Announcements

• Assignment 01
  • Due Thursday 09/15 (tonight)

• Assignment 02
  • Due Thursday 09/22

• Office hours:
  • Adam’s: 10:30-11:30am on Wednesdays
Agenda

• Announcements
• More While Loops
• For Loops
Exercise

Suppose we wanted to ask the user for 6 numbers (int) and output their sum?
Loops

• Easy way to repeat some computation

• Two kinds of loops:
  • While
  • For

• Loops repeat block of code until the condition becomes false
Example: While Loop

```java
int val = 0;
String valStr = "";
int sum = 0;

int count = 0;
while (count < 6) {
    System.out.print("Enter a number: ");
    valStr = System.console().readLine();
    val = Integer.parseInt(valStr);
    sum = sum + val;
    count = count + 1;
}
System.out.println("The sum is "+sum);
```
Tracing Loops

```c
int sum = 1;
int count = 0;
while (count < 3) {
    sum = sum + 2;
    count = count + 1;
}
```

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Count &lt; 6</th>
<th>count</th>
<th>sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>T</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>T</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>T</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>T</td>
<td>3</td>
<td>7</td>
</tr>
</tbody>
</table>
Exercise: Tracing loops

```c
int sum = 10;
int count = 0;
while (count < 6) {
    sum = sum - 1;
    count = count + 2;
}
```

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Count &lt; 6</th>
<th>count</th>
<th>sum</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Exercise: Tracing loops

```c
int sum = 10;
int count = 0;
while (count < 6) {
    sum = sum - 1;
    count = count + 2;
}
```

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Count &lt; 6</th>
<th>count</th>
<th>sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>T</td>
<td>0</td>
<td>10</td>
</tr>
</tbody>
</table>
Exercise: Tracing loops

```c
int sum = 10;
int count = 0;
while (count < 6) {
    sum = sum - 1;
    count = count + 2;
}
```

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Count &lt; 6</th>
<th>count</th>
<th>sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>T</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>T</td>
<td>2</td>
<td>9</td>
</tr>
</tbody>
</table>

Exercise: Tracing loops

```c
int sum = 10;
int count = 0;
while (count < 6) {
    sum = sum - 1;
    count = count + 2;
}
```

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Count &lt; 6</th>
<th>count</th>
<th>sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>T</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>T</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>T</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>T</td>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>
Exercise: Tracing loops

```c
int sum = 10;
int count = 0;
while (count < 6) {
    sum = sum - 1;
    count = count + 2;
}
```

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Count &lt; 6</th>
<th>count</th>
<th>sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>T</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>T</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>T</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>T</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>F</td>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>
Accumulator pattern

Idea: Repeatedly update a variable (typically in a loop)

Pattern:
1. Initialize accumulator variable
2. Loop until done
   1. Update the accumulator variable
Convenience syntax: Assignment

Because updating variable values is so common, language such as Java provide shorthand syntax for it
  • Analogy: contractions in English

sum = sum + 2
count = count + 1
count = count – 1
product = product * 2
divisor = divisor / 2
message = message + “lol!”
Convenience syntax: Assignment

Because updating variable values is so common, language such as Java provide shorthand syntax for it

- Analogy: contractions in English

<table>
<thead>
<tr>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>sum = sum + 2</td>
</tr>
<tr>
<td>count = count + 1</td>
</tr>
<tr>
<td>count = count - 1</td>
</tr>
<tr>
<td>product = product * 2</td>
</tr>
<tr>
<td>divisor = divisor / 2</td>
</tr>
<tr>
<td>message = message + “ lol”</td>
</tr>
</tbody>
</table>
Convenience syntax: Assignment

Because updating variable values is so common, language such as Java provide shorthand syntax for it

- Analogy: contractions in English

<table>
<thead>
<tr>
<th>Variable Assignment</th>
<th>Shorthand Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sum = sum + 2</code></td>
<td><code>sum += 2</code></td>
</tr>
<tr>
<td><code>count = count + 1</code></td>
<td><code>count += 1</code></td>
</tr>
<tr>
<td><code>count = count - 1</code></td>
<td><code>count -= 1</code></td>
</tr>
<tr>
<td><code>product = product * 2</code></td>
<td><code>product *= 2</code></td>
</tr>
<tr>
<td><code>divisor = divisor / 2</code></td>
<td><code>divisor /= 2</code></td>
</tr>
<tr>
<td><code>message = message + “ lol”</code></td>
<td><code>message += “ lol”</code></td>
</tr>
</tbody>
</table>
Exercise: Write a program that computes powers of 2

Write a program, LoopPow2.java, that computes powers of twos. For example,

```
$ java LoopPow2
Enter an exponent: 0
2 to the power of 0 is 1

$ java LoopPow
Enter an exponent: 1
2 to the power of 1 is 2

$ java LoopPow
Enter an exponent: 4
2 to the power of 4 is 16
```
Agenda

- Announcements
- More While Loops
- For Loops
Example: For Loop

```java
int val = 0;
String valStr = "";
int sum = 0;

for (int count = 0; count < 6; count = count + 1) {
    System.out.print("Enter a number: ");
    valStr = System.console().readLine();
    val = Integer.parseInt(valStr);
    sum = sum + val;
}
System.out.println("The sum is "+sum);
```
Example: For Loop

```
for (int count = 0; count < 6; count = count + 1) {

}
```
Exercise: Tracing loops

String pattern = "";
for (int i = 0; i < 3; i++) {
    pattern = pattern + "*";
}
System.out.println(pattern);
Exercise: Tracing loops

```java
String pattern = "";
for (int i = 0; i < 3; i++) {
    pattern = pattern + "*";
}
System.out.println(pattern);
```

<table>
<thead>
<tr>
<th>Iteration</th>
<th>i &lt; 3</th>
<th>i</th>
<th>pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>T</td>
<td>0</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>1</td>
<td>T</td>
<td>1</td>
<td>&quot;*&quot;</td>
</tr>
<tr>
<td>2</td>
<td>T</td>
<td>2</td>
<td>&quot;**&quot;</td>
</tr>
<tr>
<td>3</td>
<td>F</td>
<td>3</td>
<td>&quot;***&quot;</td>
</tr>
</tbody>
</table>
```
Exercise: LoopPattern.java

```
$ java LoopPattern
Enter a length: 5
*_*_*_*

$ java LoopPattern
Enter a length: 10
*_*_*_*_*_*_*_*_*

$ java LoopPattern
Enter a length: 0

$ java LoopPattern
Enter a length: 1
*
```
Exercise: Nested loops

$ java Square
Enter a size: 5
*****
*****
*****
*****
*****

$ java Square
Enter a size: 1
*

$ java Square
Enter a size: 0
Printf

- https://docs.oracle.com/javase/tutorial/java/data/numberformat.html

```java
printf(String format, Object... args)
```
Arrays

(next week)
Arrays

Idea: Store multiple values into a single variable

Values are sequential

Analogous to a list
Arrays

double val = 3.0;

double[] vals = {3.0, 6.0, 7.0, -2.5};
Arrays

```java
boolean[] flags = {true, false};

String[] greetings ={"hi", "hola", "ciao", "aloha"};
```
Array Indexing

Access individual elements of an array with indexing
array[index]

We use zero-based indexing
   first element is 0
   last element is length-1

Accessing indices out of range results in a runtime error!
Arrays

```java
int[] sequence = new int[10];
for (int i = 0; i < sequence.length; i++)
{
    sequence[i] = i+1;
}
```
Arrays

Three ways to initialize an array

1. With an initial value

2. With allocated space, but uninitialized

3. With an empty array reference