Loops
Program Structure: Code Blocks

Defined by opening and closing curly bracket (e.g., { & })

Can be nested
- innermost opening curly bracket matches innermost closing curly bracket
- can nest conditionals, loops

```java
/**
 * Our first program
 */
public class ExampleClass {
    public static void main(String[] args) {
        // Your code goes here!
    }
}
```
Scope and Variables

`scope` defines the region of code where a variable can be used.
Scope is defined by code blocks:
- Variables declared inside a code block are *local* to that block.
- Variables anywhere inside that block (even nested blocks!)
- Variables **cannot** be used outside that block.
Control Flow in Programs

Last week, how to make decisions about whether or not to execute code
This week, how to make decisions about whether to execute code again

Example

previously saw how we could use conditionals to calculate a single person’s age

loops will allow us to repeat that same code for multiple people
Why Loops?

Often want to repeat code zero or more times

Two options
- copy and paste code multiple times
- use a loop

What problems arise with the first option?
Loops

Loops allow us to repeat one or more statements while some boolean condition is true, and stop when the condition is false.

conditional statements

loops
Types of Loops

- **while loops**
- **for loops**
- **do-while loops**
Types of Loops

while loops

for loops

do-while loops
While the condition is true, execute the statements inside the loop

```java
while (<boolean expression>) {
    //code to execute if boolean expression is true
}
//code to execute after while loop
```
While Loops

While the condition is true, execute the statements inside the loop

```
int value = 5;
while (value < 8) {
    System.out.println(value);
    value++;
}
```

//code to execute after while loop

memory

value (int)

5
While Loops

While the condition is true, execute the statements inside the loop

```java
int value = 5;
while (value < 8) {
    System.out.println(value);
    value++;
}
//code to execute after while loop
```

memory

```
value (int)
5
```
While Loops

While the condition is true, execute the statements inside the loop

```java
int value = 5;
while (value < 8) {
    System.out.println(value);
    value++;
}
// code to execute after while loop
```

Memory:
- `value (int)`
  - `5`
While Loops

While the condition is true, execute the statements inside the loop

```java
int value = 5;
while (value < 8) {
    System.out.println(value);
    value++;
}
// code to execute after while loop
```
While Loops

While the condition is true, execute the statements inside the loop

```java
int value = 5;
while (value < 8) {
    System.out.println(value);
    value++;
}
//code to execute after while loop
```
While Loops

While the condition is true, execute the statements inside the loop

```java
int value = 5;
while (value < 8) {
    System.out.println(value);
    value++;
}
// code to execute after while loop
```

memory

<table>
<thead>
<tr>
<th>value (int)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
</tr>
</tbody>
</table>
While Loops

While the condition is true, execute the statements inside the loop

```java
int value = 5; // is this true?
while (value < 8) {
    System.out.println(value);
    value++;
}
// code to execute after while loop
```

Memory

<table>
<thead>
<tr>
<th>value (int)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
</tr>
</tbody>
</table>

5
While Loops

While the condition is true, execute the statements inside the loop

```java
int value = 5;
while (value < 8) {
    System.out.println(value);
    value++;
}
// code to execute after while loop
```
While Loops

While the condition is true, execute the statements inside the loop

```java
int value = 5;
while (value < 8) {
    System.out.println(value);
    value++;
}
// code to execute after while loop
```

```
memory

<table>
<thead>
<tr>
<th>value (int)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
</tr>
</tbody>
</table>
```

```text
true
```

```text
false
```
While Loops

While the condition is true, execute the statements inside the loop

```
int value = 5;
while (value < 8) {
    System.out.println(value);
    value++;  
}

//code to execute after while loop
```
While Loops

While the condition is true, execute the statements inside the loop

```java
int value = 5;
while (value < 8) {
    System.out.println(value);
    value++;
}
//code to execute after while loop
```

memory

<table>
<thead>
<tr>
<th>value (int)</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
</tr>
</tbody>
</table>
While Loops

While the condition is true, execute the statements inside the loop

```
int value = 5;
while (value < 8) {
    System.out.println(value);
    value++;
}
```

//code to execute after while loop
While Loops

While the condition is true, execute the statements inside the loop

```
int value = 5;
while (value < 8) {
    System.out.println(value);
    value++;
}
```

//code to execute after while loop
While Loops

While the condition is true, execute the statements inside the loop

```java
int value = 5;
while (value < 8) {
    System.out.println(value);
    value++;
}
```

//code to execute after while loop

```
true
false
```

```
memory
```

```
value (int)
7
```

```
5
6
```
**While Loops**

While the condition is true, execute the statements inside the loop.

```java
int value = 5;
while (value < 8) {
    System.out.println(value);
    value++;
}
// code to execute after while loop
```

Memory:
- `value (int) = 7`
While Loops

While the condition is true, execute the statements inside the loop

```java
int value = 5;
while (value < 8) {
    System.out.println(value);
    value++;
}
```

memory

<table>
<thead>
<tr>
<th>value (int)</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
</tr>
</tbody>
</table>

true

false

5

6

7
While Loops

While the condition is true, execute the statements inside the loop

```
int value = 5;
while (value < 8) {
    System.out.println(value);
    value++;  
}
//code to execute after while loop
```
While Loops

While the condition is true, execute the statements inside the loop

```java
int value = 5;
while (value < 8) {
    System.out.println(value);
    value++;
}
// code to execute after while loop
```

memory

<table>
<thead>
<tr>
<th>value (int)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
</tr>
</tbody>
</table>
While Loops

While the condition is true, execute the statements inside the loop

```java
int value = 5;  // is this true?
> while (value < 8) {
    System.out.println(value);
    value++;
}
// code to execute after while loop
```

memory

<table>
<thead>
<tr>
<th>value (int)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
</tr>
</tbody>
</table>

true

false
While Loops

While the condition is true, execute the statements inside the loop

```java
int value = 5;
while (value < 8) {
    System.out.println(value);
    value++;
}
```

//code to execute after while loop

```
5
6
7
```

```
memory

value (int)
8
```
Loops vs Conditionals

Conditionals decide whether or not to execute a block of code once.
Loops decide whether or not to execute a block of code multiple times.
Parts of a Loop

Every loop has four parts

```java
int value = 5;
while (value < 8) {
    System.out.println(value);
    value++;
}
//code to execute after while loop
```
Parts of a Loop

Every loop has four parts

```
int value = 5;
while
{
}
//code to execute after while loop
```

initialization

set up a variable that will control the loop
Parts of a Loop

Every loop has four parts

**initialization**
- set up a variable that will control the loop

**condition**
- a boolean expression to control when the loop stops

```java
int
while  value < 8
{
}
//code to execute after while loop
```
Parts of a Loop

```java
int
while
    
}
//code to execute after while loop
System.out.println(value);
```
Parts of a Loop

Every loop has four parts

- **initialization**: set up a variable that will control the loop
- **condition**: a boolean expression to control when the loop stops
- **work**: the code the loop will repeat
- **progress**: how the loop moves closer to termination

```cpp
int value = 0;
while (value < 10) {
    // code to execute
    value++;
}
// code to execute after while loop
```
Finite vs Infinite Loops

Usually, we want loops to stop at some point, resume with code after loop isn’t true for all applications
will always be true for this class!

*finite loops* are those that stop
*infinite loops* are those that repeat forever

will require you to manually terminate your program
Types of Loops

while loops

true
false

for loops

true
false

do-while loops

true
false
For Loops

Explicitly recognizes the four parts of a loop in a single structure

```plaintext
for (<var init>; <boolean expr>; <progress>) {
    //code to execute if boolean expression is true
}
//code to execute after while loop
```
For Loops

Explicitly recognizes the four parts of a loop in a single structure

```plaintext
for (<var init>; <boolean expr>; <progress>) {
    //code to execute if boolean expression is true
}
//code to execute after while loop
```
For Loops

Explicitly recognizes the four parts of a loop in a single structure

```plaintext
for (<var init>; <boolean expr>; <progress>) {
    //code to execute if boolean expression is true
}
//code to execute after while loop
```
Explicitly recognizes the four parts of a loop in a single structure

```plaintext
for (<var init>; <boolean expr>; <progress>) {
  //code to execute if boolean expression is true
  2 //code to execute after while loop
}
//code to execute after while loop
```
For Loops

Explicitly recognizes the four parts of a loop in a single structure

```
for (<var init>; <boolean expr>; <progress>) {
    //code to execute if boolean expression is true
}
//code to execute after while loop
```
For Loops

Explicitly recognizes the four parts of a loop in a single structure

```
for (<var init>; <boolean expr>; <progress>) {
  //code to execute if boolean expression is true
}
//code to execute after while loop
```
For Loops

Explicitly recognizes the four parts of a loop in a single structure

```plaintext
for (<var init>; <boolean expr>; <progress>) {
    //code to execute if boolean expression is true
}
//code to execute after while loop
```
For Loops

Explicitly recognizes the four parts of a loop in a single structure

for (<var init>; <boolean expr>; <progress>) {
    //code to execute if boolean expression is true
}
//code to execute after while loop
For Loops

Explicitly recognizes the four parts of a loop in a single structure

```
for (<var init>; <boolean expr>; <progress>) {
    //code to execute if boolean expression is true
}
//code to execute after while loop
```
For Loops

Explicitly recognizes the four parts of a loop in a single structure

```java
for (<var init>; <boolean expr>; <progress>) {
    //code to execute if boolean expression is true
}
//code to execute after while loop
```
For i = 5, print the value of i while i is less than 8…

```java
> for (int i = 5; i < 8; i++) {
    System.out.println(i);
}

// code to execute after while loop
```
For Loops

“For i = 5, print the value of i while i is less than 8…”

```java
> for int i = 5

// code to execute after while loop

i (int)
5
```
For Loops

“For i = 5, print the value of i while i is less than 8...”

```java
for (i = 5; i < 8; i++)
{
    // code to execute after while loop
}
```

// code to execute after while loop

memory

<table>
<thead>
<tr>
<th>i (int)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
</tr>
</tbody>
</table>
For Loops

“For i = 5, print the value of i while i is less than 8…”
For Loops

“For i = 5, print the value of i while i is less than 8...”

```java
for (int i = 5; i < 8; i++) {
    System.out.println(i);
}
```

//code to execute after while loop

memory

```
> System.out.println(i);
```

```
5
```

true

false
For Loops

“For i = 5, print the value of i while i is less than 8…”

```
for (int i = 5; i < 8; i++) {
    System.out.println(i);
}
```

//code to execute after while loop

memory

true

false

```java
int i = 5;
```
For Loops

“For i = 5, print the value of i while i is less than 8...”
For Loops

“For i = 5, print the value of i while i is less than 8...”

```c
for (int i = 5; i < 8; i++) {
  //code to execute after while loop
}
```

memory

```
<table>
<thead>
<tr>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
</tr>
</tbody>
</table>
```
"For i = 5, print the value of i while i is less than 8..."
For Loops

“For i = 5, print the value of i while i is less than 8…”

```java
for (int i = 5; i < 8; i++) {
    //code to execute after while loop
    System.out.println(i);
}
```

memory

```
<table>
<thead>
<tr>
<th>i (int)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
</tr>
</tbody>
</table>
```

true

false
For Loops

“For i = 5, print the value of i while i is less than 8…”

```java
for (int i = 5; i < 8; i++) {
    System.out.println(i);
}
```

//code to execute after while loop
For Loops

“For i = 5, print the value of i while i is less than 8...”

```java
for (int i = 5; i < 8; i++) {
    System.out.println(i);
}
```

//code to execute after while loop

<table>
<thead>
<tr>
<th>i (int)</th>
<th>memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>
For Loops

“For i = 5, print the value of i while i is less than 8…”
For Loops

“For i = 5, print the value of i while i is less than 8...”
For Loops

“For i = 5, print the value of i while i is less than 8…”

```
for i = 5, print the value of i while i is less than 8...

is this true?

memory

```

```
> for i < 8
}
//code to execute after while loop
```

```
5
6
```
“For i = 5, print the value of i while i is less than 8...”
For Loops

“For i = 5, print the value of i while i is less than 8…”

```java
for (int i = 5; i < 8; i++) {
    System.out.println(i);
}
```

-memory

```
<table>
<thead>
<tr>
<th>i (int)</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
</tr>
</tbody>
</table>
```

-memory

```
5
6
```
For Loops

“For i = 5, print the value of i while i is less than 8…”

```java
for (int i = 5; i < 8; i++) {
    System.out.println(i);
}
```

//code to execute after while loop

memory

```
<table>
<thead>
<tr>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
</tbody>
</table>
```
"For i = 5, print the value of i while i is less than 8..."
For Loops

“For i = 5, print the value of i while i is less than 8…”
For Loops

“For i = 5, print the value of i while i is less than 8…”
For Loops

“For i = 5, print the value of i while i is less than 8...”

```java
for (int i = 5; i < 8; i++) {
    System.out.println(i);
}
```
For Loop Notes

For loop variables are one of the few places where you can get away with single letter variable names

but, if you can come up with a sensible name, use it!

Why use the for loop?

for loops ensure you have all four parts of the loop there
easier to miss one or more with the while loop
For loops are used when we know how many times we want the loop to execute

While loops are used when we aren’t sure how many times we want the loop to execute

In reality, can use for or while loops interchangeably

...although it is often more natural to use one over the other in most cases

should be able to understand how both work!
Types of Loops

while loops

for loops

do while loops
Do While Loops

Similar to a while loop, but checks the condition last

do {
    //code to execute if boolean expression is true
} while (<boolean expr>);

//code to execute after while loop
Do While Loops

Similar to a while loop, but checks the condition last

```java
int value = 5;
>
do {
    System.out.println(value);
    value++;
}
while (value < 8);
//code to execute after while loop
```
Do While Loops

Similar to a while loop, but checks the condition last

```java
int value = 5;
do {
    System.out.println(value);
    value++;
} while (value < 8);

// code to execute after while loop
```
Do While Loops

Similar to a while loop, but checks the condition last

```java
int value = 5;
do {
    System.out.println(value);
    value++;
} while (value < 8);
//code to execute after while loop
```
Do While Loops

Similar to a while loop, but checks the condition last

```java
int value = 5;
do {
    System.out.println(value);
    value++;
} while (value < 8);
// code to execute after while loop
```
Do While Loops

Similar to a while loop, but checks the condition last

```java
int value = 5;

do {
    System.out.println(value);
    value++;
} while (value < 8);

//code to execute after while loop
```

```
memory

<table>
<thead>
<tr>
<th>value (int)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
</tr>
</tbody>
</table>
```

5
Do While Loops

Similar to a while loop, but checks the condition last

```
int value = 5;
> do {
    System.out.println(value);
    value++;
} while (value < 8);

// code to execute after while loop
```
Do While Loops

Similar to a while loop, but checks the condition last

```
int value = 5;
do {
    System.out.println(value);
    value++;
} while (value < 8);

// code to execute after while loop
```
Do While Loops

Similar to a while loop, but checks the condition last

```java
int value = 5;
do {
    System.out.println(value);
    value++;
} while (value < 8);
//code to execute after while loop
```

memory

<table>
<thead>
<tr>
<th>value (int)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
</tr>
</tbody>
</table>

5
6
Do While Loops

Similar to a while loop, but checks the condition last

```java
int value = 5;
do {
    System.out.println(value);
    value++;
} while (value < 8);

// code to execute after while loop
```

memory

<table>
<thead>
<tr>
<th>value (int)</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
</tr>
</tbody>
</table>

5
6

true

false
Do While Loops

Similar to a while loop, but checks the condition last

```
int value = 5;

do {
    System.out.println(value);
    value++;
} while (value < 8);

// code to execute after while loop
```
Do While Loops

Similar to a while loop, but checks the condition last

```java
int value = 5;
>
> do {
    System.out.println(value);
    value++;
>
} while (value < 8);
>
// code to execute after while loop
```

memory

```
value (int)

7
```

```
5

6
```
Do While Loops

Similar to a while loop, but checks the condition last

```
int value = 5;
do {
    System.out.println(value);
    value++;
} while (value < 8);
```

//code to execute after while loop

memory

```
value (int)
7
```

```
5
6
```
Do While Loops

Similar to a while loop, but checks the condition last

```java
int value = 5;
do {
    System.out.println(value);
    value++;
} while (value < 8);

// code to execute after while loop
```

memory

<table>
<thead>
<tr>
<th>value (int)</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
</tr>
</tbody>
</table>

| 5 |
| 6 |
| 7 |
Do While Loops

Similar to a while loop, but checks the condition last

```java
int value = 5;

do {
    System.out.println(value);
    value++;
} while (value < 8);

// code to execute after while loop
```

5 6 7

memory

value (int)

8
Do While Loops

Similar to a while loop, but checks the condition last

```
int value = 5;
do {
    System.out.println(value);
    value++;
} while (value < 8);

// code to execute after while loop
```
Why Do While Loops?

When we want to guarantee that our work is performed at least once
Primary use is for checking validity of user input
   we always want to ask for input once
   only if the input is invalid do we want to ask again

```java
int input;

do {
    System.out.print("Enter a " + "number [1-10] ");
    input = scan.nextInt();
} while (input < 1 || input > 10);

//code to execute after while loop
```