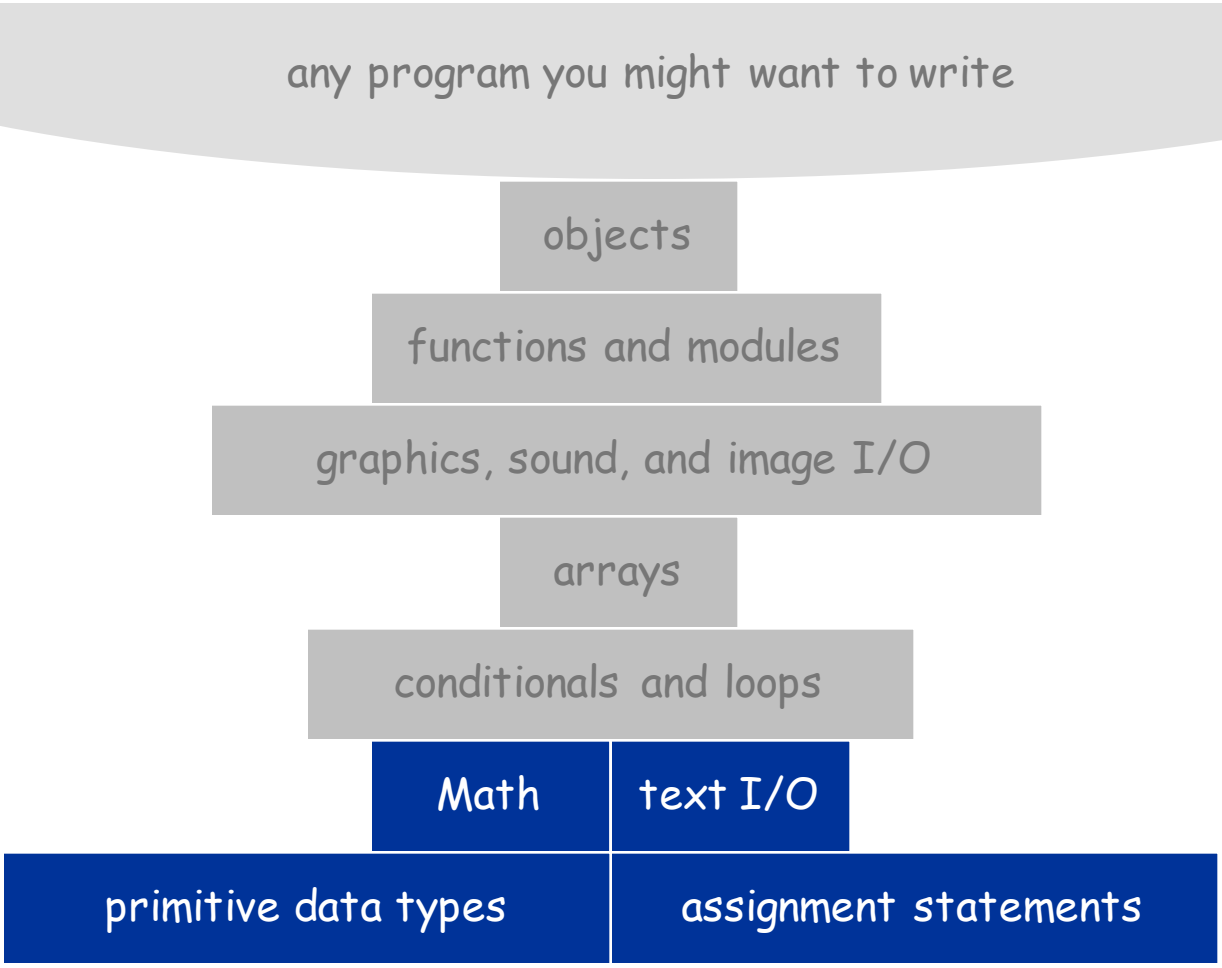
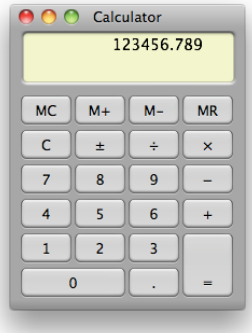
1.3 Conditionals and Loops



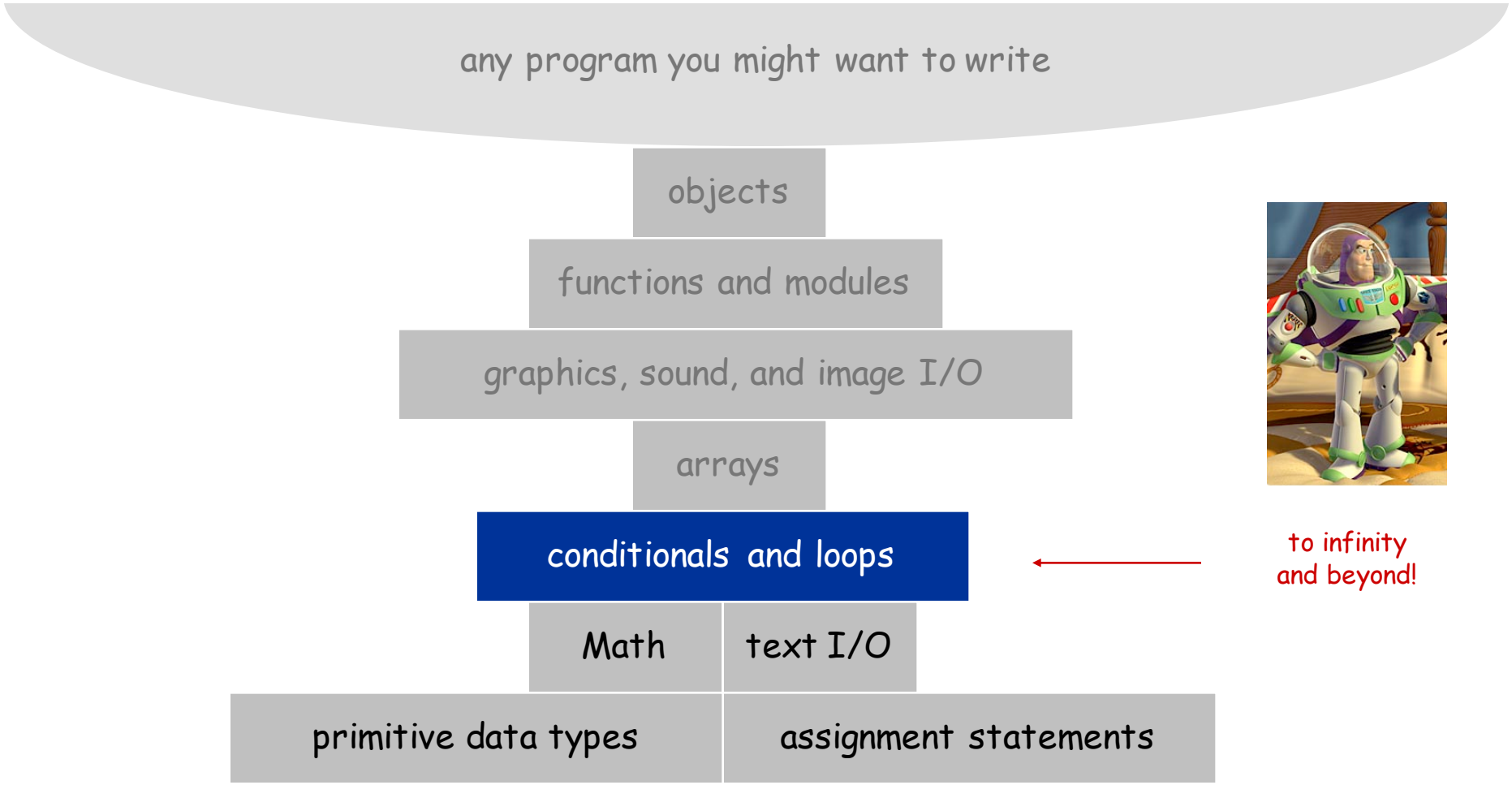
A Foundation for Programming





last lecture: equivalent to a calculator

A Foundation for Programming

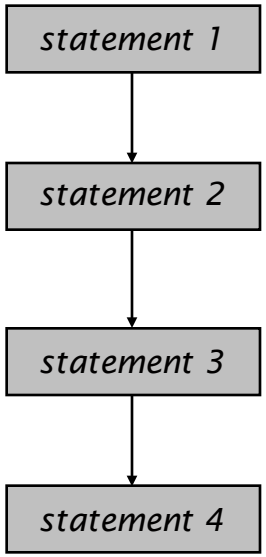


to infinity and beyond!

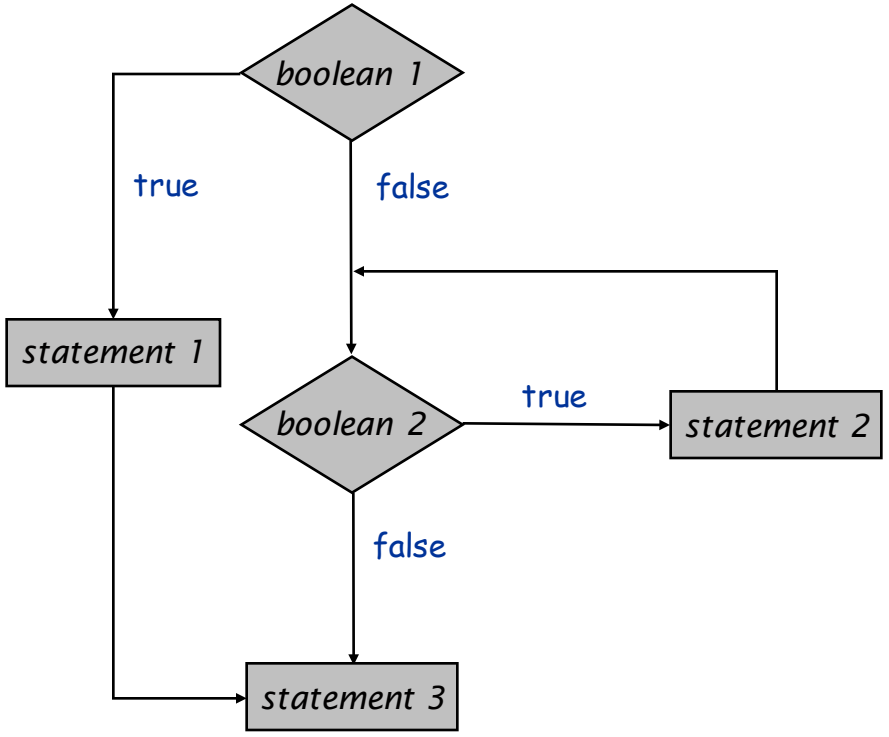
Control Flow

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

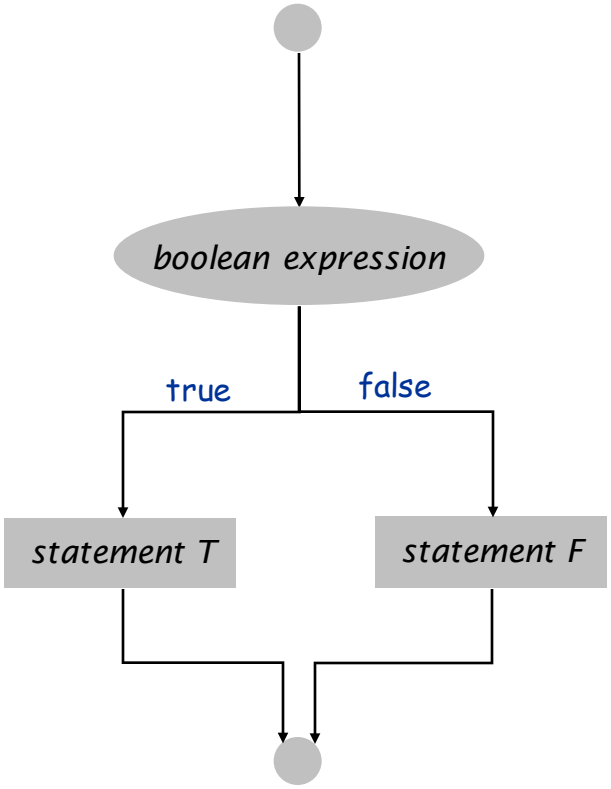
If Statement

The `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

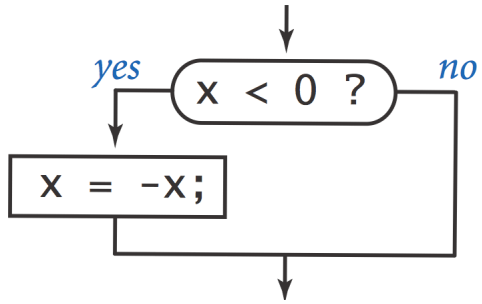


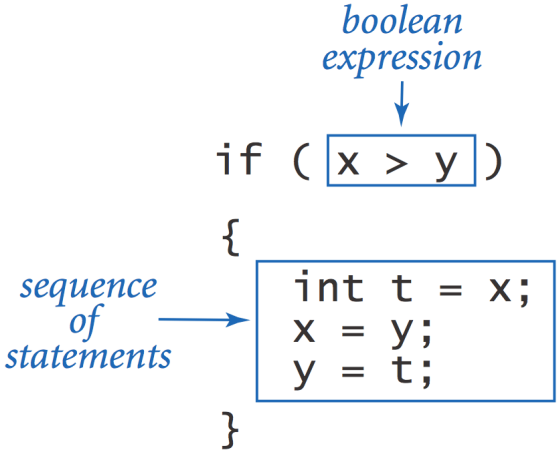
If Statement

The `if` statement. A common branching structure.

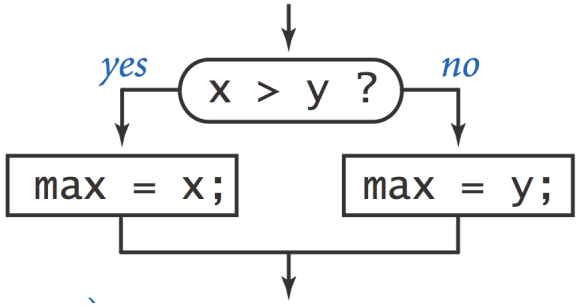
- Evaluate a `boolean` expression.
- If `true`, execute some statements.
- If `false`, execute other statements.

```
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);`

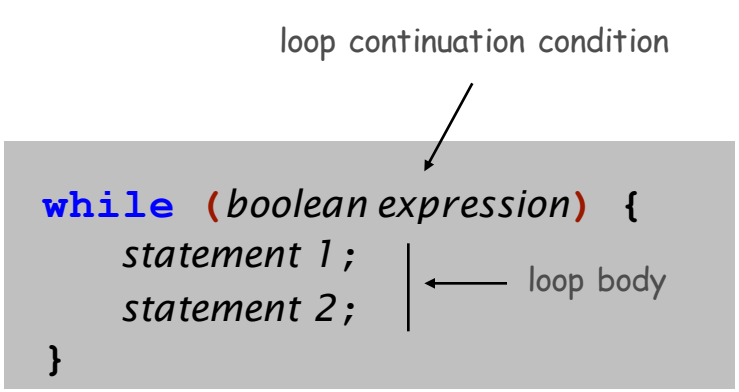
error check for quadratic formula	`double discriminant = b*b - 4.0*c;`
	`if (discriminant < 0.0)`
	`{`
	`  System.out.println("No real roots");`
	`}`
	`else`
	`{`
	`  System.out.println((-b + Math.sqrt(discriminant))/2.0);`
	`  System.out.println((-b - Math.sqrt(discriminant))/2.0);`
	`}`

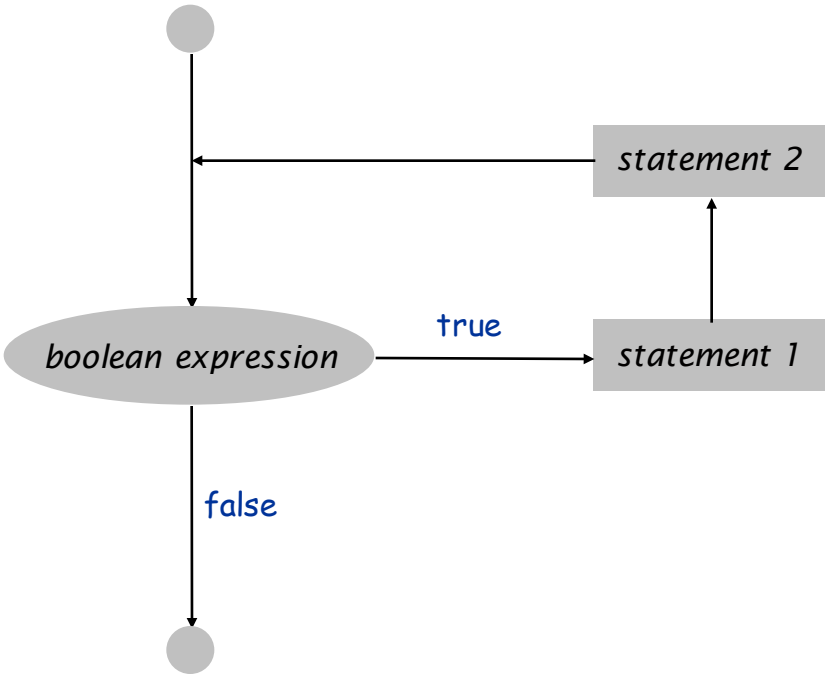
The While Loop

While Loop

The `while` loop. A common repetition structure.

- Evaluate a `boolean` expression.
- If `true`, execute some statements.
- Repeat.





While Loop: Powers of Two

Ex. Print powers of 2 that are ≤ 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

N = 6

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

Powers of Two

```
public class PowersOfTwo {
   public static void main(String[] args) {

      // last power of two to print
      int N = Integer.parseInt(args[0]);

      int i = 0;  // loop control counter
      int v = 1;  // current power of two
      while (i <= N) {
         System.out.println(i + " " + v);
         i = i + 1;
         v = 2 * v;
      }
   }
}
```

print i and ith power of two

```
% java PowersOfTwo 3
0 1
1 2
2 4
3 8

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

While Loop Challenge

Q. Anything wrong with the following code for printing powers of 2?

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

While Loop Challenge

Q. Anything wrong with the following code for printing powers of 2?

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

A. Need curly braces around statements in while loop; otherwise it enters an infinite loop, printing "0 1".

Moment of panic. How to stop infinite loop?

While Loops: Square Root

Goal. Implement `Math.sqrt()`.

```
% java Sqrt 2.0
1.414213562373095
```

15 decimal digits of accuracy in 5 iterations

Newton-Raphson method to compute the square root of c:

- Initialize $t_0 = c$.
- Repeat until $t_i = c / t_i$, up to desired precision: set t_{i+1} to be the average of t_i and c / t_i.

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

computing the square root of 2

"A wonderful square root. Let's hope it can be used for the good of mankind."

While Loops: Square Root

Goal. Implement `Math.sqrt()`.

```
% java Sqrt 2.0
1.414213562373095
```

15 decimal digits of accuracy in 5 iterations

Newton-Raphson method to compute the square root of c:

- Initialize $t_0 = c$.
- Repeat until $t_i = c / t_i$, up to desired precision: set t_{i+1} to be the average of t_i and c / t_i.

```
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);
   }
}
```

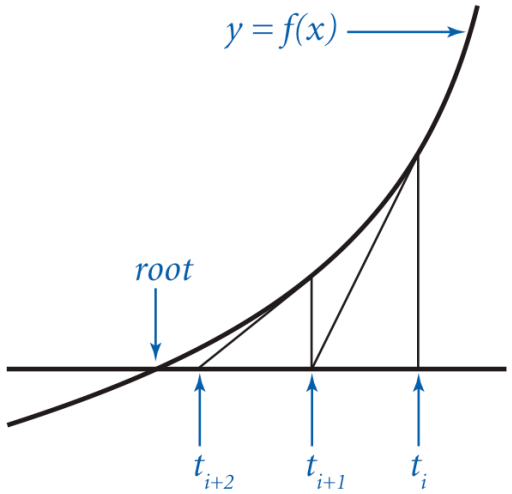
relative error tolerance

Newton-Raphson Method

Square root method explained.

- Goal: find root of any function f(x).
- Start with estimate t_0.
- Draw line tangent to curve at $x = t_i$.
- Set t_{i+1} to be x-coordinate where line hits x-axis.
- Repeat until desired precision.

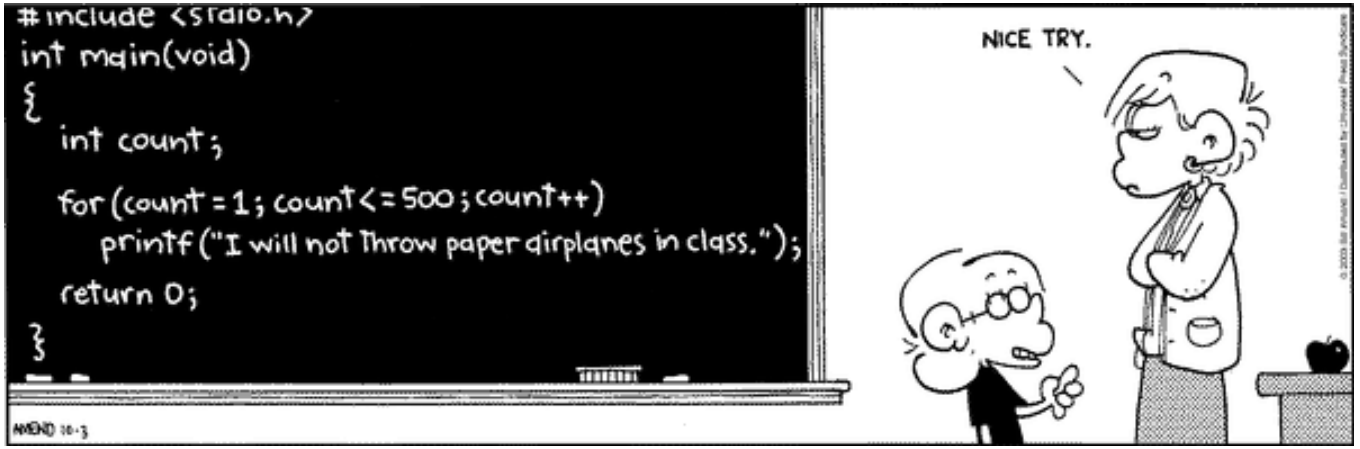
f(x) = x^2 - c to compute $\sqrt{c}$



$$t_{i+1} = t_i - \frac{f(t_i)}{f'(t_i)}$$

Technical conditions. f(x) is smooth; t_0 is good estimate.

The For Loop



www.ucomics.com/foxtrot/2003/10/03

For Loops

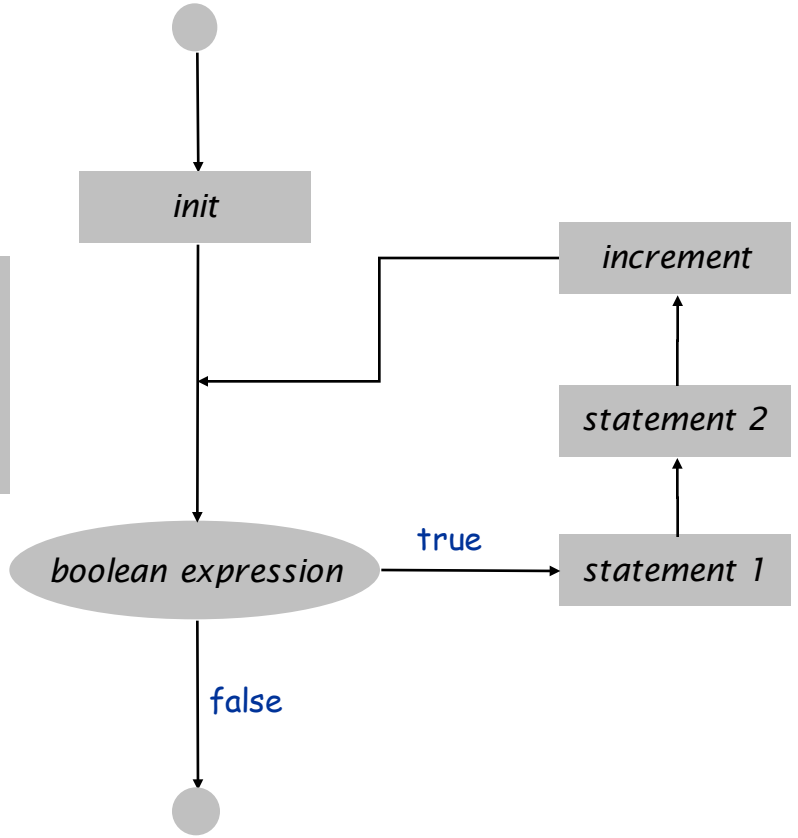
The `for` loop. Another common repetition structure.

- Execute initialization statement.
- Evaluate a `boolean` expression.
- If `true`, execute some statements.
- And then the increment statement.
- Repeat.

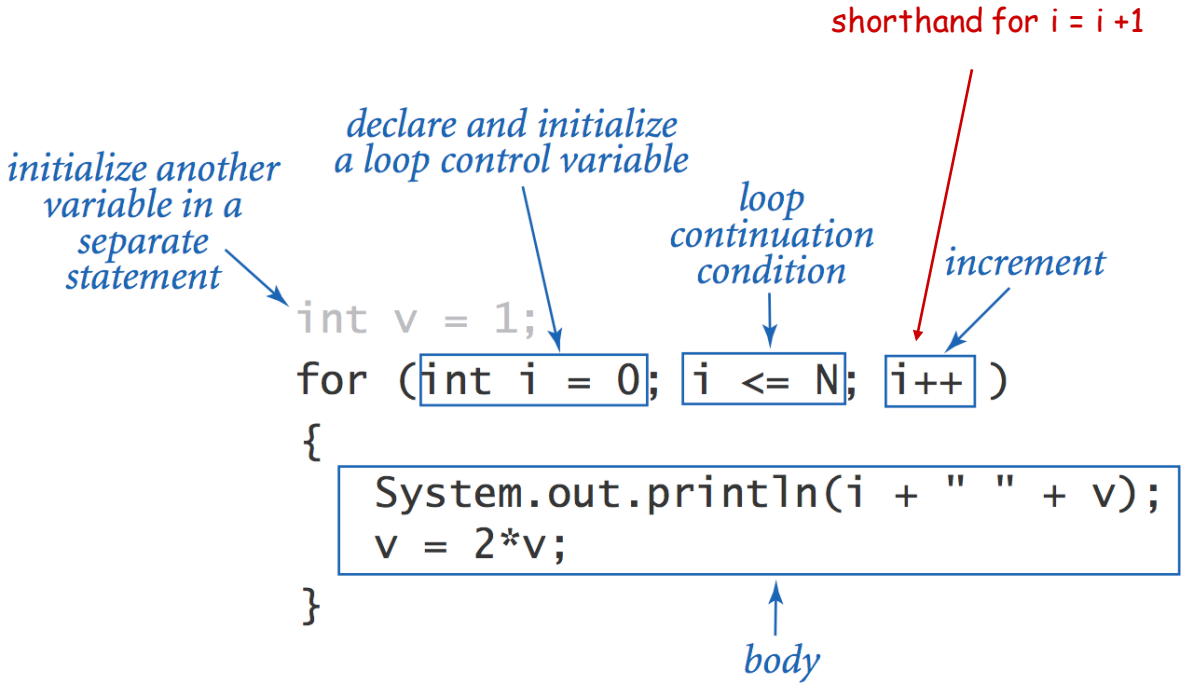
loop continuation condition

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

body



Anatomy of a For Loop



Q. What does it print?

A.

For Loops: Subdivisions of a Ruler

Create subdivision of a ruler.

- Initialize `ruler` to `" "`.
- For each value `i` from `1` to `N`:
  sandwich two copies of `ruler` on either side of `i`.

```
public class RulerN {
   public static void main(String[] args) {
      int N = Integer.parseInt(args[0]);
      String ruler = " ";
      for (int i = 1; i <= N; i++) {
         ruler = ruler + i + ruler;
      }
      System.out.println(ruler);
   }
}
```

i	ruler
	" "
1	" **1** "
2	" 1 **2** 1 "
3	" 1 2 1 **3** 1 2 1 "

For Loops: Subdivisions of a Ruler

```
% java RulerN 1
 1

% java RulerN 2
 1 2 1

% java RulerN 3
 1 2 1 3 1 2 1

% java RulerN 4
 1 2 1 3 1 2 1 4 1 2 1 3 1 2 1

% java RulerN 5
 1 2 1 3 1 2 1 4 1 2 1 3 1 2 1 5 1 2 1 3 1 2 1 4 1 2 1 3 1 2 1

% java RulerN 100
Exception in thread "main"
java.lang.OutOfMemoryError
```

Observation. Loops can produce a huge amount of output!

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 $(1 + 2 + \ldots + N)$	`int sum = 0;`
	`for (int i = 1; i <= N; i++)`
	`  sum += i;`
	`System.out.println(sum);`

compute a finite product $(N! = 1 \times 2 \times \ldots \times N)$	`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);`

Nesting

Nesting Conditionals and Loops

Conditionals enable you to do one of 2^n sequences of operations with n lines.

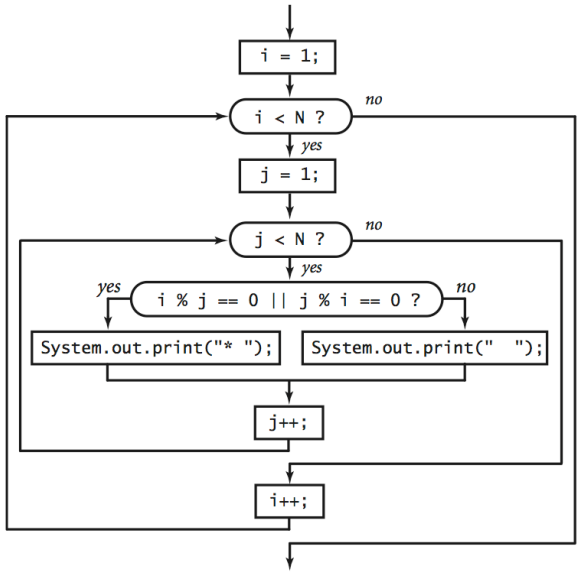
```
if (a0 > 0) System.out.print(0);
if (a1 > 0) System.out.print(1);
if (a2 > 0) System.out.print(2);
if (a3 > 0) System.out.print(3);
if (a4 > 0) System.out.print(4);
if (a5 > 0) System.out.print(5);
if (a6 > 0) System.out.print(6);
if (a7 > 0) System.out.print(7);
if (a8 > 0) System.out.print(8);
if (a9 > 0) System.out.print(9);
```

2^{10} = 1024 possible results, depending on input

Loops enable you to do an operation n times using only 2 lines of code.

```
double sum = 0.0;
for (int i = 1; i <= 1024; i++)
   sum = sum + 1.0 / i;
```

computes 1/1 + 1/2 + ... + 1/1024



More sophisticated programs.

- Nest conditionals within conditionals.
- Nest loops within loops.
- Nest conditionals within loops within loops.

Nested If Statements

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;
```

graduated income tax calculation

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;
      }
   }
}
```

Nested If Statements

Need all those braces? Not always.

```
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;
```

is shorthand for

```
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;
      }
   }
}
```

but be careful when nesting if-else statements. [See Q+A on p. 75.]

Nested If Statement Challenge

Q. What's wrong with the following for income tax calculation?

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

```
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;
```

wrong graduated income tax calculation

Monte Carlo Simulation

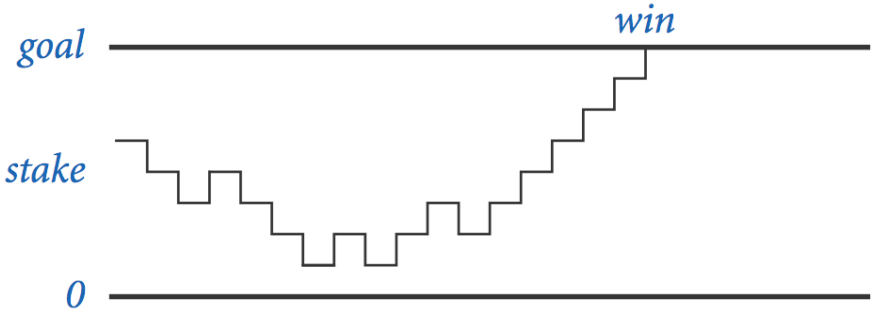
Gambler's Ruin

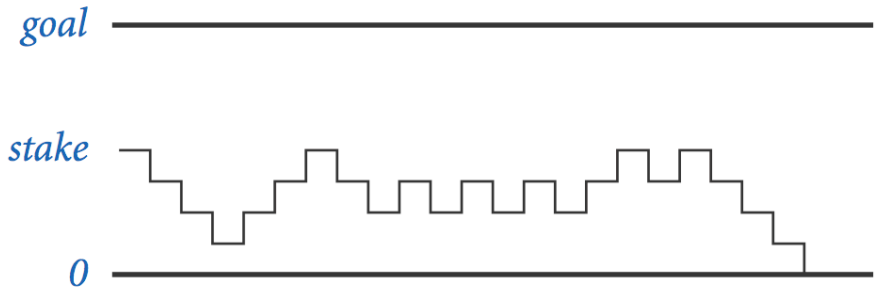
Gambler's ruin. Gambler starts with $stake and places $1 fair bets until going broke or reaching $goal.

- 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

```
public class Gambler {
   public static void main(String[] args) {
      int stake  = Integer.parseInt(args[0]);
      int goal   = Integer.parseInt(args[1]);
      int 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);
   }
}
```

Digression: Simulation and Analysis

after a substantial wait....

Fact. [see ORF 309] Probability of winning = stake ÷ goal.

Fact. [see ORF 309] Expected number of bets = stake × desired gain.

Ex. 20% chance of turning $500 into $2500, but expect to make one million $1 bets.

500/2500 = 20%

500 * (2500 - 500) = 1 million

Remark. Both facts can be proved mathematically; for more complex scenarios, computer simulation is often the best (only) plan of attack.

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`

loops	certain statements are executed repeatedly until certain conditions are met	`while`
		`for`
		`do-while`

Q & A

Q. My program is stuck in an infinite loop. How do I stop it?

Q. Why doesn't the statement if (a <= b <= c) work?

Q. What (if anything) is wrong with each of the following statements?

- if (a > b) then c = 0;
- if a > b { c = 0; }
- if (a > b) c = 0;
- if (a > b) c = 0 else b = 0;

Assignments #2

- Due: 21 Sep. 12:00 pm
- Ex 1.3.17, 1.3.27, and 1.3.34
- How to submit:

1. Prepare your HW by using wordprocesses if necessary (e.g. Word, HWP…)
   Note: Don't use papers!
2. When you need to write code, make the separated files. (and refer to the files in the document.)
3. Zip all files
4. Submit the zib file to:
   https://www.dropbox.com/request/XVNtQiUo4XpEWgJ68Rhh

5. Important!: Input your name correctly!

Extra Slides

Oblivious Sorting

Sort. Read in 3 integers and rearrange them in ascending order.

```
public class Sort3 {
   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 > c) { int t = b; b = c; c = t; }
      if (a > b) { int t = a; a = b; b = t; }
      if (b > c) { int t = b; b = c; c = t; }

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

read in 3 integers from command-line

swap b and c

swap a and b

swap b and c

```
% java Sort3 9 8 7
7 8 9

% java Sort3 2 1 7
1 2 7
```

Puzzle 1. Sort 4 integers with 5 compare-exchanges.

Puzzle 2. Sort 6 integers with 12.

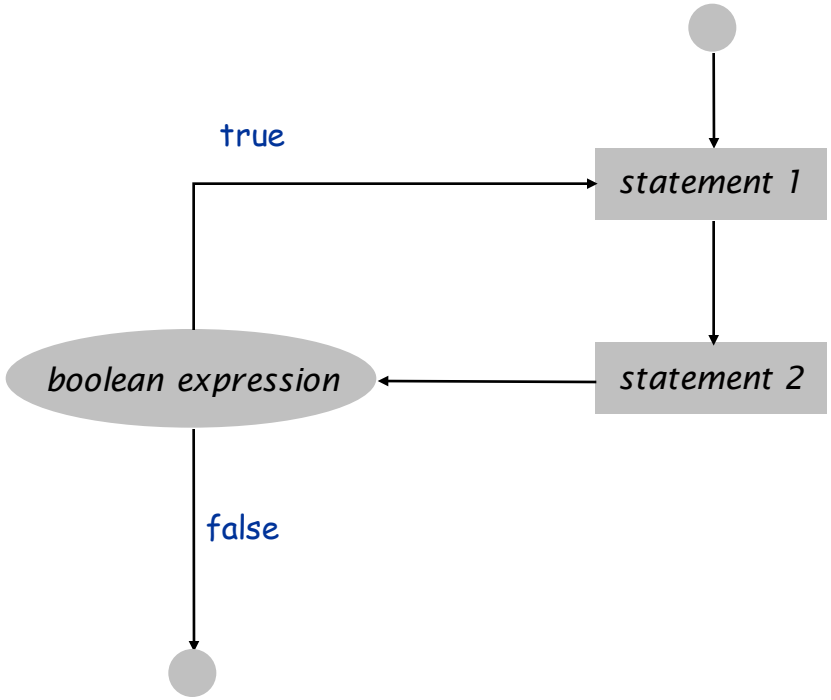
Do-While Loop

The `do-while` loop. A less common repetition structure.

- Execute sequence of statements.
- Check loop-continuation condition.
- Repeat.

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

do-while loop syntax



Do-While Loop

Ex. Find a point (x, y) that is uniformly distributed in unit disc.

- Pick a random point in unit square.
- Check if point is also in unit disc.
- Repeat.

```
do {
   x = 2.0 * Math.random() - 1.0;
   y = 2.0 * Math.random() - 1.0;
} while (x*x + y*y > 1.0);
```

between -1 and 1