Control Statements in Java
Announcements

• Assignment 1 (Karel) due at 3:15PM today.
  • You can use a late period and submit on Wednesday of next week by 3:15PM.
  • *It's okay to use a late period on the Karel assignment - this is your first time programming!*

• Email assignment due Sunday.
  • Looking forward to meeting you!

• Assignment 2 (*Welcome to Java!* ) goes out, is due on Monday, January 26 at 3:15PM.
  • Play around with graphics, control structures, and methods!
  • Some of these program require the use of methods. We'll cover methods today and at the start of Wednesday's lecture.
Announcements

• Continuing a longstanding tradition, Eric Roberts will be showing a video of the “I Have a Dream” speech on Monday in Gates B12 at 2:15PM.

• Highly recommended, especially if you haven't seen it before.
Outline for Today

• **The if Statement Revisited**
  - Now with variables!

• **The for Loop Revisited**
  - Now with graphical goodies!

• **Methods and Parameters**
  - Customizing the behavior of your methods.
Control Structures
Control Structures

- When using Karel, we used these three control structures:
  - if statements.
  - for loops.
  - while loops.
- These exist in standard Java as well!
- Let's see what they look like.
Control Structures

if
for
while
Control Structures

if
class
for
while
if statements

```java
if (condition) {
    ... statements to run if condition holds ...
}
```
Boolean Expressions

• A **boolean expression** is a test for a condition (it is either **true** or **false**).

• Value comparisons:

  
  
  \[
  \begin{align*}
  &= & \text{“equals”} & \text{(note: not single =)} \\
  \neq & \text{“not equals”} \\
  > & \text{“greater than”} \\
  < & \text{“less than”} \\
  \geq & \text{“greater than or equal to”} \\
  \leq & \text{“less than or equal to”}
  \end{align*}
  \]
Logical Operators

- We use **logical operators** combine or modify boolean values.
- Logical **NOT**: \(!p\)
  
  ```java
  if (!isWeekday()) {
      relaxAndUnwind();
  }
  ```

- Logical **AND**: \(p \&\& q\)
  
  ```java
  if (youreHappy() \&\& youKnowIt()) {
      clapYourHands();
  }
  ```

- Logical **OR**: \(p || q\)  *(inclusive OR)*
  
  ```java
  if (hasPuppy() || hasKitty()) {
      beHappy();
  }
  ```

- Order of precedence given above.
Or else

```java
if (condition) {
    ... statements to run if condition holds ...
} else {
    ... statements to run if condition doesn't hold ...
}
```
Cascading if

```java
if (score >= 90) {
    println(" AWWW YEAHHHHH ");
} else if (score >= 80) {
    println(" <(^_^)> ");
} else if (score >= 70) {
    println(" : - | ");
} else if (score >= 60) {
    println(" ocracy ");
} else {
    println(" (╯°□°)╯︵ ┻━┻ ");
}
```

Based on slides by Mehran Sahami
Control Statements

if
for
while
Control Statements

if
for
while
The Syntax

• As with Karel, to repeat a set of commands \( N \) times, use the following code:

```java
for (int i = 0; i < N; i++) {
    // … statements to execute …
}
```

• We'll talk about how exactly this works next time. For now, let's focus on what we can do with it!
Accessing the Counter

- Inside a for loop, the variable $i$ keeps track of the index of the current loop, starting at 0.
  - First time through the loop: $i = 0$
  - Second time through the loop: $i = 1$
  - Third time through the loop: $i = 2$
- Let's see an example of this.
Accessing the Counter

• Suppose we want to print out the first fifteen multiples of 50 (0, 50, 100, ...).

• We can accomplish this using a for loop.

```java
for (int i = 0; i < 15; i++) {
    println(i * 50);
}
```

• Do you see why?
Accessing the Counter

- Suppose we want to draw a row of boxes, like these:

```
   │   │   │   │   │
```

- Suppose each box is 50 pixels wide and 50 pixels tall.
- Look where their corners are... seem familiar?
Accessing the Counter

- Suppose we want to draw a row of boxes, like these:

  \[(0, 0) \quad (50, 0) \quad (100, 0) \quad (150, 0) \quad (200, 0) \quad (250, 0)\]

- Suppose each box is 50 pixels wide and 50 pixels tall.
- Look where their corners are... seem familiar?
Double For Loops

- You can put `for` loops inside of `for` loops! This is sometimes called a **double for loop**.
- Syntax:

```java
for (int i = 0; i < M; i++) {
    for (int j = 0; j < N; j++) {
        // ... statements to execute ...
    }
}
```
- This will run through all possible combinations of $i$ and $j$ where $i$ is less than $M$ and $j$ is less than $N$. 
Double For Loops

- Double for loops arise frequently when working with graphics.
- Suppose we want to draw this grid of boxes, each of which is $50 \times 50$:

```
+-----+-----+-----+-----+-----+
|     |     |     |     |     |
+-----+-----+-----+-----+-----+
|     |     |     |     |     |
+-----+-----+-----+-----+-----+
|     |     |     |     |     |
+-----+-----+-----+-----+-----+
|     |     |     |     |     |
+-----+-----+-----+-----+-----+
```

- Notice anything about the corner positions?
Drawing a Checkerboard, Java Style
<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
</tr>
</tbody>
</table>
Methods Revisited
Each point $k$ is connected to point $k + 2$, after wrapping around.
Each point $k$ is connected to point $k + 2$, after wrapping around.

Point $k$ is at $\frac{k}{\text{numSides}} \times 360^\circ$
Each point $k$ is connected to point $k + 2$, after wrapping around.

Point $k$ is at $\frac{k}{\text{numSides}} \times 360^\circ$
Each point \( k \) is connected to point \( k + 2 \), after wrapping around.

Point \( k \) is at \( \frac{k}{\text{numSides}} \times 360^\circ \)

\[ (x + r \cos \theta, y - r \sin \theta) \]

Point 1 (51.43°)
Passing Parameters

• A method can accept *parameters* when it is called.

• Syntax:

  ```java
  private void name(parameters) {
      /* ... method body ... */
  }
  ```

• The values of the parameters inside the method are set when the method is called.

• The values of the parameters can vary between calls.