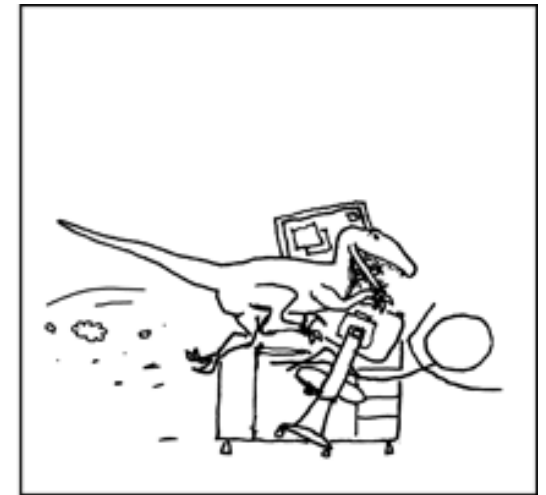
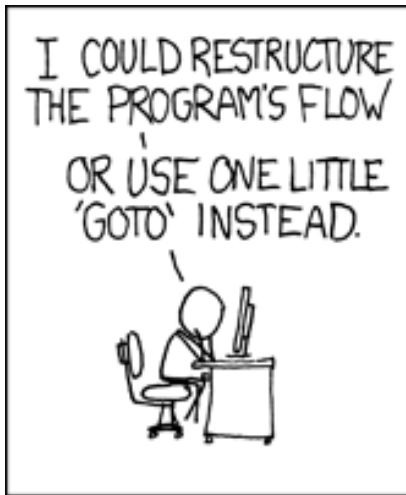
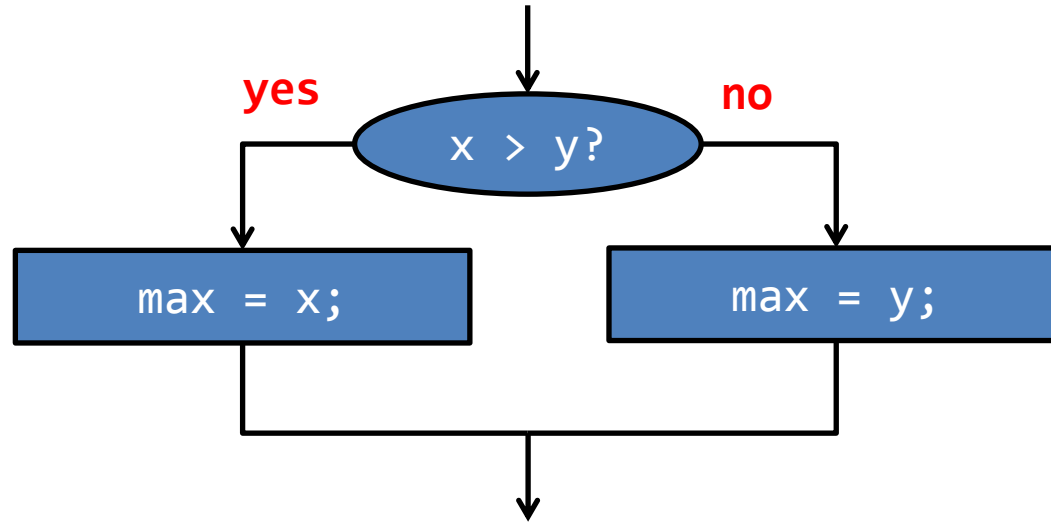


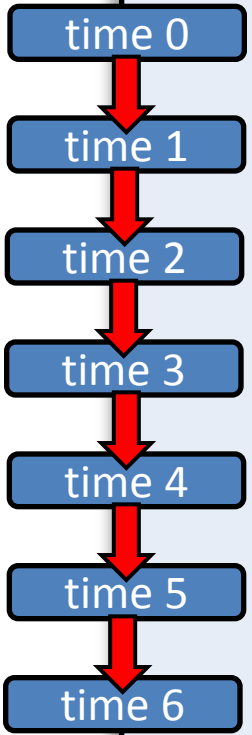
Conditionals, Loops, and Style



<http://xkcd.com/292/>

Control flow thus far

```
public class ArgsExample
{
    public static void main(String [] args)
    {
        String product = args[0];
        int qty = Integer.parseInt(args[1]);
        double cost = Double.parseDouble(args[2]);
        double total = qty * cost;
        System.out.print("To buy " + qty);
        System.out.print(" " + product);
        System.out.println(" you will need $" + total);
    }
}
```



Control flow

- Interesting and powerful programs need:
 - To skip over some lines
 - To repeat lines
- **Conditionals** → sometimes skip parts
- **Loops** → allow repetition of lines


if statement

- Common branching statement
- Evaluate a boolean expression
 - If true, do some stuff
 - If false, do some other stuff (optional)

Note lack of
semicolon!

```
if (expression)
{
    statement1;
    statement2;
    ...
}
```

```
if (expression)
{
    statement1;
    statement2;
    ...
}
else
{
    statement3;
    statement4;
    ...
}
```



if statement

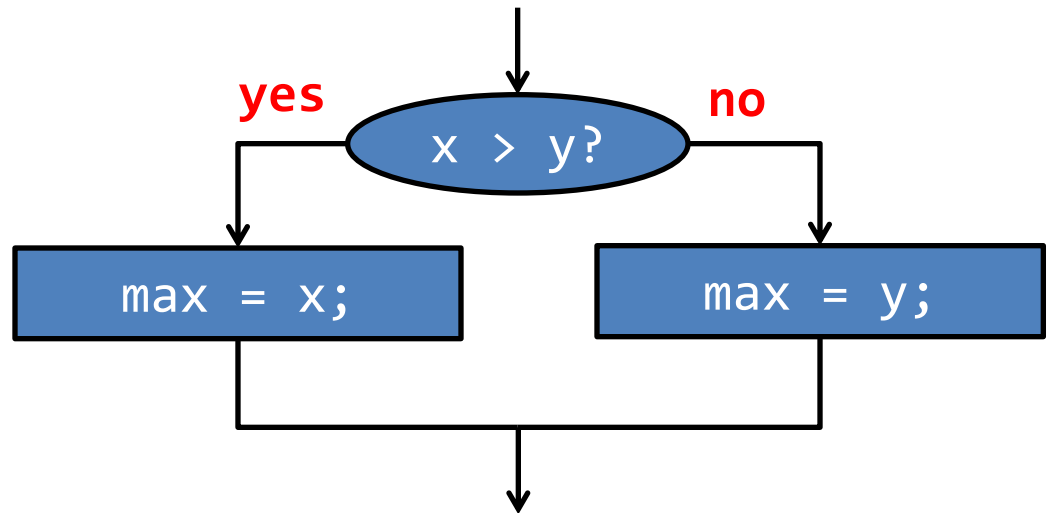
- {}'s optional if only one statement

```
if (expression)  
statement1;
```

```
if (expression)  
statement1;  
else  
statement2;
```

- Example:

```
if (x > y)  
max = x;  
else  
max = y;
```



if examples

```
if (x < 0)
    x = -x;
```

Take absolute value of x

```
if (Math.random() < 0.5)
    money = money * 2;
else
    money = 0.0;
```

Make a double or nothing bet with 50-50 odds.

```
if (x > y)
{
    int t = x;
    x = y;
    y = t;
}
```

Put x and y into sorted order

```
num = 0;
if (args.length > 0)
{
    num = Integer.parseInt(args[0]);
}
```

If a command line option is passed in, use it as the value for num.

Nested if

- Execute one of three options:

```
if (category == 0)
{
    title = "Books";
}
else
{
    if (category == 1)
    {
        title = "CDs";
    }
    else
    {
        title = "Misc";
    }
}
```

=

```
if (category == 0)
{
    title = "Books";
}
else if (category == 1)
{
    title = "CDs";
}
else
{
    title = "Misc";
}
```

- Both do exactly same thing
- Right probably more readable in general

while loop

- **while-loop**: common way to repeat code
 - Evaluate a boolean expression
 - If true, do a block a code
 - Go back to start of while loop
 - If false, skip over block

```
while (expression)
{
    statement1;
    statement2;
    ...
}
```

while loop with multiple statements in a {} block

```
while (expression)
    statement1;
```

while loop with a single statement

while loop example 1

- Print out summations, $0 + 1 + 2 + \dots + N$

```
public class Summation
{
    public static void main(String [] args)
    {
        int limit = Integer.parseInt(args[0]);
        int i      = 1;
        int sum    = 0;

        while (i <= limit)
        {
            sum += i;
            System.out.println("sum 0..." + i +
                               " = " + sum);

            i++;
        }
    }
}
```

```
% java Summation 4
```

```
sum 0...1 = 1
```

```
sum 0...2 = 3
```

```
sum 0...3 = 6
```

```
sum 0...4 = 10
```

while loop example 2

- Print powers of 2 up to but not including limit

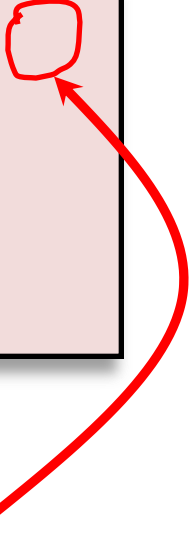
```
public class Powers2
{
    public static void main(String [] args)
    {
        int limit = Integer.parseInt(args[0]);
        long total = 1;
        while (total < limit)
        {
            System.out.println(total);
            total = total * 2;
        }
    }
}
```

```
% java Powers2 16
1
2
4
8
```

while loop

```
while (expression)
{
    statement1;
    statement2;
}
```

```
while (expression);
{
    statement1;
    statement2;
}
```




This semicolon is the entire body of
the while loop!

Almost *never* what you want.

while loop

```
while (expression)
{
    statement1;
    statement2;
}
```

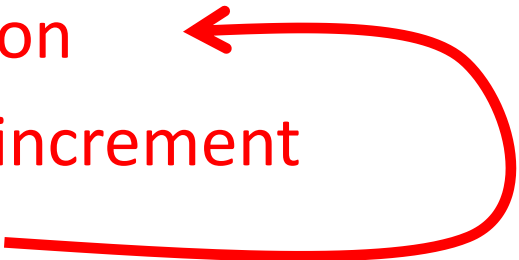
```
while (expression)
statement1;
statement2;
```



Only statement1 considered inside the
while loop.

Java doesn't care about indentation.
But I do (and so does your TA).

for loop

- **for-loop**: another common type of loop
 - Execute an **initialization** statement
 - Evaluate a **boolean expression**
 - If true, do **code block then increment**
 - If false, done with loop
- 

```
for (init; expression; increment)
{
    statement1;
    statement2;
    ...
}
```

for loop versions

```
for (init; expression; increment)
{
    statement1;
    statement2;
    ...
}
```

} block version

```
for (init; expression; increment)
    statement1;
```

single line version

```
for (init; expression; increment);
{
    statement1;
    statement2;
    ...
}
```

buggy version

for loop example

- Print out summations, $0 + 1 + 2 + \dots + N$

```
public class SummationFor
{
    public static void main(String [] args)
    {
        int limit = Integer.parseInt(args[0]);
        int sum = 0;

        for (int i = 1; i <= limit; i++)
        {
            sum += i;
            System.out.println("sum 0..." + i +
                               " = " + sum);
        }
    }
}
```

for loop anatomy

Declare and initialize a variable for use inside and outside the loop body

Condition which must be true to execute loop body

Changes the loop counter variable

```
int sum = 0;

for (int i = 1; i <= limit; i++)
{
    sum += i;
    System.out.println("sum 0..." + i +
        " = " + sum);
}
```

Declare and initialize a loop control variable

Loop body, executes 0 or more times

do while loop

- do while loop

- Always executes loop body at least once
- Do a block a code ←
- Evaluate a boolean expression
- If expression true, do block again

```
do
{
    statement1;
    statement2;
    ...
}
while (condition);
```

do while needs this
semicolon!

do while example

- Draw random points in $[0, 1)$ until we draw one in interval $[\text{left}, \text{right}]$

```
public class DrawPoints
{
    public static void main(String[] args)
    {
        double left = Double.parseDouble(args[0]);
        double right = Double.parseDouble(args[1]);
        double point = 0.0;
        int count = 0;

        do
        {
            point = Math.random();
            count++;
        }
        while ((point < left) || (point > right));

        System.out.println(count + " random draws");
    }
}
```

do while example runs

```
% java DrawPoints 0.1 0.2  
9 random draws
```

```
% java DrawPoints 0.1 0.2  
2 random draws
```

```
% java DrawPoints 0.1 0.11  
74 random draws
```

```
% java DrawPoints 0.1 0.2  
198 random draws
```

```
% java DrawPoints -0.2 -0.1  
(never terminates!)
```

```
% java DrawPoints 0.2 0.1  
(never terminates!)
```

- **Infinite loop:** possible for all loop types (while/for)
 - Eclipse, hit the **red stop button**
 - Command line, hit **ctrl-c**

Nested loops

- A loop inside another loop

```
public class StarTriangle
{
    public static void main(String[] args)
    {
        int limit = Integer.parseInt(args[0]);
        for (int i = 0; i < limit; i++)
        {
            for (int j = 0; j <= i; j++)
                System.out.print("*");
            System.out.println();
        }
    }
}
```

```
% java StarTriangle 4
```

```
*
**
***
****
```

Loop choice

- Does your loop need a **counter variable**?
 - e.g. Going from 0 to N or N to 0 in fixed steps
 - Use a **for loop**
 - Counter variable is local to loop
 - Harder to forget the increment/decrement
- Do you need an **unknown number of loops**?
 - Use a **while loop**
- Do you need to **loop at least once**?
 - Use a **do while loop**



<http://www.flickr.com/photos/onepointzero/1381580071/sizes/l/in/photostream/>

Style: comments

- Comments help reader/grader understand your program
 - Good comments explain why something is done
 - Write comments before coding tricky bits
 - Helps you formulate a plan
 - Don't comment the obvious:
 - `i++; // Increment i by one`

```
// Two slashes means a comment only on this line
```

```
/* Slash start means a comment  
that can go over multiple lines  
end with a start slash */
```

```
int dist = x + y; // Short comments can go here too
```

Style: naming things

- Variable names

- Begin with lowercase, uppercase each new word

- `int totalWidgets;`

- Class names

- Begin uppercase, then lowercase except for new words

- `public class InventoryTracker`

- Name exactly as in assignment description

- Constants

- All upper case, use `_` between words

- `double SPEED_LIGHT = 3.0e8;`

Style: whitespace

```
public class StarTriangle
{
    public static void main(String[] args)
        {int limit = Integer.parseInt(args[0]);
    for (int i=0;i<limit;i++){
            for (int j = 0; j <= i; j++)
                System.out.print("*");System.out.println();
        }
    }
}
```

- Indent each level of conditionals/loops
 - Indent a fixed number of spaces (3-4)
 - Eclipse can fix automatically, **ctrl-a then ctrl-i**
- Use blank lines to separate logical sections
- Only one statement per line

Style: whitespace

```
for (int i=0;i<limit;i++)
```

vs.

```
for (int i = 0; i < limit; i++)
```

```
a=b*c/d-(8.12*e);
```

vs.

```
a = b * c / d - (8.12 * e);
```

```
//this is a comment  
//describing my code
```

vs.

```
// this is a comment  
// describing my code
```

- Use spaces between
 - Statements in for loop
 - Operators in math expressions
 - After the // starting a comment

Style: whitespace

```
Math . random ();
```

vs.

```
Math.random();
```

```
args [0];
```

vs.

```
args[0];
```

```
i = i + 1 ;
```

vs.

```
i = i + 1;
```

- Do NOT use spaces between
 - method class, dot, name, or ()'s
 - array name and []'s
 - statement and ending semicolon

Style: whitespace

- Use **spaces to align parallel code** if it makes it more readable
 - Often **helps to spot mistakes**

```
int numPoints = Integer.parseInt(args[0]);
int startX = Integer.parseInt(args[0]);
int startY = Integer.parseInt(args[2]);
double velX = Integer.parseInt(args[3]);
double velY = Integer.parseInt(args[4]);
```

```
int    numPoints = Integer.parseInt(args[0]);
int    startX    = Integer.parseInt(args[0]);
int    startY    = Integer.parseInt(args[2]);
double velX     = Integer.parseInt(args[3]);
double velY     = Integer.parseInt(args[4]);
```

Style: curly bracing

```
public class HelloWorld
{
    public static void main(String [] args)
    {
        System.out.println("Hello world!");
    }
}
```

BSD-Allman style

```
public class HelloWorld {
    public static void main(String [] args) {
        System.out.println("Hello world!");
    }
}
```

K&R style

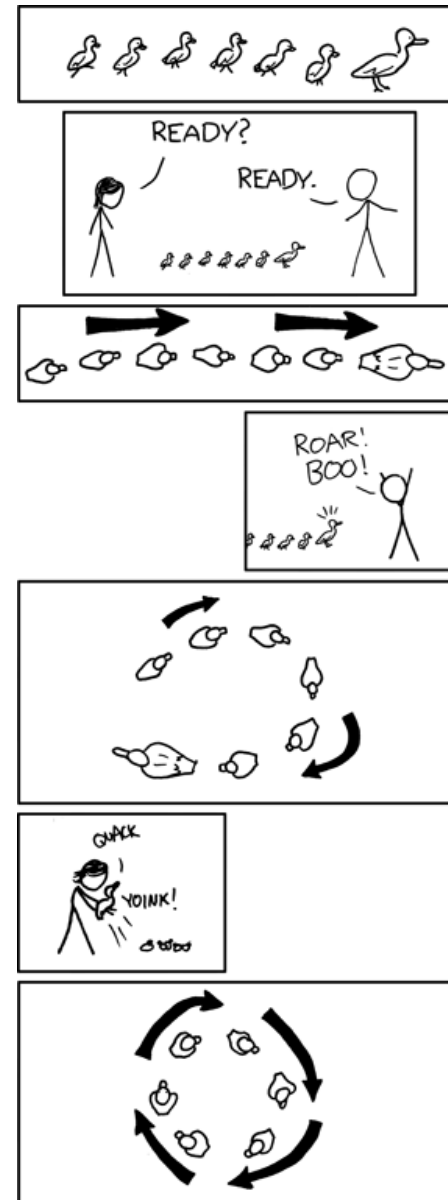
```
public class HelloWorld {
    public static void main(String [] args)
    {
        System.out.println("Hello world!");
    }
}
```

Choose a bracing style and stick to it!

No mixing and matching!

Summary

- Program flow of control
 - Conditionals skip sections
 - if statement
 - Loops repeat sections
 - while loop, for loop, do while loop
 - Conditionals and loops can be nested
 - Best loop depends on the situation
- Style
 - Makes code easier to read + grade
 - Good style = fewer bugs



OPERATION: DUCKLING LOOP

<http://xkcd.com/537/>