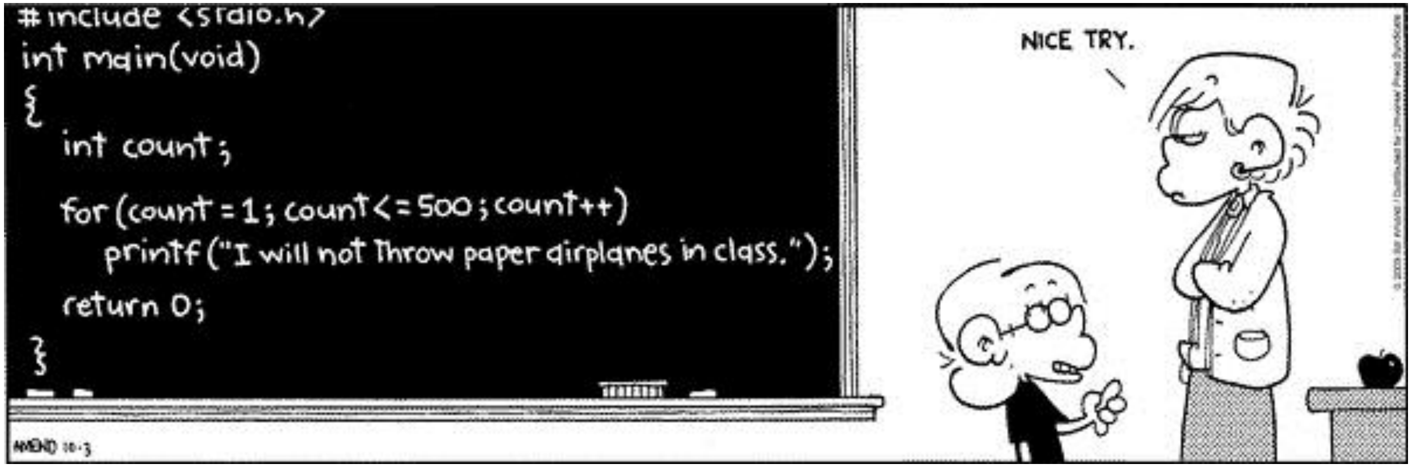
Loops



```
#include <stdio.h>
int main(void)
{
  int count;
  for (count = 1; count <= 500; count++)
     printf("I will not Throw paper airplanes in class.");
  return 0;
}
```

Flow of Control

- Normally, the flow of control is to go through code line-by-line.
- We’ve seen times when this changes:
  - A method is invoked
    - Control jumps to that method
  - Conditionals
    - Execute some code if true, other code if not
- Repetition statements (loops) also change the flow of control by repeating statements.

Repetition Statements

- Repetition statements (also called *loops*) allow you to execute a statement repeatedly.
- Loops are controlled by boolean expressions.
- Java has three kinds of loops:
  - while
  - do… while
  - for

WHILE LOOP

The while Loop

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

1. Check condition.
   a) If true:
      i. Execute Statement.
      ii. Return to Step 1.
   b) If false:
      i. Move on to the statement after the while loop.

The while Loop (continued)

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

- As long as *condition* is true, keep executing the code inside (statement).
- Be sure that somewhere inside the loop the *condition* is set to false.
  - Otherwise the loop never ends!

The while Loop (continued)

- The body of a while loop is executed zero or more times.
  - If the condition is false, the statement will never be executed.

The while Loop

- The condition can be a complex boolean statement.

```
while (condition1 && condition2) {
      statement;
}

while (condition1 || condition2) {
      statement;
}
```

Local Variables

- Variables declared inside the while loop are local only to that single iteration of the while loop.

```
while(condition) {
    int n=0;
    System.out.println(n);
    …
}
```

Example

- What is printed?

```
int count = 1;
while(count < 5) {
   System.out.println("Hello "+ count);
   count++;
}
```

Example

- What is printed?

```
int count = 1;
while(count <= 5) {
   System.out.println("Hello "+ count);
   count++;
}
```

Example

- What is printed?

```
int count = 1;
while(count <= 5) {
   System.out.println("Hello "+ count);
   count--;
}
```

Example

- What is printed?

```
int count = 1, sum = 0;
while(count <= 4) {
   sum += count;
   count++;
}
System.out.println(count);
System.out.println(sum);
```

Uses of the while Loop

- You don’t know how many times you want to repeat something ahead of time
- You want to read in multiple values from the user (who uses a “sentinel” value to indicate input is done).
- You want to read in values from the user continually until you get a valid value.

Practice

- Modify the math program to allow the user to repeatedly perform the operations.
- Write a text-based program to calculate a running sum of numbers entered by the user. Quit when the user enters 0.
  - Modify the program so the user can only enter numbers between 1 and 100.

THE DO… WHILE LOOP

The do… while Loop

```
do {
   statement;
} while (condition);
```

1. Execute statement.
2. Check condition.
   1. If true:
      1. Return to Step 1.
   2. If false:
      1. Move on to the statement after the do-while loop.

The do… while Loop (continued)

- The body of a do-while loop is executed **one** or more times.

- Which to use?
  - Do you want the condition evaluated *before* or *after* the code is executed?
  - Do you want the code executed always once or maybe executed never?

Practice

- Write a text-based slot machine in which three numbers between 0 and 9 are randomly selected and printed side by side. Print a “winning” statement if all three number are the same or if any two of the numbers are the same. Continue playing until the user chooses to stop.

THE FOR LOOP

The for Loop

```
for(initialization; condition; update) {
    statement;
}
```

1. Perform initialization.
2. Check condition.
   a) If true:
      i. Execute Statement.
      ii. Perform update.
      iii. Return to Step 2.
   b) If false:
      i. Move on to the statement after the for loop.

The for Loop

- The initialization section can be used to declare a variable.
  - Note that this variable is then *local* to the loop- it cannot be seen outside of the loop.
- The update section can perform any calculation.
- Each section of the loop is optional.

The for Loop (continued)

- A for loop is functionally equivalent to a while loop:

```
initialization;
while (condition) {
    statement;
    update;
}
```

The for Loop (continued)

- The condition of a for loop is tested first.
- The body of a for loop is executed zero or more times.
  - If the condition is false, the statement will never be executed.

Example

- What is printed?

```
for(int i=0; i<3; i++) {
   System.out.println("Hello "+ i);
}
```

The for Loop

```
for(int i=0; i<3; i++) {
  System.out.println("Hello" + i);
}
```

initialization happens first

i $\boxed{0}$

The for Loop

```
for(int i=0; i<3; i++) {
  System.out.println("Hello" + i);
}
```

then check the condition
(it’s true)

i $\boxed{0}$

The for Loop

```
for(int i=0; i<3; i++) {
  System.out.println("Hello" + i);
}
```

then execute the statement (print “Hello0”)

i $\boxed{0}$

The for Loop

```
for(int i=0; i<3; i++) {
  System.out.println("Hello" + i);
}
```

then execute the update

i 1

The for Loop

```
for(int i=0; i<3; i++) {
  System.out.println("Hello" + i);
}
```

then check the condition
(it’s true)

i 1

The for Loop

```
for(int i=0; i<3; i++) {
  System.out.println("Hello" + i);
}
```

then execute the statement
(print “Hello1”)

i $\boxed{1}$

The for Loop

```
for(int i=0; i<3; i++) {
  System.out.println("Hello" + i);
}
```

then execute the update

i $\boxed{2}$

The for Loop

```
for(int i=0; i<3; i++) {
  System.out.println("Hello" + i);
}
```

then check the condition
(it’s true)

i 2

The for Loop

```
for(int i=0; i<3; i++) {
  System.out.println("Hello" + i);
}
```

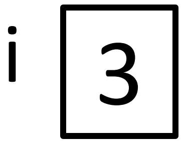
then execute the statement (print “Hello2”)

i 2

The for Loop

```
for(int i=0; i<3; i++) {
  System.out.println("Hello" + i);
}
```

then execute the update



The for Loop

```
for(int i=0; i<3; i++) {
  System.out.println("Hello" + i);
}
```

then check the condition
(it’s false- we’re done)

i 3

Example

- What is printed?

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

Example

- What is printed?

```
for(int i=3; i>0; i--) {
    System.out.println("Hello "+ i);
}
```

Example

- What is printed?

```
for(int i=0; i<=10; i=i*2) {
    System.out.println("Hello "+ i);
}
```

Uses of the for Loop

- Executing statements a pre-defined number of times

Local Variables

- Variables declared inside the for loop are local only to that single iteration of the while loop.

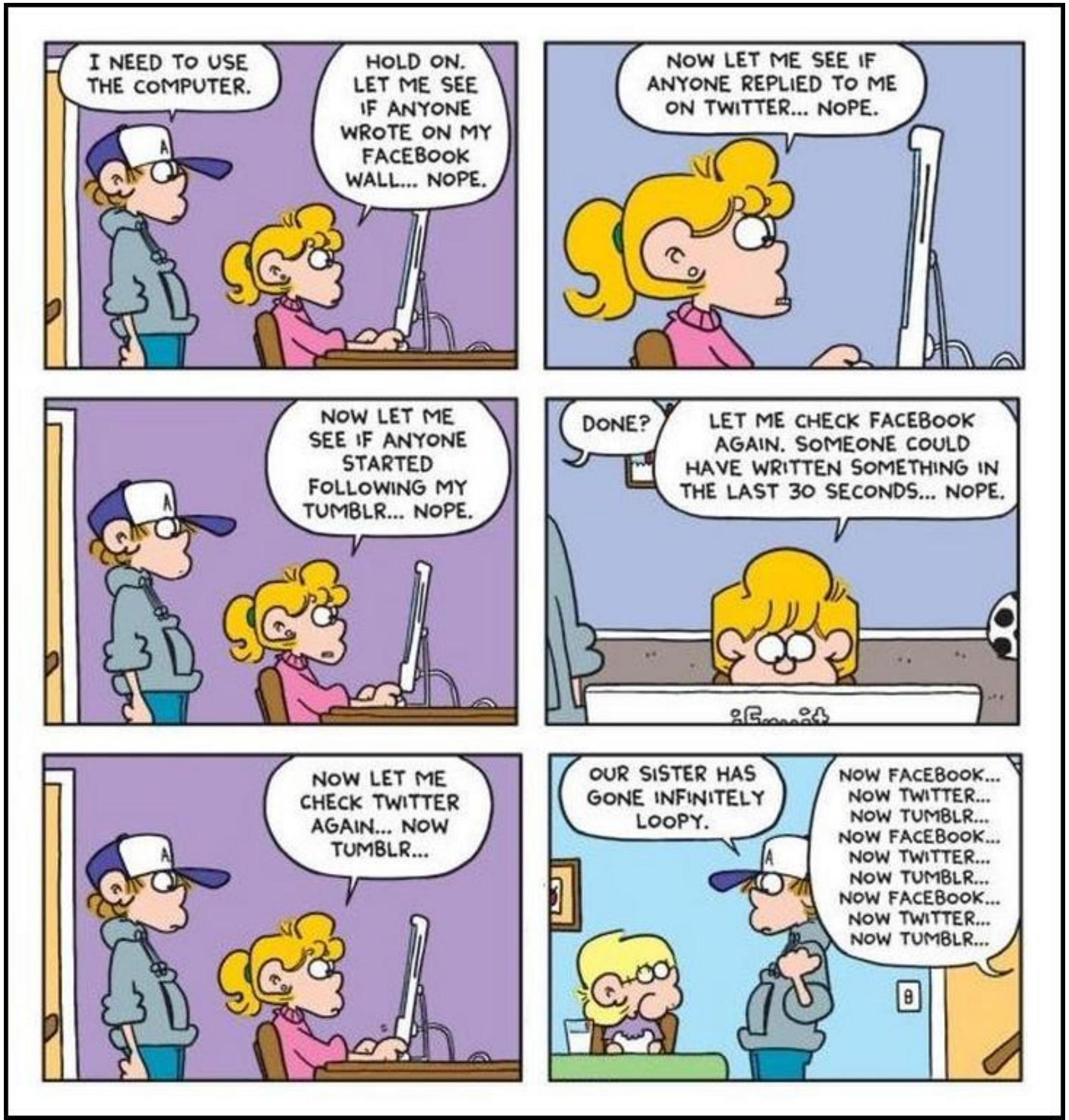
```
for(int i=0; i<n; i++) {
    System.out.println(i);
}
System.out.println(i); // ERROR!
```

Practice

- Modify the running sum program.
  - Sum up 10 numbers, instead of entering until the user enters 0.
  - Ask the user how many numbers they will enter.
- Write a text-based program to read in text from the user and echo the text back out in reverse.
- Write a text-based program to read in text from the user and print the number of each vowel and the number of non-vowels. (Use a switch!)

LOOPS

Beware of Infinite Loops!


I NEED TO USE THE COMPUTER.
HOLD ON. LET ME SEE IF ANYONE WROTE ON MY FACEBOOK WALL... NOPE.
NOW LET ME SEE IF ANYONE REPLIED TO ME ON TWITTER... NOPE.
NOW LET ME SEE IF ANYONE STARTED FOLLOWING MY TUMBLR... NOPE.
DONE?
LET ME CHECK FACEBOOK AGAIN. SOMEONE COULD HAVE WRITTEN SOMETHING IN THE LAST 30 SECONDS... NOPE.
NOW LET ME CHECK TWITTER AGAIN... NOW TUMBLR...
OUR SISTER HAS GONE INFINITELY LOOPY.
NOW FACEBOOK... NOW TWITTER... NOW TUMBLR... NOW FACEBOOK... NOW TWITTER... NOW TUMBLR... NOW FACEBOOK... NOW TWITTER... NOW TUMBLR...

Beware of Infinite Loops!

- The body of a while loop must eventually make a condition false.
- The update section of a for loop or the body of the for loop must eventually make the conditional false.
- Infinite loops are common logical errors.

When to use which?

- In general, use a for-loop if you know in advance how many times you want to repeat the code.
- In general use a while-loop if you want to keep repeating code until some condition changes (something becomes true/false), but you don’t know ahead of time when this will happen.

Nested Loops

- Loops can be nested such that the body of one loop contains another loop.
- For each iteration of the outer loop, the inner loop goes through a full execution.
  - Statements inside an inner loop will execute (outer * inner) times.

Example

- How many times is “Here” printed?

```
int count1 = 1;
while (count1 <= 10)
{
   int count2 = 1;
   while (count2 <= 20)
   {
      System.out.println ("Here");
      count2++;
   }
   count1++;
}
```

Example

- How many times is “Here” printed?

```
int count1 = 1;
while (count1 < 10)
{
   int count2 = 1;
   while (count2 < 20)
   {
      System.out.println ("Here");
      count2++;
   }
   count1++;
}
```

Example

- What is printed?

```
int count1 = 1;
while (count1 <= 2)
{
   int count2 = 1;
   while (count2 <= 5)
   {
         count2++;
   }
   count1++;
}
System.out.println(“outer loop executed “
+ count1 + “ times and inner executed “
+ count2 + “times.”);
```

Example

- How many times is “Here” printed?

```
for(int i=0; i<20; i++) {
   for(int j=0; j<40; j++) {
      System.out.println ("Here");
   }
}
```

Practice

- Write a Hi-Lo guessing game.