Finishing Ch. 2

Shortcut Assignments

```
int numStudents = 350;
// We want to increment?

numStudents = numStudents + 1;

// Some alternatives/shortcuts to adding one

numStudents++;

// Or more generally

numStudents += 1;
```

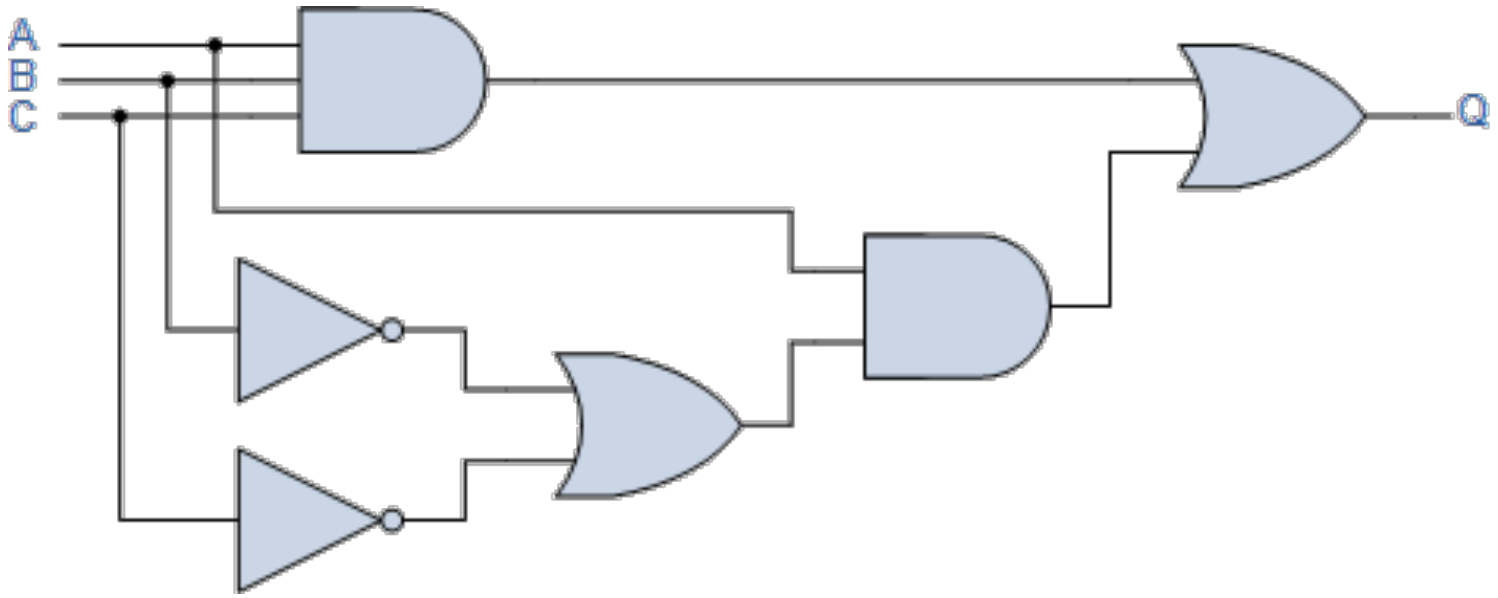
Shortcut Assignments

Various Operations

```
i++, ++i, x+=value, x-=value, x*=value, x/=value
```

Boolean Expressions

- Any expression that evaluates to true or false.
- Recall: an expression is a construct that evaluates to a *value*



Logical Operators

Operators that have boolean operands −boolean is output

A && B − true if both A and B are true

A || B − true if A, or B, or both are true

!A − true if A is false, false if A is true. Read Not A

Truth Table

A	**B**	**A && B**	**A || B**	**!A**
True	True	True	True	False
True	False	False	True	
False	True	False	True	True
False	False	False	False	

```
boolean leftArg = true;
boolean rightArg = false;

// What gets printed out?
System.out.println(!leftArg && rightArg);

System.out.println(!leftArg || !rightArg);

System.out.println(!(leftArg && rightArg));

System.out.println(!(leftArg && !rightArg));
```

For every boolean value of arg1 and arg2, are these two expressions equal?

```
!arg1 && !arg2
!(arg1 || arg2)
```

De Morgan's Law

A	B	!A && !B	!(A || B)
True	True	False	False
True	False	False	False
False	True	False	False
False	False	True	True

A	B	!A || !B	!(A && B)
True	True	False	False
True	False	True	True
False	True	True	True
False	False	True	True

Logical Expressions

Logical Expressions – two boolean operands, output is boolean.

Relational Operators

<, >, <=, >=

```
boolean result;
int arg1 = 5;
int arg2 = 10;
result = arg1 < arg2; // result is true
result = arg1 > arg2; // result is false
```

Logical Expressions

Equality Operators

==, !=

```
boolean flag;
int i = 5, j = 7;
flag = (i == j); // flag is false
flag = (i != j); // flag is true
```

Documentation

- http://docs.oracle.com/javase/8/docs/api/
- Detailed documents describing how to use the extensive set of classes for creating programs.
  - Import packages
  - Use of classes
  - Use of methods

Debugging

- Determining and fixing the cause of a problem in a computer program.

Debugging

- Syntax errors – relatively easy to find; at least the compiler will tell us where these are.

Test.java | StarTest.java | Star.java | Errors.java | CircumferenceToArea.java

```
4    public static void main (String [] args) {
5        Scanner scr = new Scanner(System.in);
6        double circleRadius        = 0.0;
7        double circleCircumference = 0.0;
8        double circleArea          = 0.0;
9        final double PI_VAL        = 3.14159265;
10
11       System.out.print("Enter circumference: ");
12       circleCircumference = scnr.nextDouble();
```

```
Dustins-MBP-3:Examples dustinadams$ javac CircumferenceToArea.java
CircumferenceToArea.java:12: error: cannot find symbol
      circleCircumference = scnr.nextDouble();
                            ^
  symbol:   variable scnr
  location: class CircumferenceToArea
1 error
```

Debugging

- Runtime errors − The program compiles but crashes. The JVM will usually tell us the line of code that caused the program to crash.

```
Exception in thread "main" java.lang.StringIndexOutOfBoundsException:
        at java.lang.String.charAt(String.java:658)
        at Errors.main(Errors.java:12)
```

Debugging

- Logical errors − Design flaw in the program. These are more subtle.
  - Get out a sheet of paper and work out the program by hand
  - Use System.out.println() statements to verify calculations along the way
  - Consider 2.16 for a thorough example of finding a logical error

Style & Formatting

- Review table in 2.17 for details