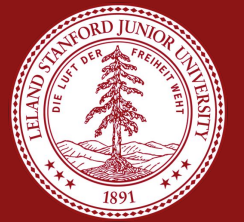
Booleans and Control Flow

Lecture 5

CS106A, Summer 2019

Sarai Gould && Laura Cruz-Albrecht

With inspiration from slides created by Keith Schwarz, Mehran Sahami, Eric Roberts, Stuart Reges, Chris Piech and others.



Plan for Today

- Announcements
- Recap: Java, Variables and Expressions
- Shorthand Operators & Constants
- Revisiting Control Flow
  - If and While
  - For

Announcements

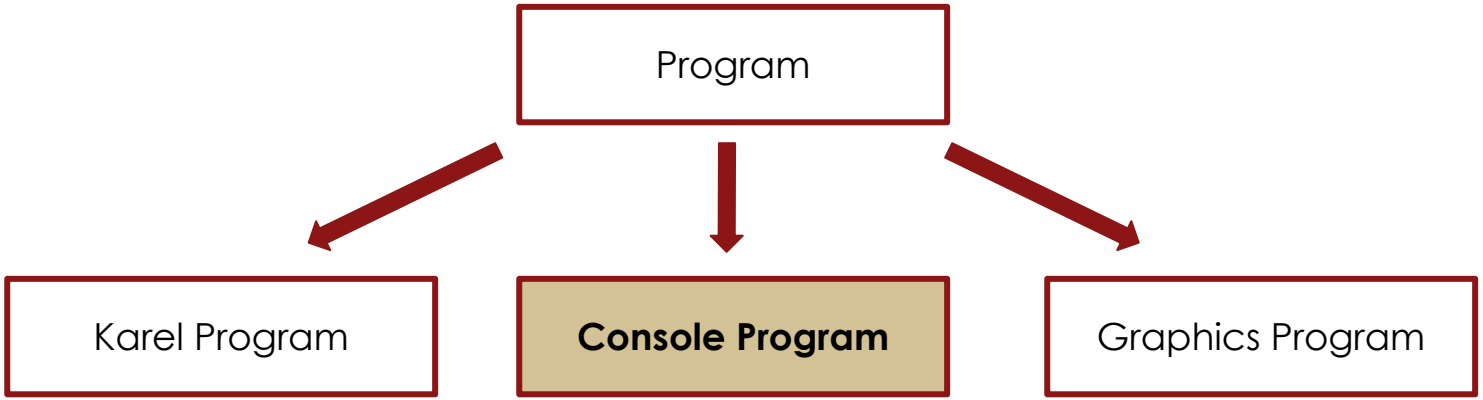
- No lecture or sections Thursday for 4th of July.
  - If your section in cancelled, please try to attend a Wednesday section or Friday's section (11:30am in Skilling Auditorium)
- No LaIR Wednesday, July 3rd due to Holiday
- Assignment 1 due 10AM Wednesday
  - Practice submitting
- Piazza

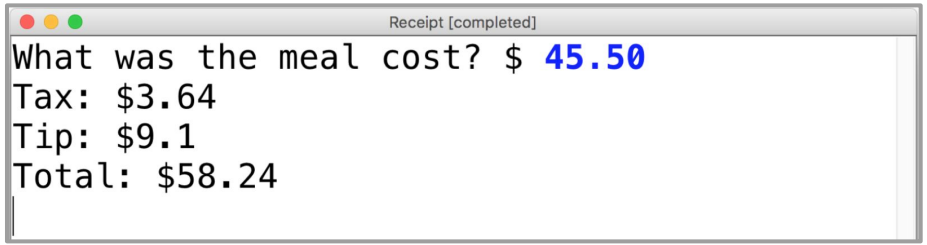
Plan for Today

- Announcements
- Recap: Java, Variables and Expressions
- Shorthand Operators & Constants
- Revisiting Control Flow
  - If and While
  - For

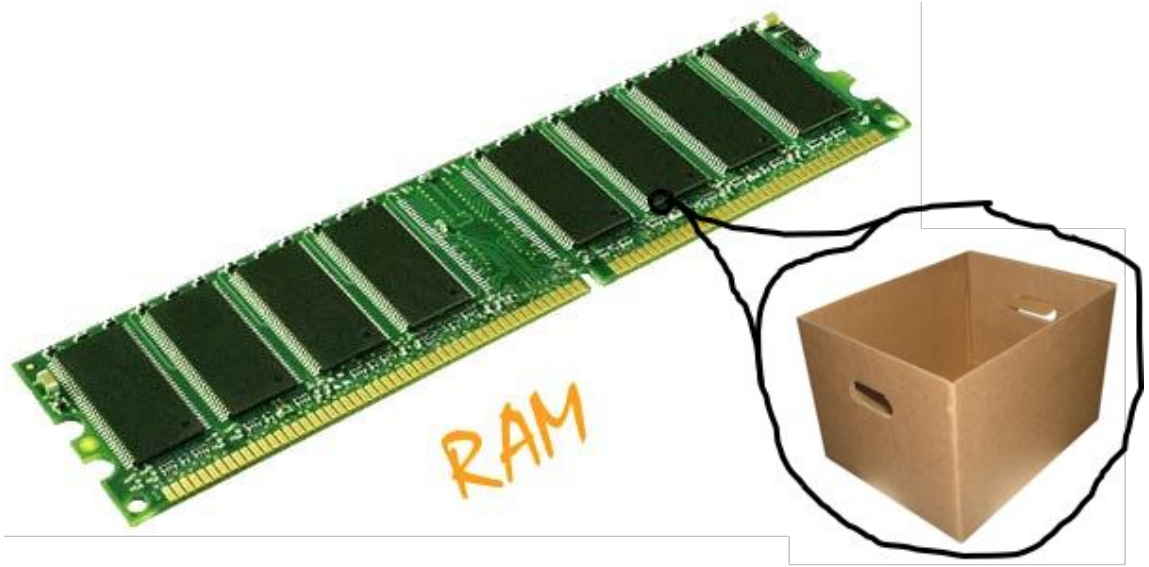
Java

Types of Programs





Variables = Boxes



* my computer has space for about 2 billion boxes

Making a New Variable

```
int myVar = 22;
```

(contains an `int`)

3 Properties

type name value

```
int myVar = 22;
```

(contains an `int`)

2 Steps

(1) **Declaration**
make a box

```
int myVar = 22;
```

(2) **Assignment**
put something inside the box

Create, Modify, Use

```
// Create a variable of type int, called coffees
// with the value 2
int coffees = 2;

// Modify coffees to be 1 greater (did someone
// just drink another coffee...)
coffees = coffees + 1;

// Use the value in coffees (output it)
println("I had " + coffees + " today.");
```

Expressions

- You can combine literals or variables together into ***expressions*** using *binary operators*:

+	Addition	*	Multiplication
-	Subtraction	/	Division
		%	Remainder

Type Interactions

`int` and `int` results in an `int`	`1 / 2`	→ `0`
`int` and `double` results in a `double`	`1 / 2.0`	→ `0.5`
`double` and `double` results in a `double`	`4.4 * 0.5`	→ `2.2`
`String` and `int` results in a `String`	`"abc" + 3`	→ `"abc3"`

* operations return the most expressive type

`String` > `double` > `int` > `char` > `boolean`

Plan for Today

- Announcements
- Recap: Java, Variables and Expressions
- Shorthand Operators & Constants
- Revisiting Control Flow
  - If and While
  - For

Shorthand Operators

Shorthand:

```
// +, -, /, *, % any value
variable += value;
x -= 3;
y /= 5;
z *= someValue;
k %= 10;
```

Equivalent Longer Version:

```
variable = variable + value;
x = x - 3;
y = y / 5;
z = z * someValue;
k = k % 10;
```

Shorthand Operators

Shorthand:

```
// +, -, /, *, % any value
variable += value;
x -= 3;
y /= 5;
z *= someValue;
k %= 10;

// add or subtract exactly 1
x++;
y--;
```

Equivalent Longer Version:

```
variable = variable + value;
x = x - 3;
y = y / 5;
z = z * someValue;
k = k % 10;

x = x + 1;
y = y - 1;
```

Receipt Program - Before

```
public class Receipt extends ConsoleProgram {
    public void run() {
        double subtotal = readDouble("Meal cost? $");
        double tax = subtotal * 0.08;
        double tip = subtotal * 0.20;
        double total = subtotal + tax + tip;

        println("Tax: $" + tax);
        println("Tip: $" + tip);
        println("Total: $" + total);
    }
}
```

Receipt Program - Before

```
public class Receipt extends ConsoleProgram {
    public void run() {
        double subtotal = readDouble("Meal cost? $");
        double tax = subtotal * 0.08;
        double tip = subtotal * 0.20;
        double total = subtotal + tax + tip;

        println("Tax: $" + tax);
        println("Tip: $" + tip);
        println("Total: $" + total);
    }
}
```

Receipt Program - After

```
public class Receipt extends ConsoleProgram {
    private static final double TAX_RATE = 0.08;
    private static final double TIP_RATE = 0.2;
    public void run() {
        double subtotal = readDouble("Meal cost? $");
        double tax = subtotal * TAX_RATE;
        double tip = subtotal * TIP_RATE;
        double total = subtotal + tax + tip;

        println("Tax: $" + tax);
        println("Tip: $" + tip);
        println("Total: $" + total);
    }
}
```

Constants

- **constant**: A variable that cannot be changed after it is initialized. Declared at the top of your class, outside of the `run()` method but inside `public class` ***Name***. Can be used anywhere in that class.

- Better style: can easily change their values in your code, and are also easier to read.

- Syntax:
  `private static final` *type UPPER_CASE_NAME = value*`;`

- Example:
```
private static final double TAX_RATE = 0.08;
```

ReceiptForFive - Before

```
public class ReceiptForFive extends ConsoleProgram {
     public void run() {
          double subtotal1 = readDouble("Meal cost? $");
          double tax1 = subtotal * 0.08;
          double tip1 = subtotal * 0.20;
          double total1 = subtotal1 + tax1 + tip1;
          println("Total for person 1: $" + total1);
          ...
          double subtotal5 = readDouble("Meal cost? $");
          double tax5 = subtotal * 0.08;
          double tip5 = subtotal * 0.20;
          double total5 = subtotal1 + tax1 + tip1;
          println("Total for person 5: $" + total5);
     }
}
```

ReceiptForFive - After

```
public class ReceiptForFive extends ConsoleProgram {
    private static final double TAX_RATE = 0.08;
    private static final double TIP_RATE = 0.2;
    public void run() {
        double subtotal1 = readDouble("Meal cost? $");
        double tax1 = subtotal * TAX_RATE;
        double tip1 = subtotal * TIP_RATE;
        double total1 = subtotal1 + tax1 + tip1;
        println("Total for person 1: $" + total1);
        ...
        double subtotal5 = readDouble("Meal cost? $");
        double tax5 = subtotal * TAX_RATE;
        double tip5 = subtotal * TIP_RATE;
        double total5 = subtotal1 + tax1 + tip1;
        println("Total for person 5: $" + total5);
    }
}
```

much better

Plan for Today

- Announcements
- Recap: Java, Variables and Expressions
- Shorthand Operators & Constants
- Revisiting Control Flow
  - If and While
  - For

Conditions in Karel

```
if (beepersPresent()) {
    body
}
```

```
while (frontIsClear()) {
    body
}
```

Conditions in Java

```
if (condition) {
    body
}
```

```
while (condition) {
    body
}
```

The **condition** should be a "boolean" which is either `true` or `false`.

Booleans

$$1 < 2$$

Booleans

```
1 < 2
```

```
true
```

Relational Operators

Operator	Meaning	Example	Value
==	equals	`1 + 1 == 2`	`true`
!=	does not equal	`3.2 != 2.5`	`true`
<	less than	`10 < 5`	`false`
>	greater than	`10 > 5`	`true`
<=	less than or equal to	`126 <= 100`	`false`
>=	greater than or equal to	`5.0 >= 5.0`	`true`

* all have equal precedence

Relational Operators

Operator	Meaning	Example	Value
==	equals	`1 + 1 == 2`	`true`
!=	does not equal	`3.2 != 2.5`	`true`
<	less than	`10 < 5`	`false`
>	greater than	`10 > 5`	`true`
<=	less than or equal to	`126 <= 100`	`false`
>=	greater than or equal to	`5.0 >= 5.0`	`true`

* all have equal precedence

Relational Operators

```
if (1 < 2) {
  println("1 is less than 2");
}
```

Relational Operators

```
if (1 < 2) {
  println("1 is less than 2");
}
```

```
int num = readInt("Enter a number: ");
if (num == 0) {
  println("That number is 0");
} else {
  println("That number is not 0.");
}
```

A note on If-Else

```
int num = readInt("Enter a number: ");
if (num == 0) {
    println("Your number is 0");
} else {
    if (num > 0) {
        println("Your number is positive");
    } else {
        println("Your number is negative");
    }
}
```

Else If

```
int num = readInt("Enter a number: ");
if (num == 0) {
    println("Your number is 0");
} else if (num > 0) {
    println("Your number is positive");
} else {
    println("Your number is negative");
}
```

Else If

```
int num = readInt("Enter a number: ");
if (num == 0) {
    println("Your number is 0");
} else if (num > 0) {
    println("Your number is positive");
} else {
    println("Your number is negative");
}
```

- Runs first group of statements if **first condition** is true;
- otherwise, runs second group of statements if **second condition** is true;
- otherwise, runs third group of statements.

You can have multiple `else if` clauses together.

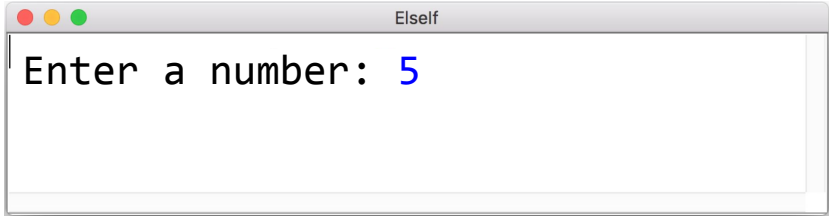
Else If

```
int num = readInt("Enter a number: ");
if (num == 0) {
    println("Your number is 0");
} else if (num > 0) {
    println("Your number is positive");
} else {
    println("Your number is negative");
}
```

Else If

```
int num = readInt("Enter a number: ");
if (num == 0) {
    println("Your number is 0");
} else if (num > 0) {
    println("Your number is positive");
} else {
    println("Your number is negative");
}
```

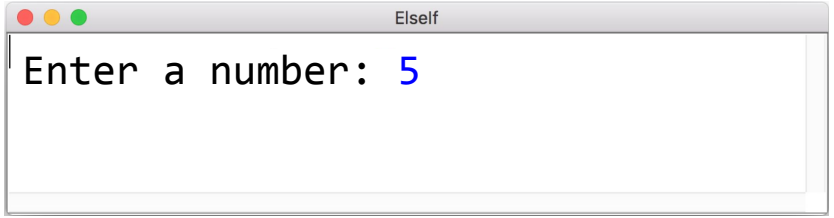
5

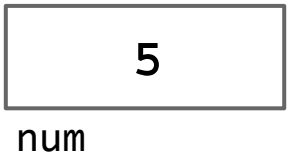


Else If

```
int num = readInt("Enter a number: ");
if (num == 0) {
    println("Your number is 0");
} else if (num > 0) {
    println("Your number is positive");
} else {
    println("Your number is negative");
}
```

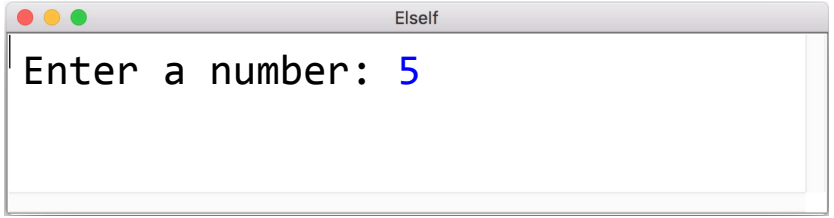
5

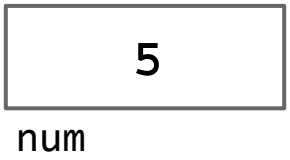




Else If

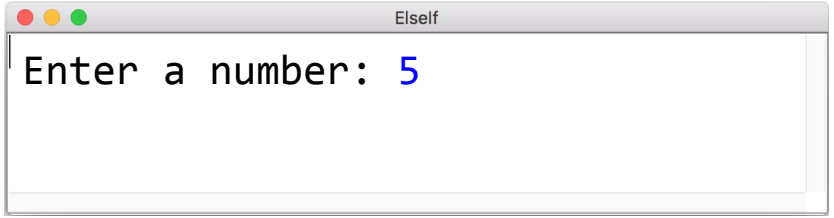
```
int num = readInt("Enter a number: ");
if (num == 0) {
    println("Your number is 0");
} else if (num > 0) {
    println("Your number is positive");
} else {
    println("Your number is negative");
}
```

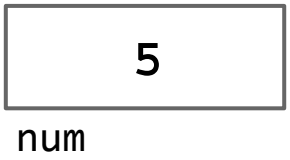




Else If

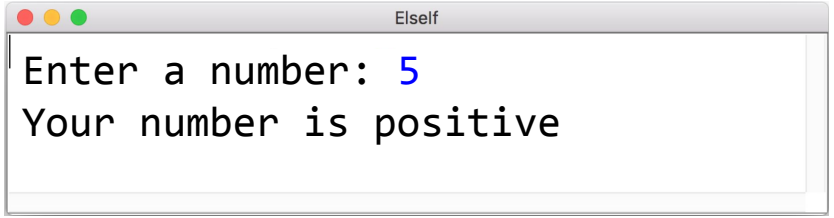
```
int num = readInt("Enter a number: ");
if (num == 0) {
    println("Your number is 0");
} else if (num > 0) {
    println("Your number is positive");
} else {
    println("Your number is negative");
}
```

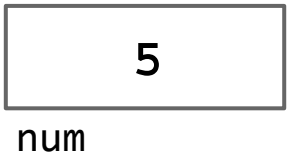




Else If

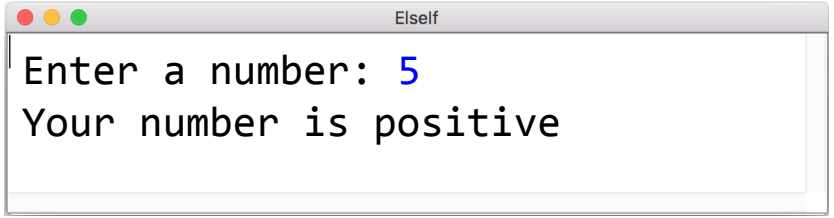
```
int num = readInt("Enter a number: ");
if (num == 0) {
    println("Your number is 0");
} else if (num > 0) {
    println("Your number is positive");
} else {
    println("Your number is negative");
}
```

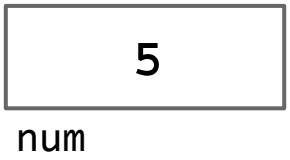




Else If

```
int num = readInt("Enter a number: ");
if (num == 0) {
    println("Your number is 0");
} else if (num > 0) {
    println("Your number is positive");
} else {
    println("Your number is negative");
}
```





Cascading `else-if` vs. Stacked `if`

```
if (condition1) {
    …
} else if (condition2) {
    …
} else if (condition3) {
    …
} else {
    …
}
```

```
if (condition1) {
    …
}
if (condition2) {
    …
}
if (condition3) {
    …
}
…
```

Cascading `else-if` vs. Stacked `if`

```
if (num < 0) {
    ...
} else if (num < 10) {
    ...
} else if (num < 20) {
    ...
} else {
    ...
}
```

Use cascading **`else-if`** statements if **only one** condition should be true.
Exactly one code block gets executed.

Cascading `else-if` vs. Stacked `if`

```
if (num < 0) {
    …
} else if (num < 10) {
    …
} else if (num < 20) {
    …
} else {
    …
}
```

Use cascading **`else-if`** statements if **only one** condition should be true. Exactly one code block gets executed.

```
if (num < 0) {
    …
}
if (num < 10) {
    …
}
if (num < 20) {
    …
}
…
```

Use repeated **`if`** statements if **more or less than one** condition can be true. Multiple code blocks (or none) may get executed.

Cascading `else-if` vs. Stacked `if`

```
if (num < 0) {
    …
} else if (num < 10) {
    …
} else if (num < 20) {
    …
} else {
    …
}
```

`num = 5`

```
if (num < 0) {
    …
}
if (num < 10) {
    …
}
if (num > 20) {
    …
}
…
```

Use cascading **`else-if`** statements if **only one** condition should be true. Exactly one code block gets executed.

Use repeated **`if`** statements if **more or less than one** condition can be true. Multiple code blocks (or none) may get executed.

Cascading `else-if` vs. Stacked `if`

```
if (num < 0) {
    ...
} else if (num < 10) {
    ...
} else if (num < 20) {
    ...
} else {
    ...
}
```

`num = 5`

```
if (num < 0) {
    ...
}
if (num < 10) {
    ...
}
if (num > 20) {
    ...
}
...
```

Use cascading **`else-if`** statements if **only one** condition should be true. Exactly one code block gets executed.

Use repeated **`if`** statements if **more or less than one** condition can be true. Multiple code blocks (or none) may get executed.

Cascading `else-if` vs. Stacked `if`

```
if (num < 0) {
    ...
} else if (num < 10) {
    ...
} else if (num < 20) {
    ...
} else {
    ...
}
```

`num = 5`

```
if (num < 0) {
    ...
}
if (num < 10) {
    ...
}
if (num < 20) {
    ...
}
...
```

Use cascading **`else-if`** statements if **only one** condition should be true. Exactly one code block gets executed.

Use repeated **`if`** statements if **more or less than one** condition can be true. Multiple code blocks (or none) may get executed.

Example: Sentinel Loops

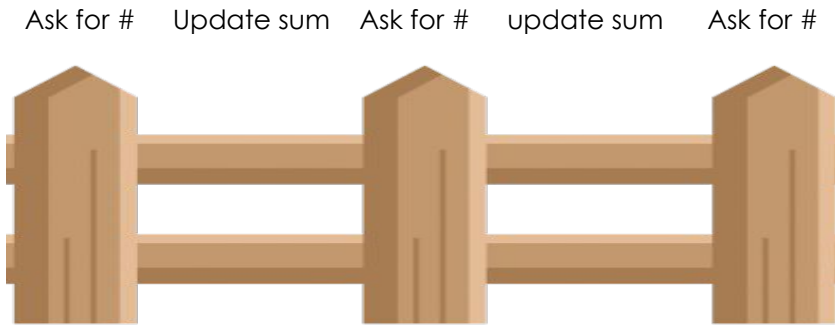
- **sentinel**: A value that signals the end of user input.
  - **sentinel loop**: Repeats until a sentinel value is seen.

- Example: Write a program that prompts the user for numbers until the user types -1, then output the sum of the numbers.
  - In this case, -1 is the sentinel value

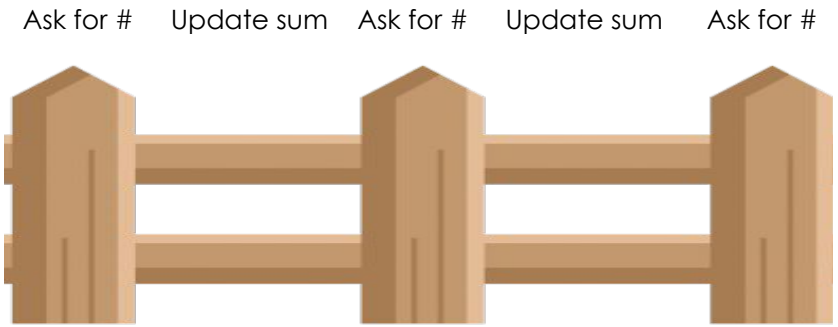
```
Type a number: 10
Type a number: 20
Type a number: 30
Type a number: -1
Sum is 60
```

Example: Sentinel Loops

```
// fencepost problem!
// ask for number - post
// add number to sum - fence
```



Example: Sentinel Loops




```
// fencepost problem!
// ask for number - post
// add number to sum - fence

int sum = 0;
int num = readInt("Enter a number: ");
while (num != -1) {
    sum += num;
    num = readInt("Enter a number: ");
}
println("Sum is " + sum);
```

Example: Sentinel Loops

```
// Solution # 2: “break” out of the loop
// harder to see loop end condition here
// ONLY appropriate to use in fencepost cases

int sum = 0;
while (true) {
    int num = readInt("Enter a number: ");
    if (num == -1) {
        break;   // immediately exits loop
    }
    sum += num;
}
println("Sum is " + sum);
```

Logical Operators

Operator	Description	Example	Result
`!`	not	`!(2 == 3)`	`true`
`&&`	and	`(2 == 3) && (-1 < 5)`	`false`
`||`	or	`(2 == 3) || (-1 < 5)`	`true`

Logical Operators

Operator	Description	Example	Result
`!`	not	`!(2 == 3)`	`true`
`&&`	and	`(2 == 3) && (-1 < 5)`	`false`
`||`	or	`(2 == 3) || (-1 < 5)`	`true`

Cannot "chain" tests as in algebra; use **&&** or **||** instead.

```
// incorrect (assume x is 15)
2 <= x <= 10
true <= 10
Error!
```

```
// correct version
2 <= x && x <= 10
true && false
false
```

Precedence Madness

Precedence: arithmetic > relational > logical

```
5 * 7 >= 3 + 5 * (7 - 1) && 7 <= 11
```

Precedence Madness

Precedence: arithmetic > relational > logical

```
5 * 7 >= 3 + 5 * (7 - 1) && 7 <= 11
5 * 7 >= 3 + 5 * 6 && 7 <= 11
35    >= 3 + 30 && 7 <= 11
35    >= 33 && 7 <= 11
true && true
true
```

Boolean Variables

```
// Store expressions that evaluate to true/false
boolean x = 1 < 2;          // true
boolean y = 5.0 == 4.0;     // false
```

Boolean Variables

```
// Store expressions that evaluate to true/false
boolean x = 1 < 2;          // true
boolean y = 5.0 == 4.0;     // false

// Directly set to true/false
boolean isFamilyVisiting = true;
boolean isRaining = false;
```

Boolean Variables

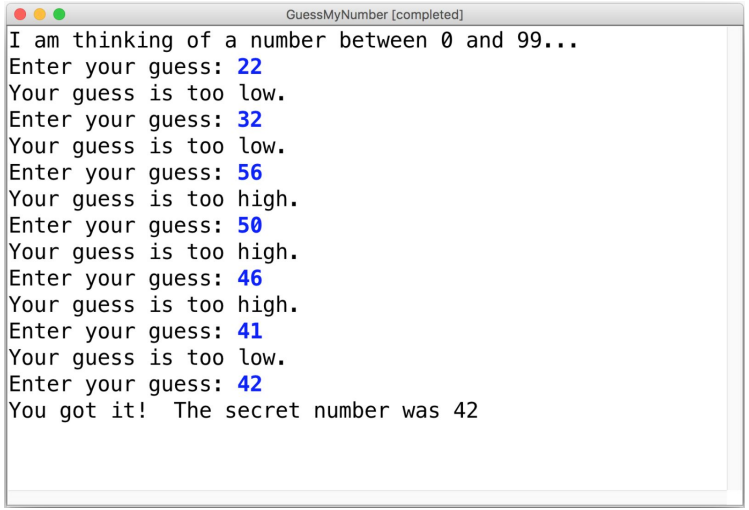
```
// Store expressions that evaluate to true/false
boolean x = 1 < 2;          // true
boolean y = 5.0 == 4.0;     // false

// Directly set to true/false
boolean isFamilyVisiting = true;
boolean isRaining = false;

// Ask the user a true/false (yes/no) question
boolean playAgain = readBoolean("Play again?", "y", "n");
if (playAgain) {
...
```

Practice: Guess My Number

- Let's write a program called *GuessMyNumber* that prompts the user for a number until they guess our secret number.
- If a guess is incorrect, the program should provide a hint; specifically, whether the guess is too high or too low.



Let’s Code It!

Summary: Conditions in Java

```
if (condition) {
    body
}
```

```
while (condition) {
    body
}
```

Similar to Karel, except that now the *condition* can be any expression that evaluates to `true` or `false`.

Plan for Today

- Announcements
- Recap: Java, Variables and Expressions
- Shorthand Operators & Constants
- Revisiting Control Flow
  - If and While
  - For

For Loops in Karel

```
for (int i = 0; i < numTimes; i++) {
    statement;
    statement;
}
```

Repeats the statements in the body *numTimes*.

For Loop Redux

```
for (int i = 0; i < 3; i++) {
    println("I love CS106A!");
}
```

For Loop Redux

This code is run once, just before the for loop starts

```
for (int i = 0; i < 3; i++) {
    println("I love CS106A!");
}
```

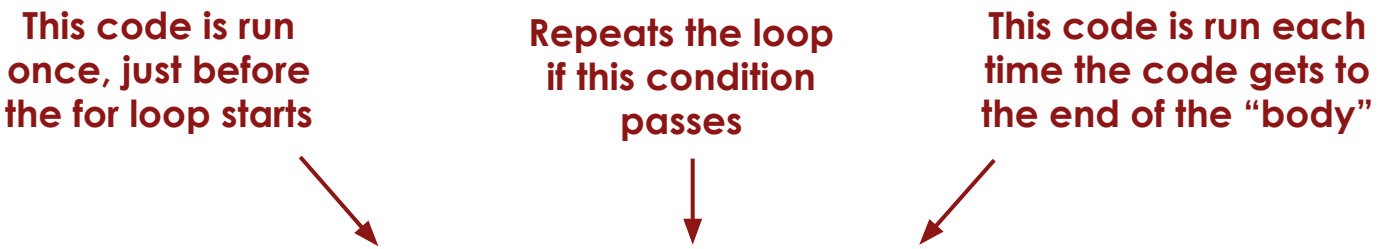
For Loop Redux

This code is run once, just before the for loop starts

This code is run each time the code gets to the end of the "body"

```
for (int i = 0; i < 3; i++) {
    println("I love CS106A!");
}
```

For Loop Redux




```
for (int i = 0; i < 3; i++) {
    println("I love CS106A!");
}
```

For Loop Redux

This code is run once, just before the for loop starts

Repeats the loop if this condition passes

This code is run each time the code gets to the end of the "body"

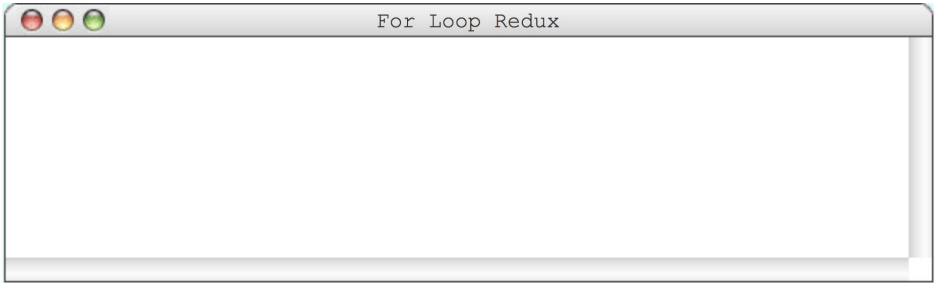
```
for (int i = 0; i < 3; i++) {
    println("I love CS106A!");
}
```

```
int i = 0;
while (i < 3) {
    println("I love CS106A!");
    i++;
}
```

* Not exactly identical, but similar conceptually.

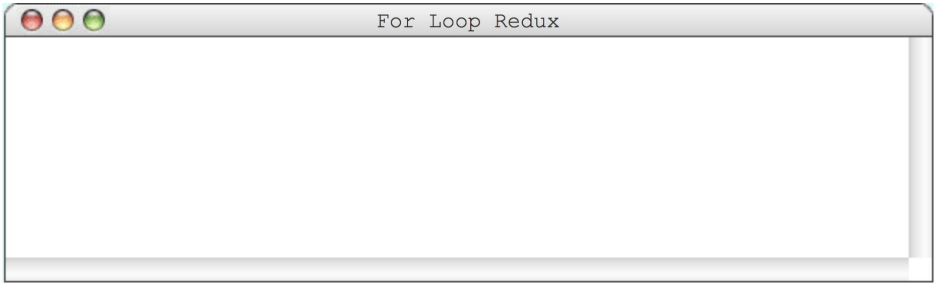
For Loop Redux

```
for (int i = 0; i < 3; i++) {
   println("I love CS106A!");
}
```



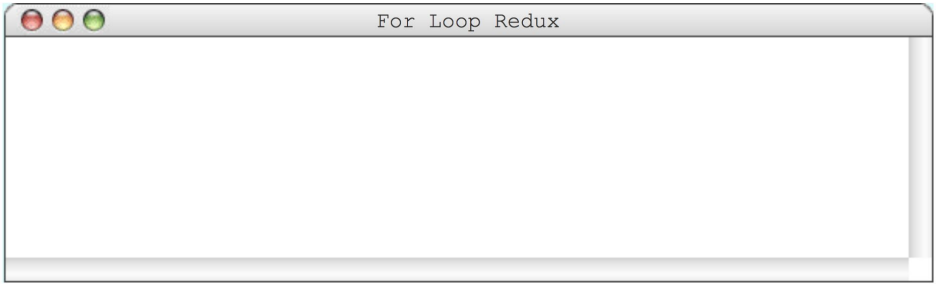
For Loop Redux

```
for (int i = 0; i < 3; i++) {
    println("I love CS106A!");
}
```



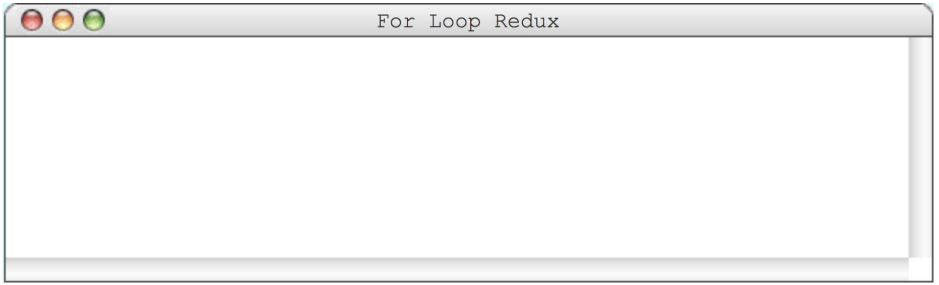
For Loop Redux


```
for (int i = 0; i < 3; i++) {
   println("I love CS106A!");
}
```



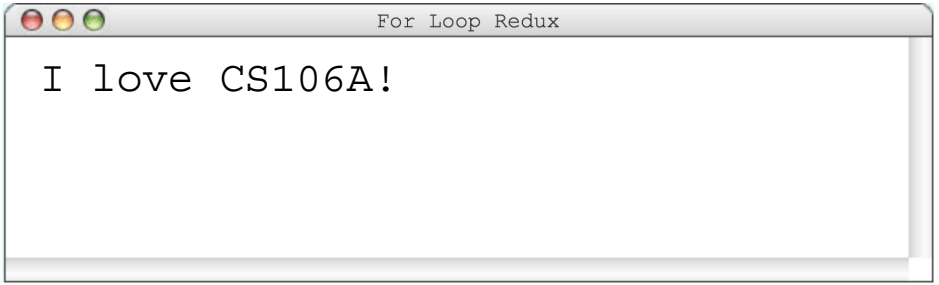
For Loop Redux


```
for (int i = 0; i < 3; i++) {
   println("I love CS106A!");
}
```



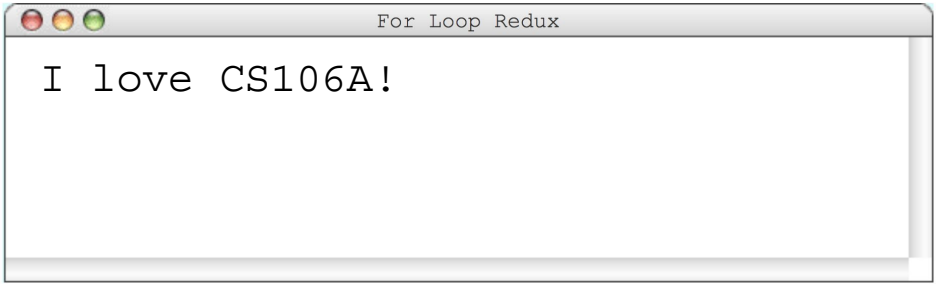
For Loop Redux


```
for (int i = 0; i < 3; i++) {
    println("I love CS106A!");
}
```



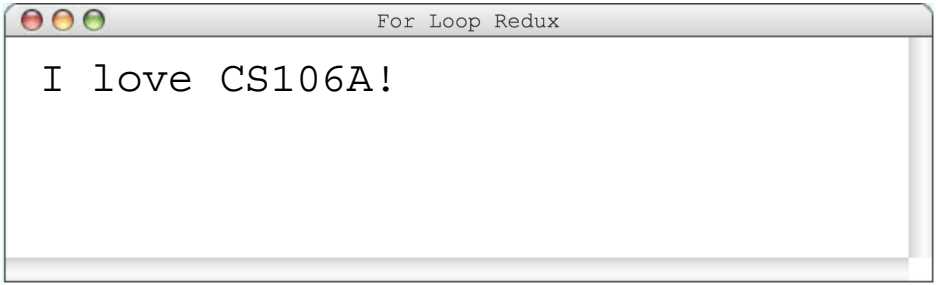
For Loop Redux


```
for (int i = 0; i < 3; i++) {
   println("I love CS106A!");
}
```



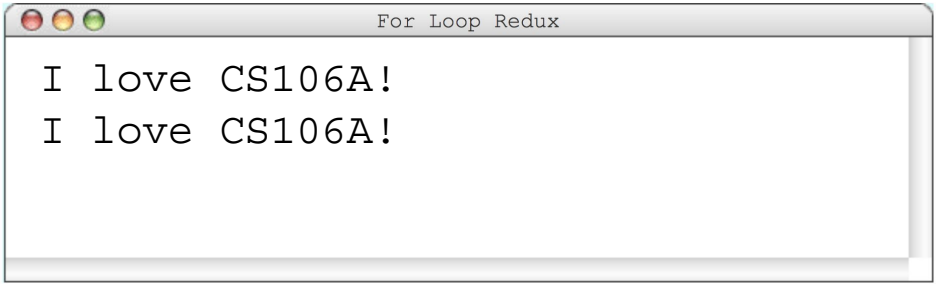
For Loop Redux


```
for (int i = 0; i < 3; i++) {
   println("I love CS106A!");
}
```



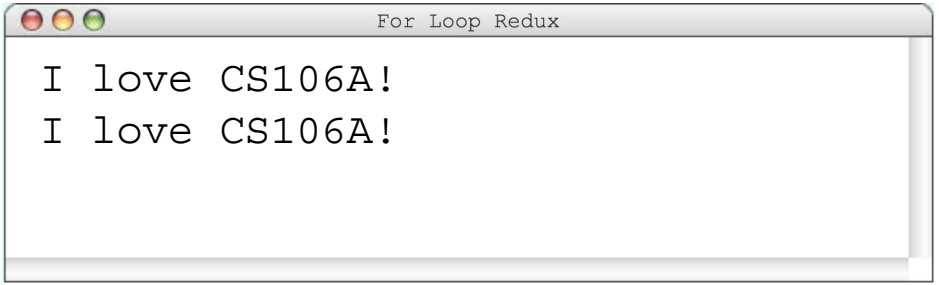
For Loop Redux


```
for (int i = 0; i < 3; i++) {
    println("I love CS106A!");
}
```



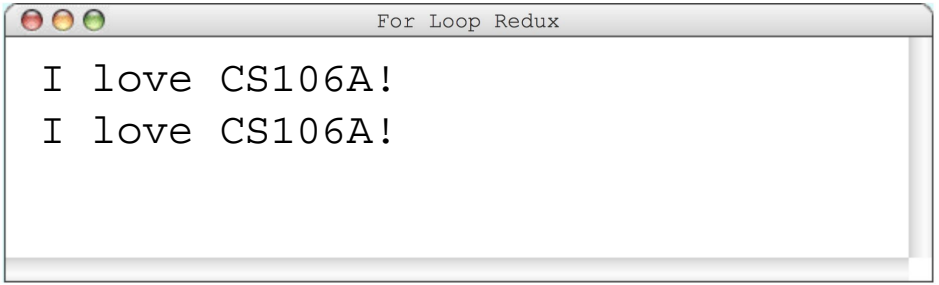
For Loop Redux


```
for (int i = 0; i < 3; i++) {
   println("I love CS106A!");
}
```



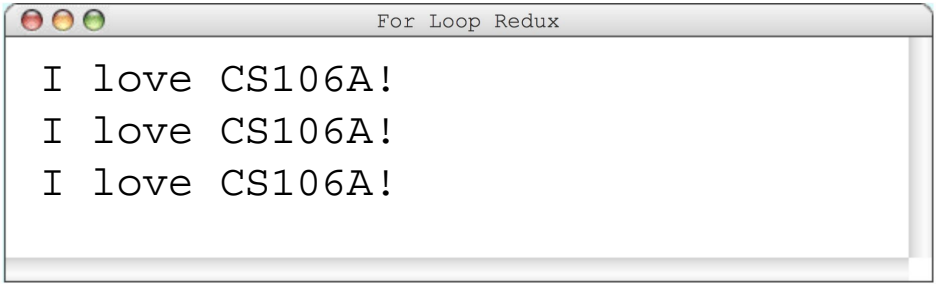
For Loop Redux


```
for (int i = 0; i < 3; i++) {
   println("I love CS106A!");
}
```



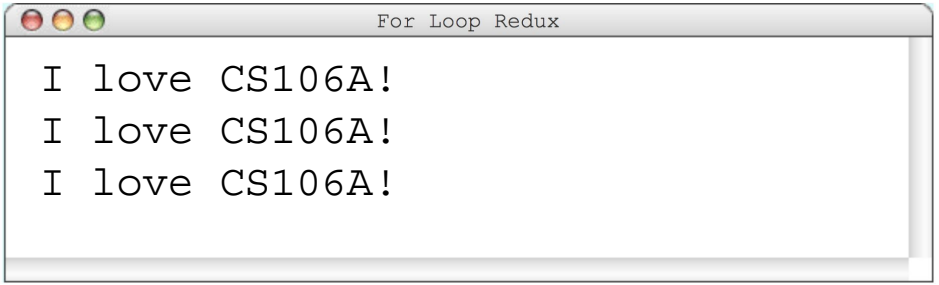
For Loop Redux


```
for (int i = 0; i < 3; i++) {
    println("I love CS106A!");
}
```



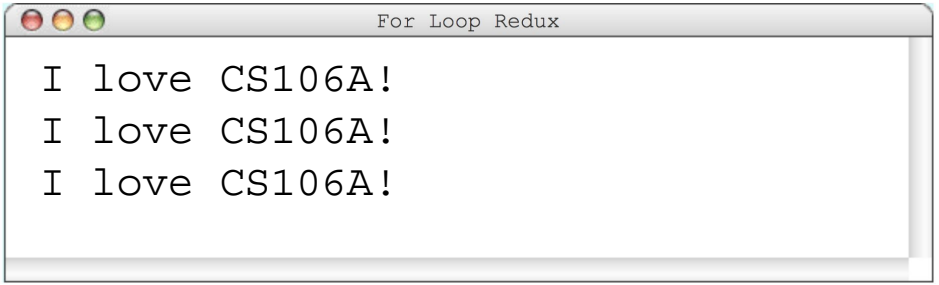
For Loop Redux


```
for (int i = 0; i < 3; i++) {
   println("I love CS106A!");
}
```



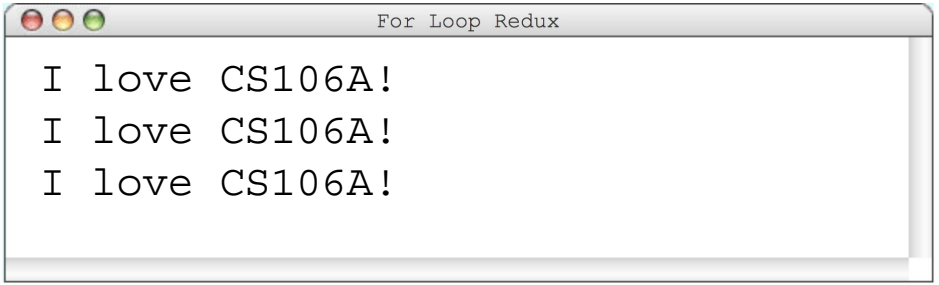
For Loop Redux


```
for (int i = 0; i < 3; i++) {
   println("I love CS106A!");
}
```



For Loop Redux


```
for (int i = 0; i < 3; i++) {
   println("I love CS106A!");
}
```



You Can Use the For Loop Variable

Using the For Loop Variable

How would you **`println`** the first 100 even numbers?

```
PrintEven...
0
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32
34
36
38
```

Using the For Loop Variable

```
// prints the first 100 even numbers
for (int i = 0; i < 100; i++) {
    println(i * 2);
}
```

Another Take

```
// prints the first 100 even numbers
for (int i = 0; i < 2 * 100; i += 2) {
    println(i);
}
```

Using the For Loop Variable

```
// Adds up the numbers 1 through 42
int sum = 0;
for (int i = 1; i <= 42; i++) {
    sum += i;
}
println("Sum is " + sum);
```

Using the For Loop Variable

```
// Launch countdown
for (int i = 10; i >= 1; i--) {
    println(i);
}
println("Blast off!");
```

Output:

```
10
9
8
...
1
Blast off!
```

Using the For Loop Variable

```
// Launch countdown
for (int i = 10; i >= 1; i--) {
    println(i);
}
println("Blast off!");
```

Output:

```
10
9
8
...
1
Blast off!
```

* **Note**: you can't use the `for` loop variable on the Karel assignment. Can only use Karel features.

Plan for Today

- Announcements
- Recap: Java, Variables and Expressions
- Shorthand Operators & Constants
- Revisiting Control Flow
  - If and While
  - For

Reminder: submit lecture feedback on your assigned days.

Next time: More control flow, methods in Java