CS 106A, Lecture 5
Booleans, Control Flow and Scope

suggested reading:
Java Ch. 3.4-4.6
Plan For Today

• Announcements
• Recap: Java, Variables and Expressions
• Aside: Shorthand Operators + Constants
• Revisiting Control Flow
  – If and While
  – For
Plan For Today

• Announcements
• Recap: Java, Variables and Expressions
• Aside: Shorthand Operators + Constants
• Revisiting Control Flow
  – If and While
  – For
Plan For Today

• Announcements
• Recap: Java, Variables and Expressions
• Aside: Shorthand Operators + Constants
• Revisiting Control Flow
  – If and While
  – For
Java

Program

Karel Program

SuperKarel Program

Console Program

Graphics Program
import acm.program.*;

public class Name extends ConsoleProgram {
    public void run() {
        statements;
    }
}

• Unlike Karel, many programs produce their behavior as text.
• console: Text box into which the behavior is displayed.
  – output: Messages displayed by the program.
  – input: Data read by the program that the user types.
println

• A statement that prints a line of output on the console, and goes to the next line.
  – pronounced "print-linn"

• Two ways to use println:
  
  • println("text");
    • Prints the given message as output, and goes to the next line.
    • A message is called a string; it starts/ends with a " quote character.
    • The quotes do not appear in the output.
    • A string may not contain a " character.

  • println();
    Prints a blank line of output.
public class HelloWorld extends ConsoleProgram {
    public void run() {
        print("Hello, ");
        print("world!");
    }
}

Same as println, but does not go to the next line.
Expressions

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

+ Addition  * Multiplication
– Subtraction  / Division
% Remainder
Precedence

• **precedence**: Order in which operators are evaluated.
  
  – Generally operators evaluate left-to-right.
    
    \[ 1 \ - \ 2 \ - \ 3 \ \text{is} \ (1 \ - \ 2) \ - \ 3 \ \text{which is} \ -4 \]
    
  – But \( \ast \ / \ \% \) have a higher level of precedence than \(+ \ - \)
    
    \[ 1 \ + \ 3 \ * \ 4 \ \text{is} \ 13 \]
    
    \[ 6 \ + \ 8 \ / \ 2 \ * \ 3 \]
    \[ 6 \ + \ 4 \ * \ 3 \]
    \[ 6 \ + \ 12 \ \text{is} \ 18 \]
    
  – Parentheses can alter order of evaluation, but spacing does not:
    
    \[ (1 \ + \ 3) \ * \ 4 \ \text{is} \ 16 \]
    
    \[ 1+3 \ * \ 4-2 \ \text{is} \ 11 \]
Type Interactions

\textbf{int} and \textbf{int} results in an \textbf{int}
\textbf{double} and \textbf{double} results in a \textbf{double}
\textbf{int} and \textbf{double} results in a \textbf{double}
\textbf{String} and \textbf{int} results in a \textbf{String}

etc.

* The general rule is: operations always return the most expressive type
Integer division

• When we divide integers, the quotient is also an integer.
  
  \[ \frac{14}{4} \text{ is } 3, \text{ not } 3.5. \]  
  \text{(Java ALWAYS rounds down.)}

\[ \begin{array}{c}
  4 ) 14 \\
  12 \\
  \hline
  2
\end{array} \quad \begin{array}{c}
  10 ) 45 \\
  40 \\
  \hline
  5
\end{array} \quad \begin{array}{c}
  27 ) 1425 \\
  135 \\
  \hline
  75 \\
  54 \\
  \hline
  21
\end{array} \]

• More examples:
  
  – \[ \frac{32}{5} \] is 6
  – \[ \frac{84}{10} \] is 8
  – \[ \frac{156}{100} \] is 1
  
  – Dividing by 0 causes an error when your program runs.
Practice

1 / 2 = 0
1.0 / 2 = 0.5
1 + 2 / 3 = 1
"abc" + (4 + 2) = "abc6"
"abc" + 4 + 2 = "abc42"
Making a new Variable

int myVariable;

- type: int
- name: myVariable
Variable Types

int  – an integer number

double  – a decimal number
myVariable = 2;

Existing variable name

value
• **assignment**: Stores a value into a variable.
  – The value can be an expression; the variable stores its result.

• Syntax:

\[ \text{name} = \text{expression}; \]

```c
int zipcode;
zipcode = 90210;

double myGPA;
myGPA = 1.0 + 2.25;
```
Declare / initialize

• A variable can be declared/initialized in one statement.
  – This is probably the most commonly used declaration syntax.

• Syntax:

  \[ \text{type name} = \text{expression}; \]

  \[
  \text{double tempF} = 98.6; \quad \text{tempF} \quad 98.6
  \]

  \[
  \text{int x} = (12 / 2) + 3; \quad \text{x} \quad 9
  \]
// Asks the user for an integer by
// displaying the given message
// and stores it in the variable 'a'
int a = readInt(message);

// Asks the user for a double by
// displaying the given message and
// stores it in the variable 'b'
double b = readDouble(message);
Practice: Receipt Program

• We wrote a ConsoleProgram called Receipt that calculates the tax, tip and total bill for us at a restaurant.
• The program asks the user for the subtotal, and then calculate and print out the tax, tip and total.

What was the meal cost? $45.50
Tax: $3.64
Tip: $9.1
Total: $58.24
Plan For Today

• Announcements
• Recap: Variables and Expressions
• Aside: Shorthand Operators + Constants
• Revisiting Control Flow
  – If and While
  – For
### Shorthand Operators

<table>
<thead>
<tr>
<th>Shorthand</th>
<th>Equivalent longer version</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>variable += value;</code></td>
<td><code>variable = variable + value;</code></td>
</tr>
<tr>
<td><code>variable -= value;</code></td>
<td><code>variable = variable - value;</code></td>
</tr>
<tr>
<td><code>variable *= value;</code></td>
<td><code>variable = variable * value;</code></td>
</tr>
<tr>
<td><code>variable /= value;</code></td>
<td><code>variable = variable / value;</code></td>
</tr>
<tr>
<td><code>variable %= value;</code></td>
<td><code>variable = variable % value;</code></td>
</tr>
<tr>
<td><code>variable++;</code></td>
<td><code>variable = variable + 1;</code></td>
</tr>
<tr>
<td><code>variable--;</code></td>
<td><code>variable = variable - 1;</code></td>
</tr>
</tbody>
</table>

```plaintext
x += 3;          // x = x + 3;
number *= 2;      // number = number * 2;
x++;              // x = x + 1;
```
Constants

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

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

- Syntax:

  ```java
  private static final type name = value;
  ```

  - name is usually in ALL_UPPER_CASE

  - Examples:
    ```java
    private static final int DAYS_IN_WEEK = 7;
    private static final double INTEREST_RATE = 3.5;
    private static final int SSN = 658234569;
    ```
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 {
    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);
    }
}
Plan For Today

• Announcements
• Recap: Variables and Expressions
• Aside: Shorthand Operators + Constants

• Revisiting Control Flow
  – If and While
  – For
If/Else in Karel

```java
if (condition) {
    statement;
    statement;
    statement;
    ...
} else {
    statement;
    statement;
    statement;
    ...
}
```

Runs the first group of statements if `condition` is true; otherwise, runs the second group of statements.
While Loops in Karel

```java
while (condition) {
    statement;
    statement;
    statement;
    ...
}
```

Repeats the statements in the body until `condition` is no longer true. Each time, Karel executes all statements, and then checks the condition.
while(frontIsClear()) {
    body
}

if(beepersPresent()) {
    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!");
}

int num = readInt("Enter a number: ");
if (num == 0) {
    println("That number is 0!");
} else {
    println("That number is not 0.");
}
```
Practice: 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
```
// 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);
```
// Solution #2 (ok, but #1 is better)
// harder to see loop end condition here

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

In order of precedence:

<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

```
// 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

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

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

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

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

true \land true

ture
// Store expressions that evaluate to true/false

```java
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;
// 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: GuessMyNumber

- 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.

```text
I am thinking of a number between 0 and 99...
Enter your guess: 22
Your guess is too low.
Enter your guess: 32
Your guess is too low.
Enter your guess: 56
Your guess is too high.
Enter your guess: 50
Your guess is too high.
Enter your guess: 46
Your guess is too high.
Enter your guess: 41
Your guess is too low.
Enter your guess: 42
You got it! The secret number was 42
```
The condition should be a **boolean** which is either **true** or **false**
if (condition1) {
    ...
} else if (condition2) { // NEW
    ...
} else {
    ...
}

Runs the first group of statements if \textit{condition1} is true; otherwise, runs the second group of statements if \textit{condition2} is true; otherwise, runs the third group of statements.

You can have multiple else if clauses together.
int num = readInt("Enter a number: ");

if (num > 0) {
    println("Your number is positive");
} else if (num < 0) {
    println("Your number is negative");
} else {
    println("Your number is 0");
}
Plan For Today

- Announcements
- Recap: Variables and Expressions
- Aside: Shorthand Operators + Constants
- Revisiting Control Flow
  - If and While
  - For
for (int i = 0; i < max; i++) {
    statement;
    statement;
    statement;
    ...
}

Repeats the statements in the body \textit{max} times.
For Loops in Java

```java
for (int i = 0; i < 3; i++) {
    println("I love CS 106A!");
}
```

- **for**: The start of the for loop.
- **(int i = 0; i < 3; i++)**: The loop condition and increment.
- **println("I love CS 106A!");**: The body of the loop.

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 Loops in Java

```java
for (int i = 0; i < 3; i++) {
    println("I love CS 106A!");
}
```
For Loops in Java

```
for (int i = 0; i < 3; i++) {
    println("I love CS 106A!");
}
```

i  0

```
for (int i = 0; i < 3; i++) {
    println("I love CS 106A!");
}
```
For Loops in Java

```
for (int i = 0; i < 3; i++) {
    println("I love CS 106A!");
}
```
For Loops in Java

\[\text{i } 0\]

\[
\text{for (int i = 0; i < 3; i++) \{ \\
\text{println("I love CS 106A!");} \\
\}}
\]

I love CS 106A!
For Loops in Java

```
for (int i = 0; i < 3; i++) {
    println("I love CS 106A!");
}
```

I love CS 106A!
For Loops in Java

```java
for (int i = 0; i < 3; i++) {
    println("I love CS 106A!");
}
```

```
I love CS 106A!
```
For Loops in Java

```java
for (int i = 0; i < 3; i++) {
    println("I love CS 106A!");
}
```

I love CS 106A!
For Loops in Java

```
for (int i = 0; i < 3; i++) {
    println("I love CS 106A!");
}
```

```
I love CS 106A!
I love CS 106A!
```
for (int i = 0; i < 3; i++) {
    println("I love CS 106A!");
}
For Loops in Java

```java
for (int i = 0; i < 3; i++) {
    println("I love CS 106A!");
}
```

```
I love CS 106A!
I love CS 106A!
```
For Loops in Java

```
for (int i = 0; i < 3; i++) {
    println("I love CS 106A!");
}
```
For Loops in Java

```java
for (int i = 0; i < 3; i++) {
    println("I love CS 106A!");
}
```

```
I love CS 106A!
I love CS 106A!
I love CS 106A!
```
For Loops in Java

```java
for (int i = 0; i < 3; i++) {
    println("I love CS 106A!");
}
```

![Image showing the output of the loop]

I love CS 106A!
I love CS 106A!
I love CS 106A!
for (int i = 0; i < 3; i++) {
    println("I love CS 106A!");
}
for (int i = 0; i < 3; i++) {
    println("I love CS 106A!");
}

I love CS 106A!
I love CS 106A!
I love CS 106A!
Using the For Loop Variable

```java
// prints the first 100 even numbers
for (int i = 0; i < 100; i++) {
    println(i * 2);
}
```
// Launch countdown
for (int i = 10; i >= 1; i--) {
    println(i * 2);
}
println("Blast off!");

Output:

10
9
8
...
Blast off!
// Adds up the first 100 numbers
int sum = 0;
for(int i = 0; i < 100; i++) {
    sum += i;
}
println("The sum is " + sum);
Recap

- Announcements
- Recap: Variables and Expressions
- Aside: Shorthand Operators + Constants
- Revisiting Control Flow
  - If and While
  - For

Next time: More control flow, methods in Java