For Loops

15-110 Summer 2010
Margaret Reid-Miller
The for Loop

• Another loop statement, for, is best for when you can determine in advance how many times you need to execute the loop (counting loop).

• The for statement includes the three parts needed for loops: initialize, test, and update.
  • All this information is conveniently placed at the beginning of the loop.

• All three loop statements (while, do, and for) are functionally equivalent.
The for statement

- The form of the `for` statement is
  
  ```
  for (<initialize>; <boolean_expression>; <update>)
  <statement>
  ```

- First, the `initialize` statement is executed.
- If `boolean_expression` evaluates to `true`, then `statement` (body of loop) is executed, followed by the `update` statement.
- The loop repeats until the `boolean_expression` evaluates to `false`.

The for statement

• The form of the for statement is
  
  ```
  for (<initialize>; <boolean_expression>; <update>)
  <statement>
  ```

• It is equivalent to

  ```
  <initialize>;
  while (<boolean_expression>) {
    <statement>
    <update>;
  }
  ```

  executed after statement (body of loop)
The for Flowchart

- **initialize**
- **boolean_expression**
  - **false**
  - **true**
- **update**
- **statement**
  - (body of loop)
A for Loop Example

```java
int sum = 0;
for (int i = 1; i <= n; i++) {
    sum += i*i;
}
System.out.println(sum);
```

Which variable is the loop control variable?

<table>
<thead>
<tr>
<th>sum</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>14</td>
<td>4</td>
</tr>
<tr>
<td>30</td>
<td>5</td>
</tr>
</tbody>
</table>
Another for Loop Example

```java
int sum = 0;
for (int i = 1; i <= n; i+=3) {
    sum += i;
}
System.out.println(sum);
```

<table>
<thead>
<tr>
<th>sum</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
</tr>
</tbody>
</table>

n = 11
Scope

• The _scope_ of a variable is the area within a program that can reference the variable.
• The scope depends on where the variable is declared.

```java
int sum = 0;
for (int i = 1; i <= n; i++) {
    sum += i*i;
}
System.out.println(sum);
```

Scope of variable _i_
**Scope**

```java
int sum = 0;
int i;
for (i = 1; i <= n; i++) {
    sum += i*i;
}
System.out.println("Sum of first "+ (i-1) + " integers squared is " + sum);
```
Nested Loops

- A loop can have another loop inside of it.
- For each iteration of the outside loop, the inside loop runs completely.
- Often it is easiest to read from the inside out.
- Example:

  How many lines are printed?

```java
for (int i = 1; i <= 5; i++) {
    for (int j = 1; j <= 3; j++) {
        System.out.println(i + " " + j);
    }
}
```

What happens if we write `println(i + j)`?
Palindromes

- A palindrome is a word, phrase, or sequence that reads the same backwards as forwards.

- Example: Bob by Weird Al Yankovic
  (A parody of Bob Dylan's Subterranean Homesick Blues)

  http://www.youtube.com/watch?v=Nej4xJe4Tdg

How would you test whether a string is a palindrome?
Which Loops?

• *for* loops are more natural when we know how many iterations we need (*definite* or *counting* loops).
  
  **Examples:**
  
  • Print "*" 10 times
  • Print the even numbers between 10 and the value of n

• *while* and *do* loops are more natural when we want to keep looping until some outcome (*indefinite* or *result controlled* loops).
  
  **Examples:**
  
  • Prompt the user *until* the user inputs the data in the correct form.
  • Continue looping *until* we reached a million dollars.