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 is 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
Types of Programs

- Karel Program
- Console Program
- Graphics Program

Receipt [completed]

What was the meal cost? $45.50
Tax: $3.64
Tip: $9.1
Total: $58.24
Variables = Boxes

* my computer has space for about 2 billion boxes
Making a New Variable

```c
int myVar = 22;
```

(contains an `int`)
3 Properties

int myVar = 22;

(type: int)
(name: myVar)
(value: 22)
2 Steps

(1) Declaration
make a box

int myVar = 22;

(2) Assignment
put something inside the box
// 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
  - Subtraction
  * Multiplication
  / 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;
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: ");
        println("Tip: ");
        println("Total: ");
    }
}
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);
    }
}
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: ");
        println("Tip: ");
        println("Total: ");
    }
}
• **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:
  ```java
  private static final type UPPER_CASE_NAME = value;
  ```

• Example:
  ```java
  private static final double TAX_RATE = 0.08;
  ```
public class ReceiptForFive extends ConsoleProgram {
    public void run() {
        double subtotal1 = readDouble("Meal cost? ");
        double tax1 = subtotal1 * 0.08;
        double tip1 = subtotal1 * 0.20;
        double total1 = subtotal1 + tax1 + tip1;
        println("Total for person 1: "+ total1);
        ...
        double subtotal5 = readDouble("Meal cost? ");
        double tax5 = subtotal5 * 0.08;
        double tip5 = subtotal5 * 0.20;
        double total5 = subtotal5 + tax5 + tip5;
        println("Total for person 5: "+ total5);
    }
}
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 = subtotal1 * TAX_RATE;
        double tip1 = subtotal1 * TIP_RATE;
        double total1 = subtotal1 + tax1 + tip1;
        println("Total for person 1: $" + total1);
        ...
        double subtotal5 = readDouble("Meal cost? $");
        double tax5 = subtotal5 * TAX_RATE;
        double tip5 = subtotal5 * TIP_RATE;
        double total5 = subtotal5 + tax5 + tip5;
        println("Total for person 5: $" + total5);
    }
}
Plan for Today

● Announcements
● Recap: Java, Variables and Expressions
● Shorthand Operators & Constants
● Revisiting Control Flow
  ○ If and While
  ○ For
if (beepersPresent()) {
    body
}

while (frontIsClear()) {
    body
}
The `condition` should be a “boolean” which is either `true` or `false`. 
Booleans

1 < 2
Booleans

$1 < 2$

true
### Relational Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning</th>
<th>Example</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>==</code></td>
<td>equals</td>
<td><code>1 + 1 == 2</code></td>
<td>true</td>
</tr>
<tr>
<td><code>!=</code></td>
<td>does not equal</td>
<td><code>3.2 != 2.5</code></td>
<td>true</td>
</tr>
<tr>
<td><code>&lt;</code></td>
<td>less than</td>
<td><code>10 &lt; 5</code></td>
<td>false</td>
</tr>
<tr>
<td><code>&gt;</code></td>
<td>greater than</td>
<td><code>10 &gt; 5</code></td>
<td>true</td>
</tr>
<tr>
<td><code>&lt;=</code></td>
<td>less than or equal to</td>
<td><code>126 &lt;= 100</code></td>
<td>false</td>
</tr>
<tr>
<td><code>&gt;=</code></td>
<td>greater than or equal to</td>
<td><code>5.0 &gt;= 5.0</code></td>
<td>true</td>
</tr>
</tbody>
</table>

* all have equal precedence
### Relational Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning</th>
<th>Example</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>==</td>
<td>equals</td>
<td>$1 + 1 == 2$</td>
<td>true</td>
</tr>
<tr>
<td>!=</td>
<td>does not equal</td>
<td>$3.2 != 2.5$</td>
<td>true</td>
</tr>
<tr>
<td>&lt;</td>
<td>less than</td>
<td>$10 &lt; 5$</td>
<td>false</td>
</tr>
<tr>
<td>&gt;</td>
<td>greater than</td>
<td>$10 &gt; 5$</td>
<td>true</td>
</tr>
<tr>
<td>&lt;=</td>
<td>less than or equal to</td>
<td>$126 &lt;= 100$</td>
<td>false</td>
</tr>
<tr>
<td>&gt;=</td>
<td>greater than or equal to</td>
<td>$5.0 &gt;= 5.0$</td>
<td>true</td>
</tr>
</tbody>
</table>

* all have equal precedence
Relational Operators

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

```java
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

```java
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

```java
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");
}
```
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.
```java
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");
}
```
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");
}
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");
}
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");
}
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");
}
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

```java
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");
}
```

Enter a number: 5
Your number is positive
Cascading else-if vs. Stacked if

```c
if (condition1) {
    ...
} else if (condition2) {
    ...
} else if (condition3) {
    ...
} else {
    ...
}
```

```c
if (condition1) {
    ...
}
if (condition2) {
    ...
}
if (condition3) {
    ...
}
...
Use cascading `else-if` statements if **only one** condition should be true. Exactly one code block gets executed.
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`

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

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

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

```java
if (num < 0) {
    ...
}
if (num < 10) {
    ...
}
if (num > 20) {
    ...
}
```
Cascading `else-if` vs. Stacked `if`

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

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

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.

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

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

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Example</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>!</td>
<td>not</td>
<td>!(2 == 3)</td>
<td>true</td>
</tr>
<tr>
<td>&amp;&amp;</td>
<td>and</td>
<td>(2 == 3) &amp;&amp; (-1 &lt; 5)</td>
<td>false</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>or</td>
</tr>
</tbody>
</table>
# Logical Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Example</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>!</td>
<td>not</td>
<td>!(2 == 3)</td>
<td>true</td>
</tr>
<tr>
<td>&amp;&amp;</td>
<td>and</td>
<td>(2 == 3) &amp;&amp; (-1 &lt; 5)</td>
<td>false</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>or</td>
</tr>
</tbody>
</table>

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
ture && false
false
Precedence Madness

Precedence: arithmetic > relational > logical

\[ 5 \times 7 \geq 3 + 5 \times (7 - 1) \land 7 \leq 11 \]
Precedence Madness

Precedence: arithmetic > relational > logical

\[
5 \times 7 \geq 3 + 5 \times (7 - 1) \land \land 7 \leq 11
\]

\[
5 \times 7 \geq 3 + 5 \times 6 \land \land 7 \leq 11
\]

\[
35 \geq 3 + 30 \land \land 7 \leq 11
\]

\[
35 \geq 33 \land \land 7 \leq 11
\]

true \land \land true

true
// Store expressions that evaluate to true/false

boolean x = 1 < 2;  // true
boolean y = 5.0 == 4.0;  // false
// 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!
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 (int i = 0; i < numTimes; i++) {
    statement;
    statement;
}

Repeats the statements in the body numTimes.
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.

```java
for (int i = 0; i < 3; i++) {
    println("I love CS106A!");
}
```
for (int i = 0; i < 3; i++) {
    println("I love CS106A!" anch 1);
}

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

* Not exactly identical, but similar conceptually.
for (int i = 0; i < 3; i++) {
    println("I love CS106A!");
}
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!");
}

i

0
For Loop Redux

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

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

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

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

I love CS106A!
For Loop Redux

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

I love CS106A!
I love CS106A!
For Loop Redux

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

```
I love CS106A!
I love CS106A!
```
For Loop Redux

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

I love CS106A!
I love CS106A!
for (int i = 0; i < 3; i++) {
    println("I love CS106A!");
}
for (int i = 0; i < 3; i++)
{
    println("I love CS106A!");
}
for (int i = 0; i < 3; i++) {
    println("I love CS106A!"};
}
for (int i = 0; i < 3; i++) {
    println("I love CS106A!");
}
You Can Use the For Loop Variable
How would you print the first 100 even numbers?
// prints the first 100 even numbers
for (int i = 0; i < 100; i++) {
    println(i * 2);
}
// 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);
// Launch countdown
for (int i = 10; i >= 1; i--) {
    println(i);
}
println("Blast off!");

Output:

10
9
8
...
1
Blast off!
// Launch countdown
for (int i = 10; i >= 1; i--) {
    println(i);
}
println("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