CMPSC 100

Computational Expression

Review: flow of control

- Java executes statements from top to bottom in the order the instructions are listed
  - Variables must be declared before they can be used
  - Variables can change values over time as program executes additional statements
- Unless altered by an `if` or `switch` statement, program executes all instructions once

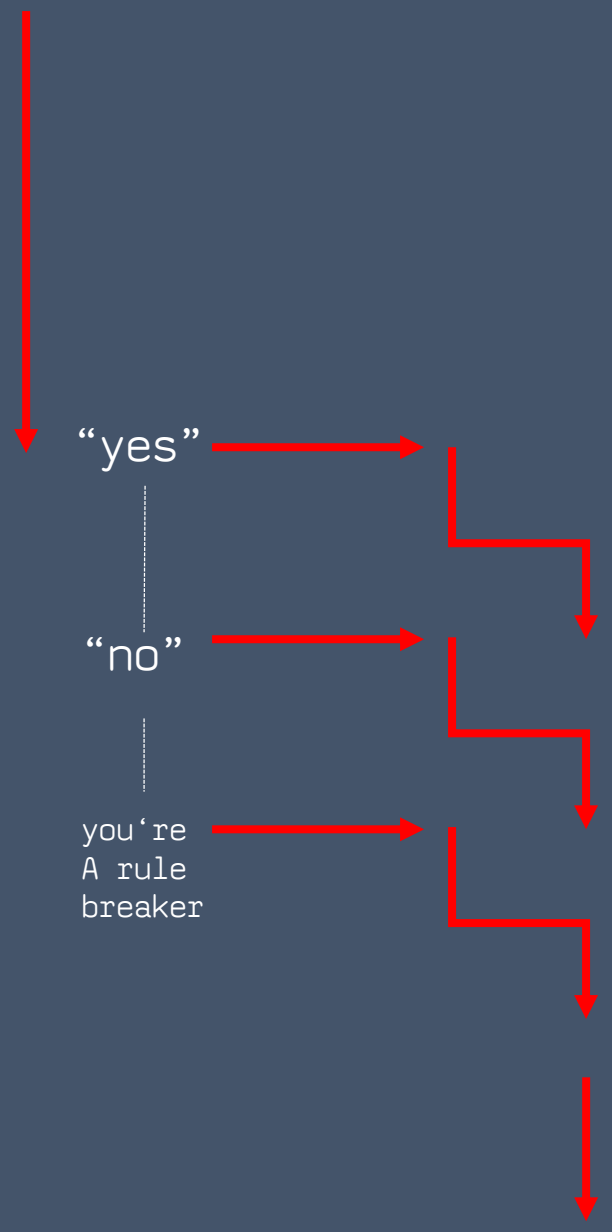
Review: flow of control

```
public class AddNumbers {
  public static void main(String[] args) {
    int a = 5;
    int b = 6;
    int c = 5 + 6;
    System.out.println(c);
  }
}

> ll
```

FLOW OF CONTROL

FLOW OF CONTROL

```
public class YesOrNo {
  public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    System.out.println("Yes or no?");
    String response = input.nextLine();
    response = response.toLowerCase();
    switch (response) {
      case "yes":
        System.out.println("You said yes!");
        break;
      case "no":
        System.out.println("You said no!");
        break;
      default:
        System.out.println("You didn't follow the rules!");
        break;
    }
  }
}
```



Loops

- Two types we encounter today: while/do…while
- Both change the way the flow of control works
- Loops allow instructions to be executed multiple times until a condition is met
- The process of repeating instructions over and over is called iteration.

The `while` loop

General form:

```
while (CONDITION) {
  // Statements to repeat
}
```

Always a boolean condition - meaning a condition which the loop evaluates for "truthiness"

Can be as many statements as desired

The do…while loop

General form:

```
do {
   // Statements to repeat
} while(CONDITION);
```

Can be as many statements as desired

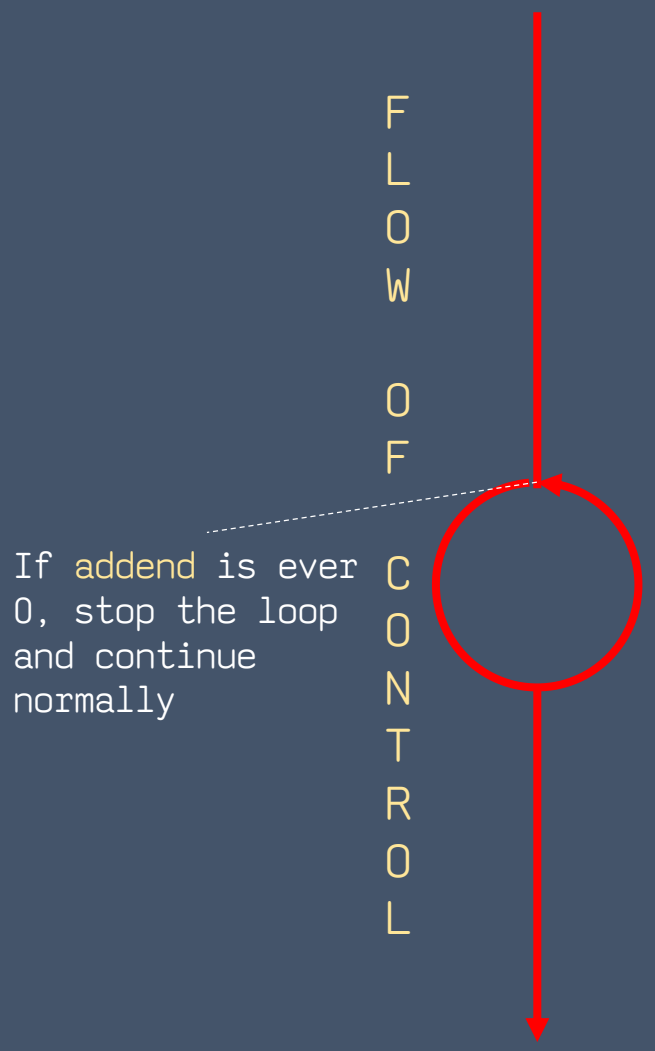
Always a boolean condition - meaning a condition which the loop evaluates for "truthiness"

while vs. do…while

while loops evaluate the condition before iterating.

do…while loops iterate at least once before evaluating the condition.

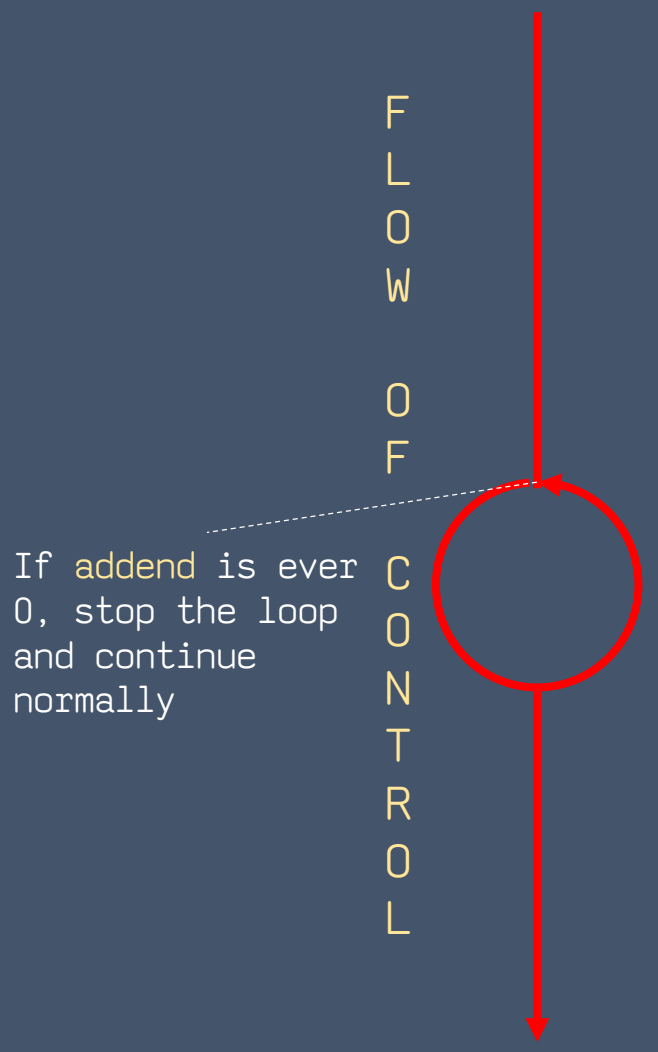
The `while` loop




```
public class DoAddition {

  public static void main(String[] args) {
    // Read keyboard input
    Scanner input = new Scanner(System.in);

    // Initialized variables
    int sum = 0;
    System.out.println("Enter an integer (0 to quit): ");
    int addend = input.nextInt();

    // Do math
    while (addend != 0) {
      sum += addend;
      System.out.print("Enter an integer (0 to quit): ");
      addend = input.nextInt();
    }

    // Print sum
    System.out.println("The sum of the numbers is: " + sum);
  }
}
```

Known as a "sentinel" value - a particular number for the loop to "watch out for."

The while loop




```
public class DoAddition {

  public static void main(String[] args) {
    // Read keyboard input
    Scanner input = new Scanner(System.in);

    // Initialized variables
    int sum = 0;
    System.out.println("Enter an integer (0 to quit): ");
    int addend = input.nextInt();
    int count = 0;

    // Do math
    while (addend != 0) {
      count++;
      sum += addend;
      System.out.print("Enter an integer (0 to quit): ");
      addend = input.nextInt();
    }

    // Print sum
    System.out.println("The sum of the numbers is: " + sum);
    System.out.println("Their average is: " + (double)sum/count);
  }
}
```

Keeps track of how many times the loop is executed

The `while` loop

```
// Initialized variables
int sum = 0;
System.out.println("Enter an integer (0 to quit): ");
int addend = input.nextInt();

// Do math
while (addend != 0) {
  sum += addend;
  System.out.print("Enter an integer (0 to quit): ");
  addend = input.nextInt();
}
```

Keep in mind that this structure still obeys some of the things we know about flow of control.

For example, variables need to be declared before we can use or evaluate them.

The `while` loop

Beware of infinite loops, though!

```
int a = 6;
int b = 5;

while (a > b) {
  a++;
}
```

To ∞ and beyond! (It never stops because the condition a > b is forever true.)

Activity

cd to your class activities repository
Perform a git pull download master
cd to the activity-11/Looping directory

We're going to improve our averaging program a bit.

The `while` loop

This is more useful than just calculating numbers. Java (like many programming languages) includes objects called iterators.

These objects allow us to test if, for example, a file has more than one line of input.

Iterators

Assume that our Scanner (input) is pointed to a file called file that has more than one line.

The Scanner object has methods which are iterators. These allow us to ask questions like this one, which is “do you have another line?”

```
while (input.hasNextLine()) {
  System.out.println(input.nextLine());
}
```

If it’s true that the file has more content, print the next line.

Iterators

We use while loops on iterators. This is because we need to ask questions about them such as: do you have more content?

This is strictly a boolean question: yes or no => true or false?

Iterators

This gives us an ability to begin to sort content, and manipulate data.

Much excite.

Very hype.

Iterators

Scanner, in particular has several different methods (“powers”) that let us interrogate data. It also allows us to examine Strings.

hasNextLine Asks a file if there’s another line

useDelimiter Changes the token use to read lines

The default was previously a space

Iterators

We're going to use these new "powers" in our second activity today.

cd to the activity-11/Iteration folder

If you're having issues getting the new content try a git stash or commit your current repository and then pull the new content!

Activity

Things we know about M & Ms:

- Come in 6 colors:
  - Brown
  - Yellow
  - Red
  - Green
  - Orange
  - Blue
- Blue is, by far, the best.
- Each bag has somewhere around 20 candies.

Activity

Our file has a given format:

Each line represents one bag
Each line's format is #B, #Y, #R, #G, #O, #B, #Total