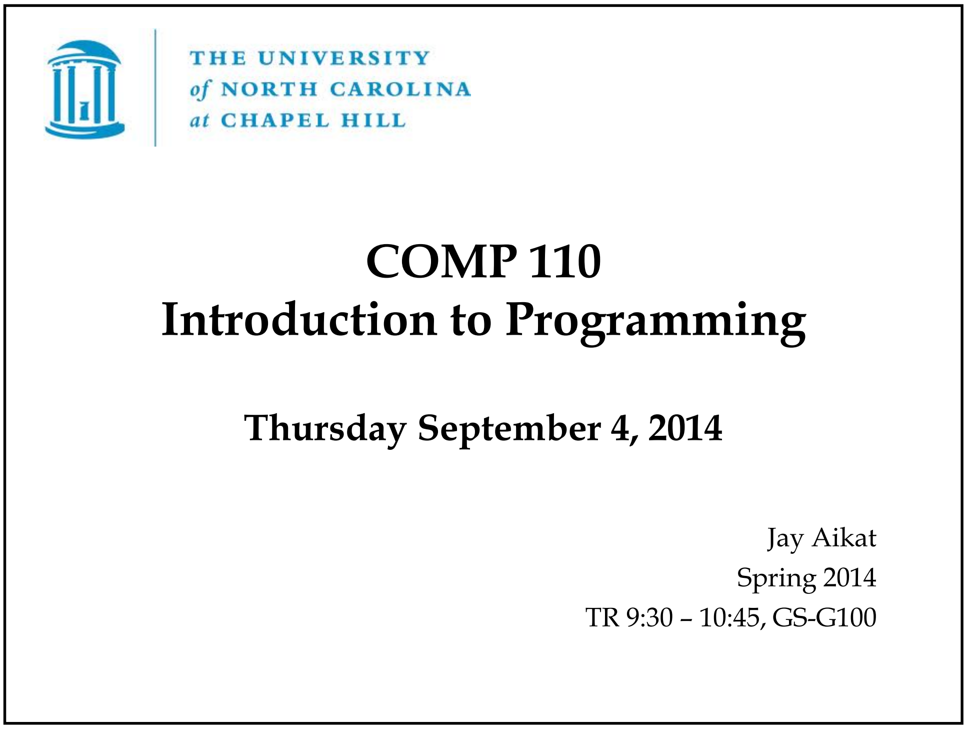
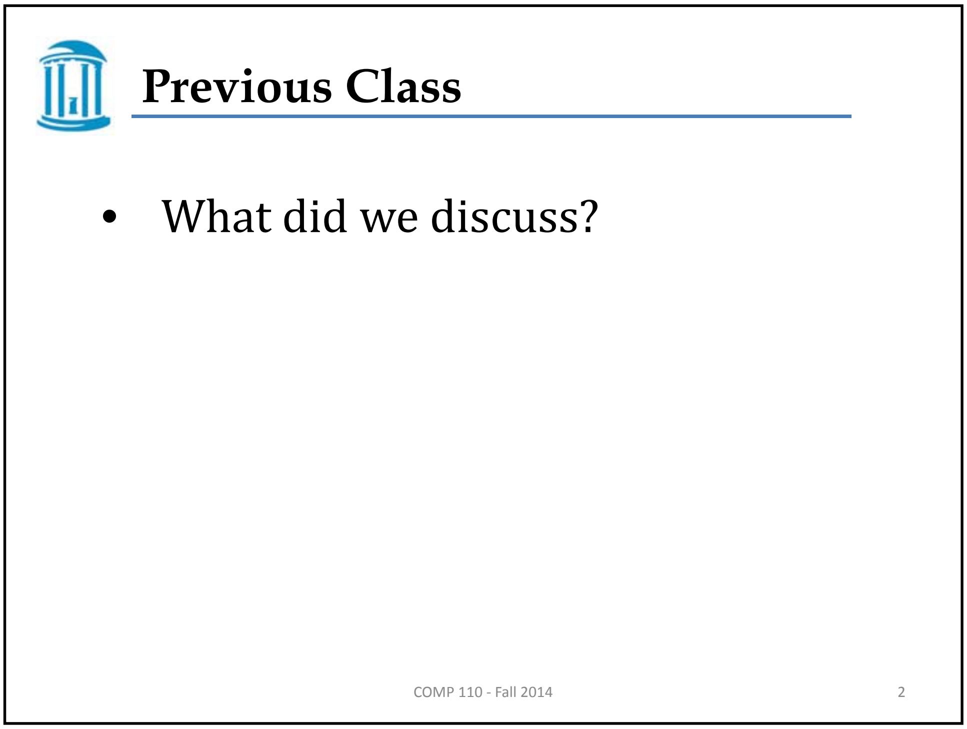


THE UNIVERSITY *of* NORTH CAROLINA *at* CHAPEL HILL

COMP 110
Introduction to Programming

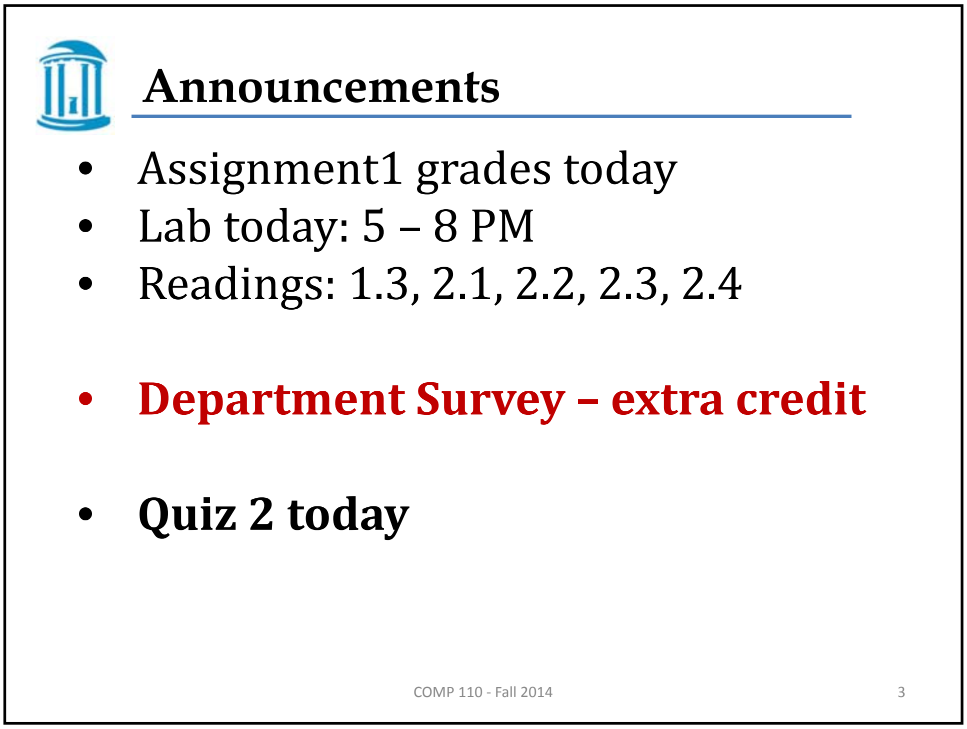
Thursday September 4, 2014

Jay Aikat
Spring 2014
TR 9:30 – 10:45, GS-G100



Previous Class

- What did we discuss?

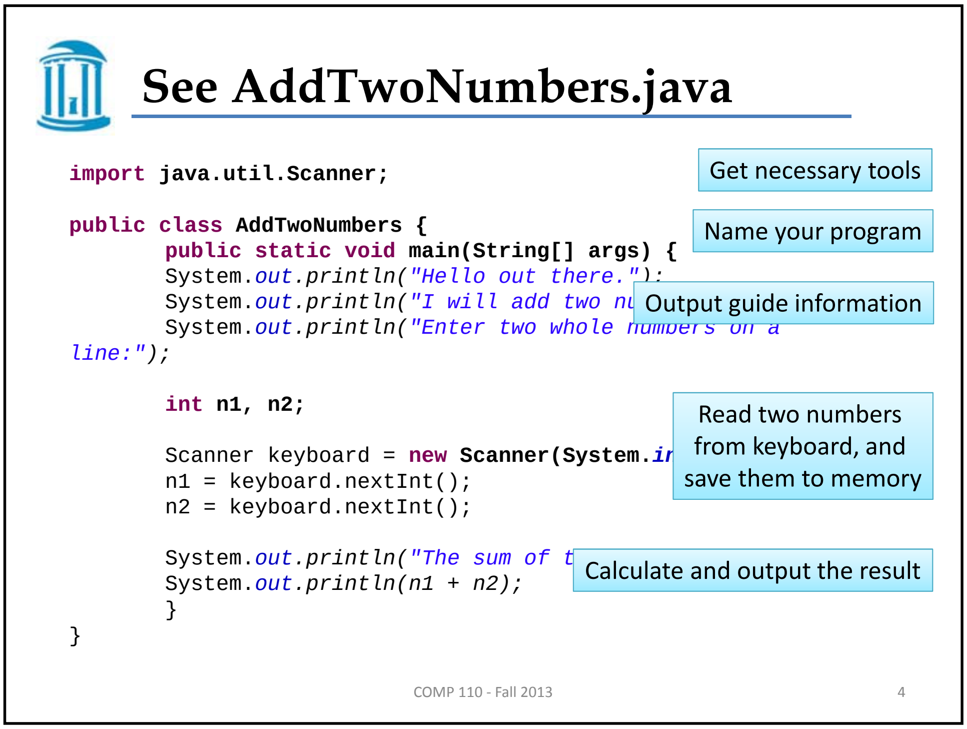


Announcements

- Assignment1 grades today
- Lab today: 5 – 8 PM
- Readings: 1.3, 2.1, 2.2, 2.3, 2.4

- **Department Survey – extra credit**

- **Quiz 2 today**



See AddTwoNumbers.java

```
import java.util.Scanner;

public class AddTwoNumbers {
        public static void main(String[] args) {
        System.out.println("Hello out there.");
        System.out.println("I will add two nu
        System.out.println("Enter two whole numbers on a
line:");

        int n1, n2;

        Scanner keyboard = new Scanner(System.in
        n1 = keyboard.nextInt();
        n2 = keyboard.nextInt();

        System.out.println("The sum of t
        System.out.println(n1 + n2);
        }
}
```

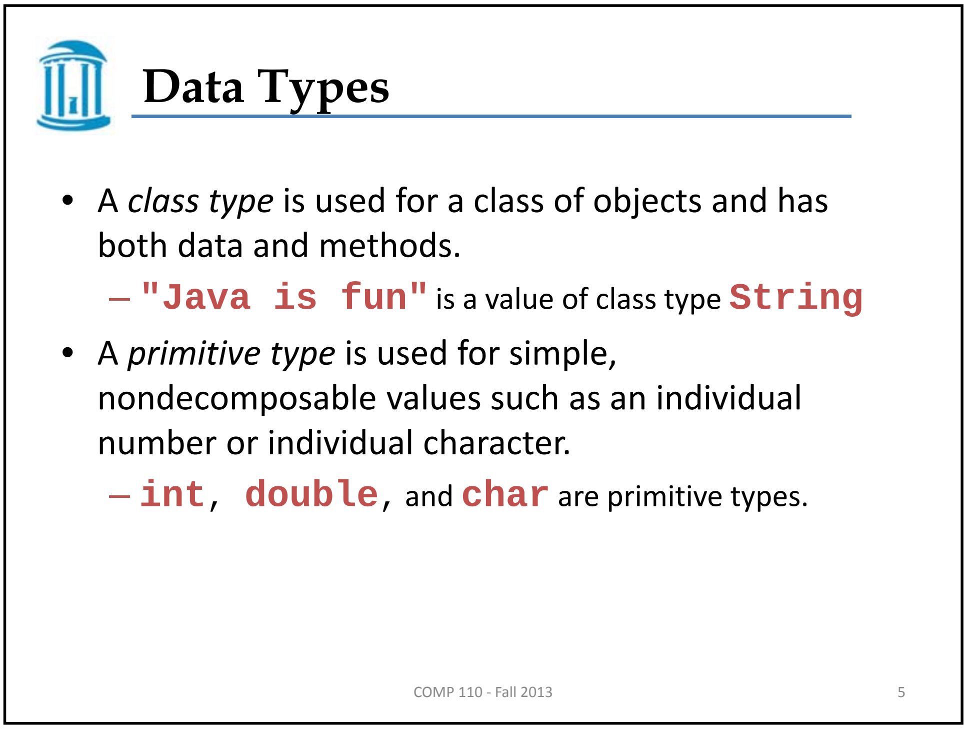
Get necessary tools

Name your program

Output guide information

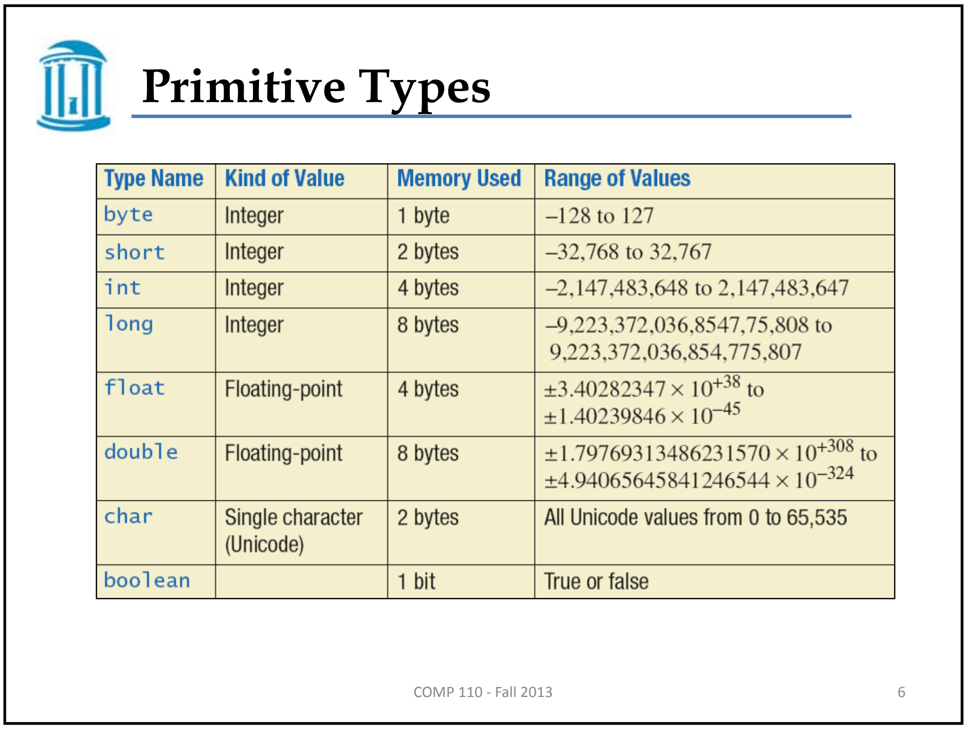
Read two numbers from keyboard, and save them to memory

Calculate and output the result



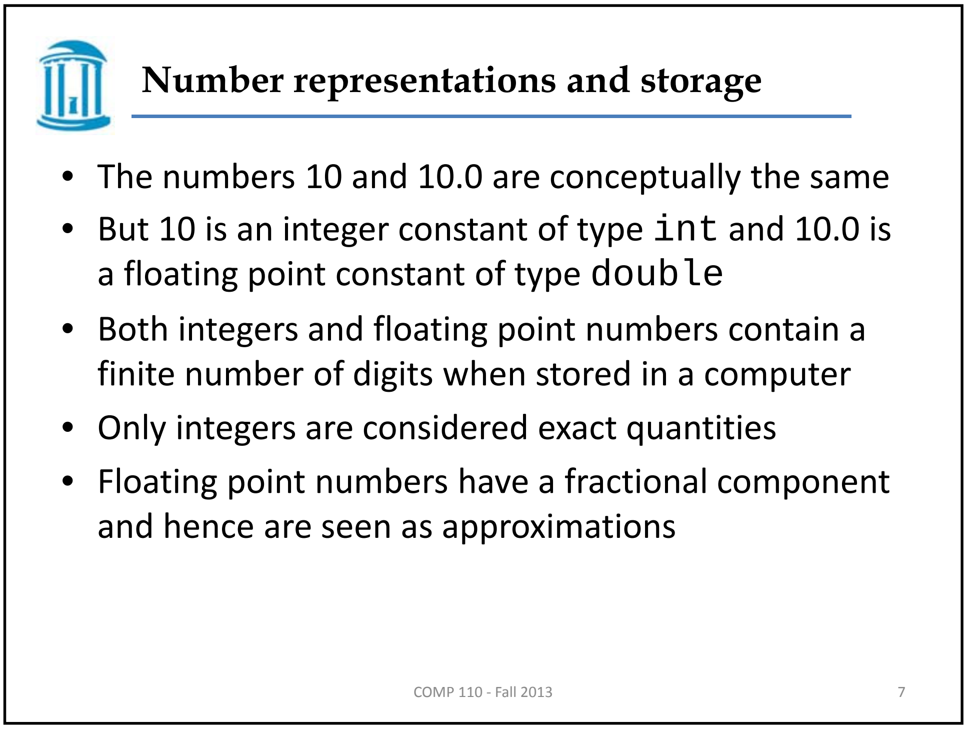
Data Types

- A *class type* is used for a class of objects and has both data and methods.
  - `"Java is fun"` is a value of class type `String`
- A *primitive type* is used for simple, nondecomposable values such as an individual number or individual character.
  - `int`, `double`, and `char` are primitive types.



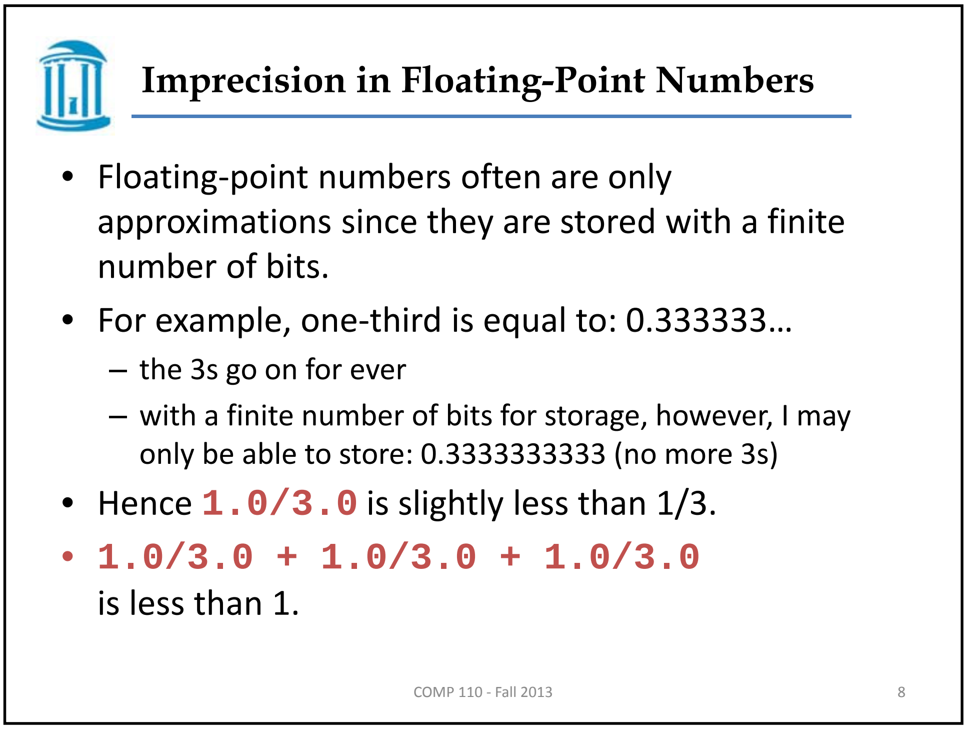
Primitive Types

Type Name	Kind of Value	Memory Used	Range of Values
`byte`	Integer	1 byte	–128 to 127
`short`	Integer	2 bytes	–32,768 to 32,767
`int`	Integer	4 bytes	–2,147,483,648 to 2,147,483,647
`long`	Integer	8 bytes	–9,223,372,036,8547,75,808 to 9,223,372,036,854,775,807
`float`	Floating-point	4 bytes	$\pm 3.40282347 \times 10^{+38}$ to $\pm 1.40239846 \times 10^{-45}$
`double`	Floating-point	8 bytes	$\pm 1.79769313486231570 \times 10^{+308}$ to $\pm 4.94065645841246544 \times 10^{-324}$
`char`	Single character (Unicode)	2 bytes	All Unicode values from 0 to 65,535
`boolean`		1 bit	True or false



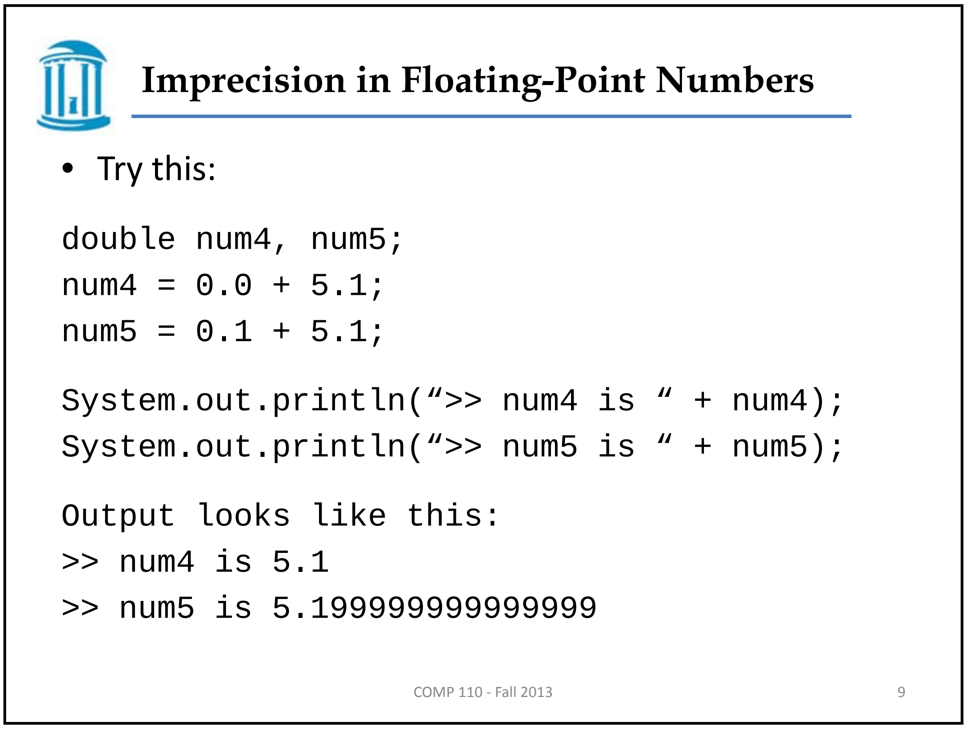
Number representations and storage

- The numbers 10 and 10.0 are conceptually the same
- But 10 is an integer constant of type `int` and 10.0 is a floating point constant of type `double`
- Both integers and floating point numbers contain a finite number of digits when stored in a computer
- Only integers are considered exact quantities
- Floating point numbers have a fractional component and hence are seen as approximations



Imprecision in Floating-Point Numbers

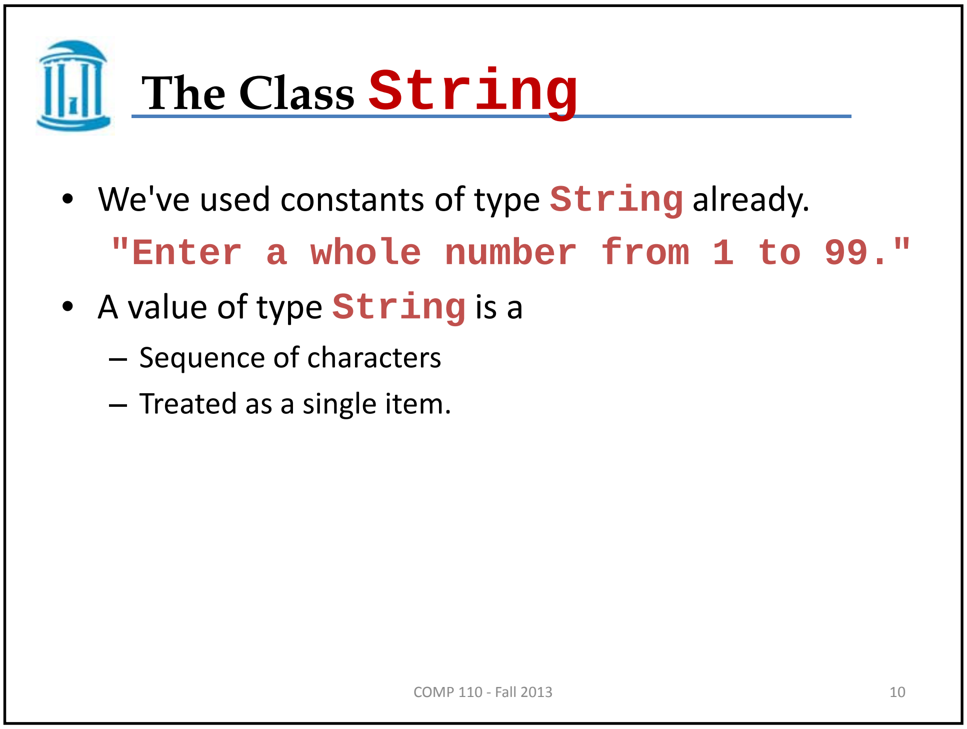
- Floating-point numbers often are only approximations since they are stored with a finite number of bits.
- For example, one-third is equal to: 0.333333…
  - the 3s go on for ever
  - with a finite number of bits for storage, however, I may only be able to store: 0.3333333333 (no more 3s)
- Hence **`1.0/3.0`** is slightly less than 1/3.
- **`1.0/3.0 + 1.0/3.0 + 1.0/3.0`** is less than 1.



Imprecision in Floating-Point Numbers

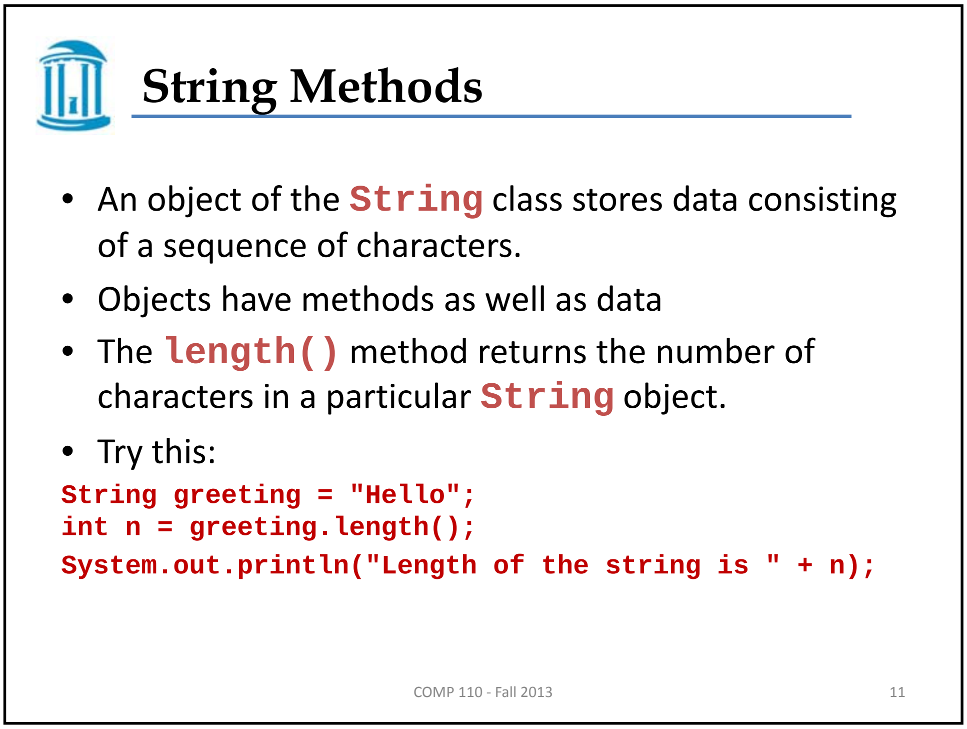
- Try this:

```
double num4, num5;
num4 = 0.0 + 5.1;
num5 = 0.1 + 5.1;

System.out.println(“>> num4 is “ + num4);
System.out.println(“>> num5 is “ + num5);

Output looks like this:
>> num4 is 5.1
>> num5 is 5.199999999999999
```



The Class `String`

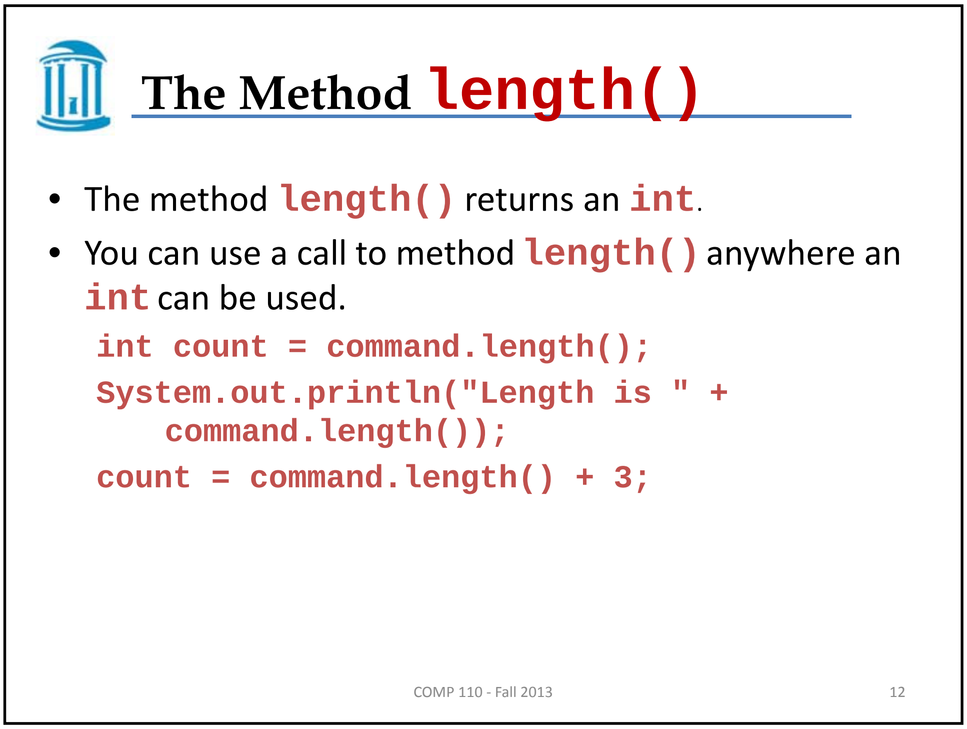
- We've used constants of type `String` already.
  `"Enter a whole number from 1 to 99."`
- A value of type `String` is a
  - Sequence of characters
  - Treated as a single item.



String Methods

- An object of the `String` class stores data consisting of a sequence of characters.
- Objects have methods as well as data
- The `length()` method returns the number of characters in a particular `String` object.
- Try this:

```
String greeting = "Hello";
int n = greeting.length();
System.out.println("Length of the string is " + n);
```

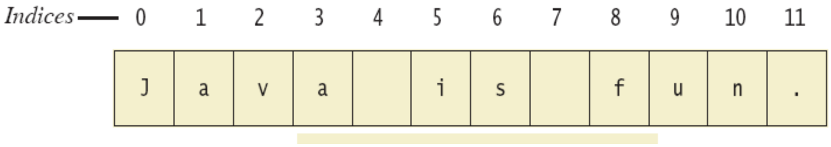


The Method `length()`

- The method `length()` returns an `int`.
- You can use a call to method `length()` anywhere an `int` can be used.

```
int count = command.length();
System.out.println("Length is " +
    command.length());
count = command.length() + 3;
```

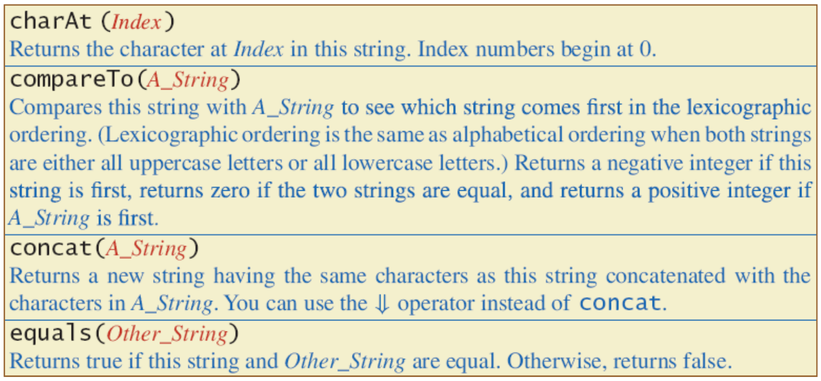
String Indices



Indices	0	1	2	3	4	5	6	7	8	9	10	11
	J	a	v	a		i	s		f	u	n	.

- Positions start with 0, not 1.
  - The 'J' in "Java is fun." is in position 0
- A position is referred to an an *index.*
  - The `'f'` in `"Java is fun."` is at index 8.

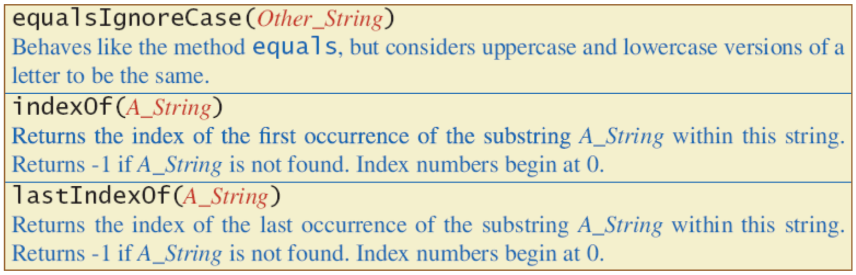
String Methods



`charAt` (*Index*) Returns the character at *Index* in this string. Index numbers begin at 0.
`compareTo`(*A_String*) Compares this string with *A_String* to see which string comes first in the lexicographic ordering. (Lexicographic ordering is the same as alphabetical ordering when both strings are either all uppercase letters or all lowercase letters.) Returns a negative integer if this string is first, returns zero if the two strings are equal, and returns a positive integer if *A_String* is first.
`concat`(*A_String*) Returns a new string having the same characters as this string concatenated with the characters in *A_String*. You can use the ⇓ operator instead of `concat`.
`equals`(*Other_String*) Returns true if this string and *Other_String* are equal. Otherwise, returns false.

Methods and their return types…

String Methods



`equalsIgnoreCase(Other_String)`

Behaves like the method `equals`, but considers uppercase and lowercase versions of a letter to be the same.

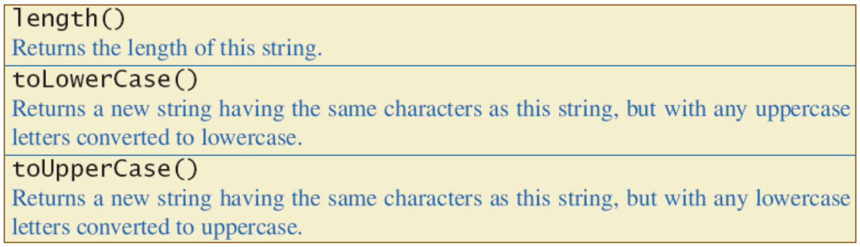
`indexOf(A_String)`

Returns the index of the first occurrence of the substring *A_String* within this string. Returns -1 if *A_String* is not found. Index numbers begin at 0.

`lastIndexOf(A_String)`

Returns the index of the last occurrence of the substring *A_String* within this string. Returns -1 if *A_String* is not found. Index numbers begin at 0.

String Methods



`length()`

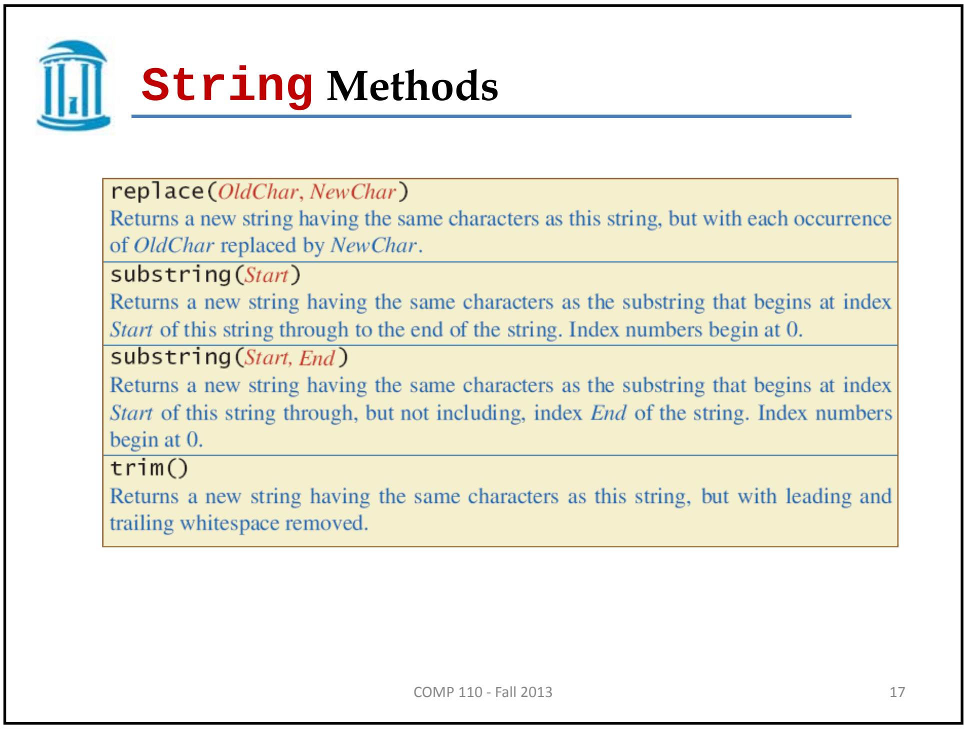
Returns the length of this string.

`toLowerCase()`

Returns a new string having the same characters as this string, but with any uppercase letters converted to lowercase.

`toUpperCase()`

Returns a new string having the same characters as this string, but with any lowercase letters converted to uppercase.



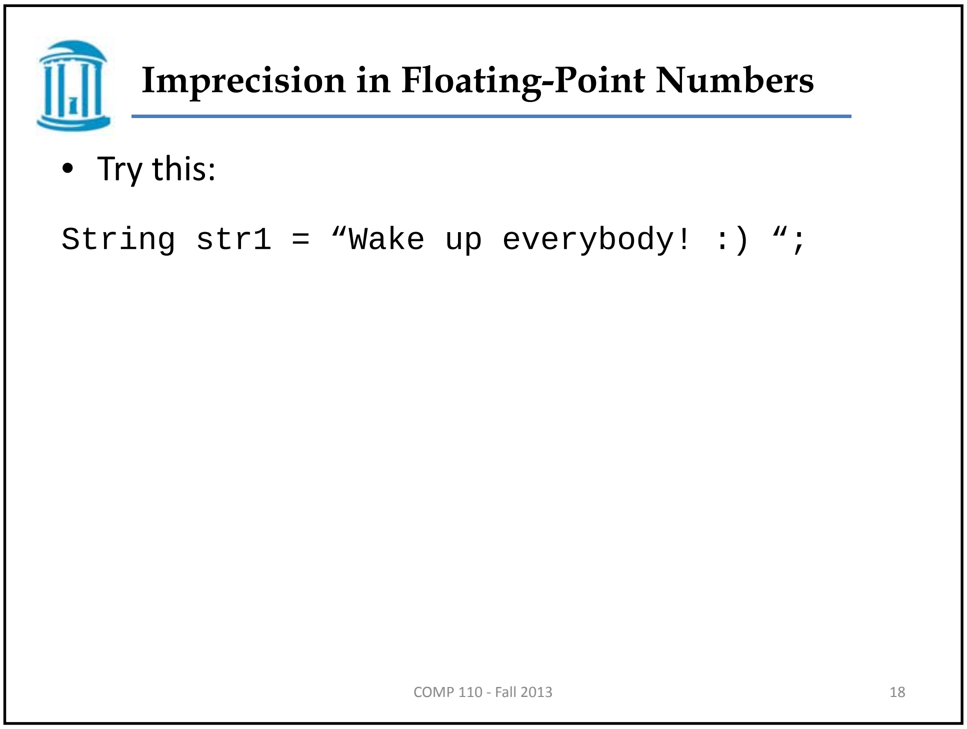
String Methods

`replace(`*OldChar, NewChar*`)`
Returns a new string having the same characters as this string, but with each occurrence of *OldChar* replaced by *NewChar*.

`substring(`*Start*`)`
Returns a new string having the same characters as the substring that begins at index *Start* of this string through to the end of the string. Index numbers begin at 0.

`substring(`*Start, End*`)`
Returns a new string having the same characters as the substring that begins at index *Start* of this string through, but not including, index *End* of the string. Index numbers begin at 0.

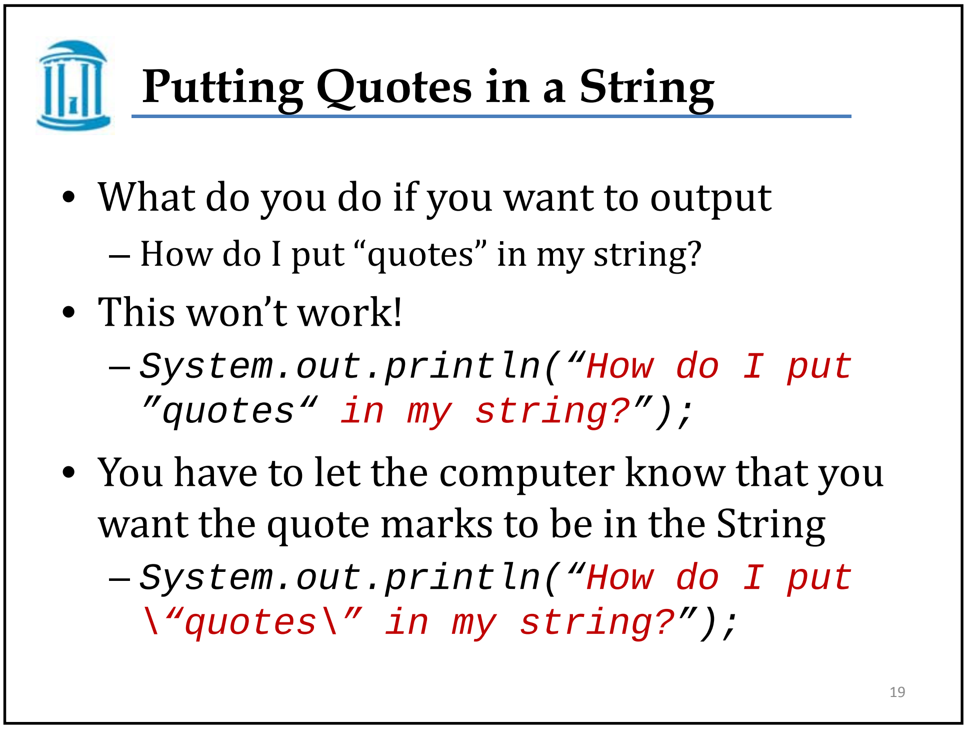
`trim()`
Returns a new string having the same characters as this string, but with leading and trailing whitespace removed.



Imprecision in Floating-Point Numbers

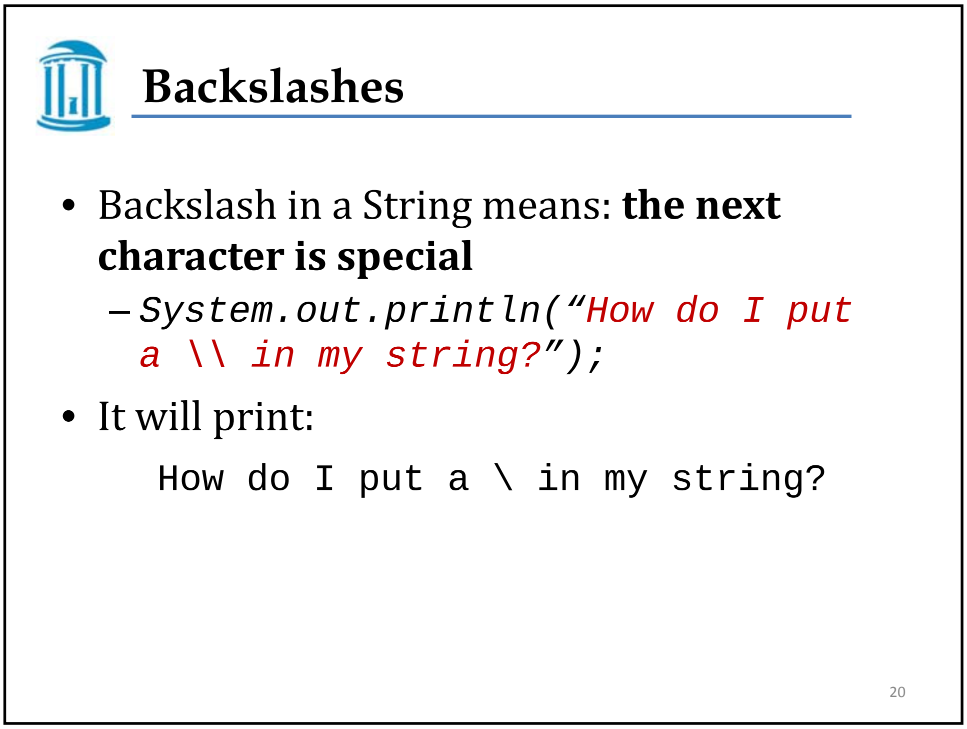
- Try this:

```
String str1 = “Wake up everybody! :) “;
```



Putting Quotes in a String

- What do you do if you want to output
  - How do I put “quotes” in my string?
- This won’t work!
  - `System.out.println(“How do I put ”quotes“ in my string?”);`
- You have to let the computer know that you want the quote marks to be in the String
  - `System.out.println(“How do I put \“quotes\” in my string?”);`



Backslashes

- Backslash in a String means: **the next character is special**
  - `System.out.println(“How do I put a \\ in my string?”);`
- It will print:

```
How do I put a \ in my string?
```

Escape Characters

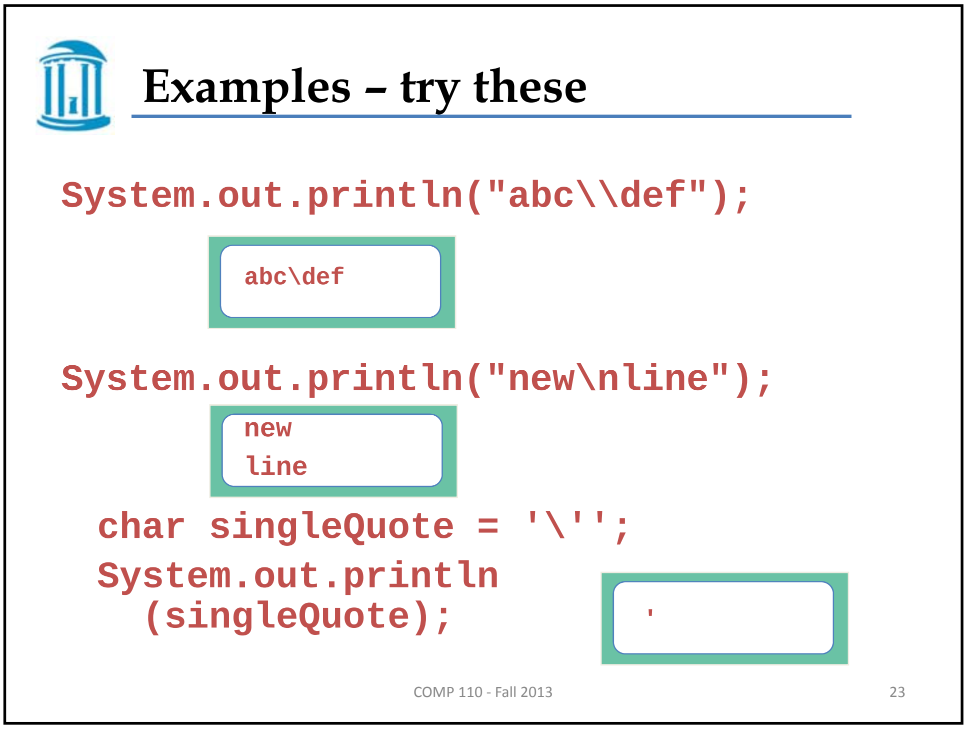
\”	Double quote
\’	Single quote
\\	Backslash
\n	New line
\t	Tab

- Each escape sequence is a single character even though it is written with two symbols.

Escape Characters

- How would you print
  `"Java" refers to a language.`   ?
- The compiler needs to be told that the quotation marks (`"`) do not signal the start or end of a string, but instead are to be printed.

```
System.out.println(
"\"Java\" refers to a language.");
```



Examples – try these

```
System.out.println("abc\\def");
```

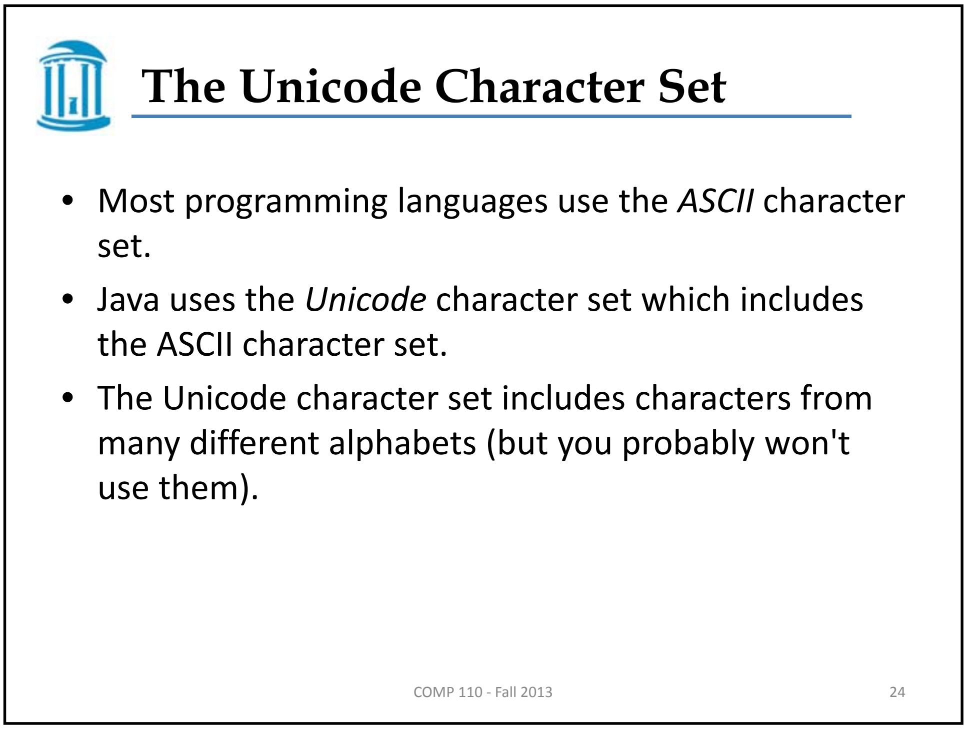
```
abc\def
```

```
System.out.println("new\nline");
```

```
new
line
```

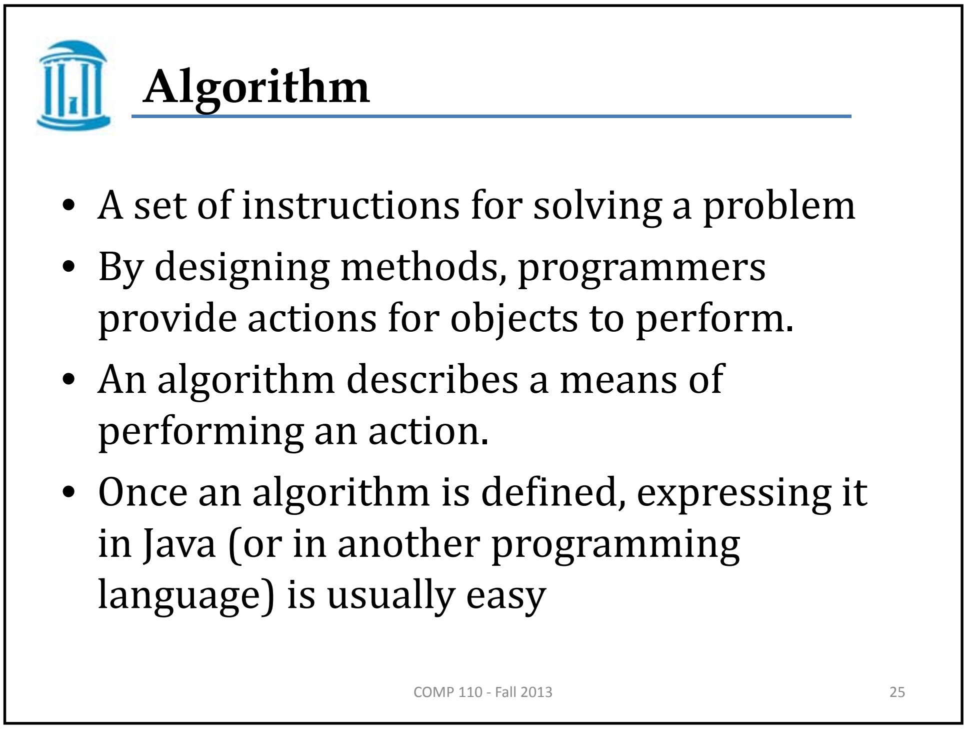
```
char singleQuote = '\'';
System.out.println
  (singleQuote);
```

```
'
```



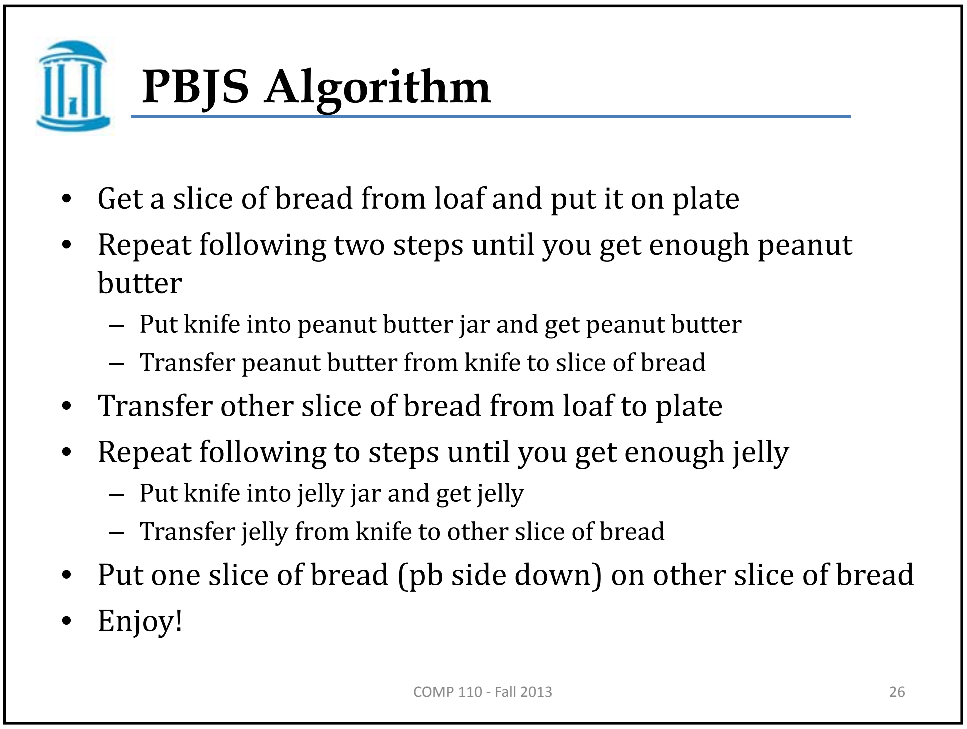
The Unicode Character Set

- Most programming languages use the *ASCII* character set.
- Java uses the *Unicode* character set which includes the ASCII character set.
- The Unicode character set includes characters from many different alphabets (but you probably won't use them).



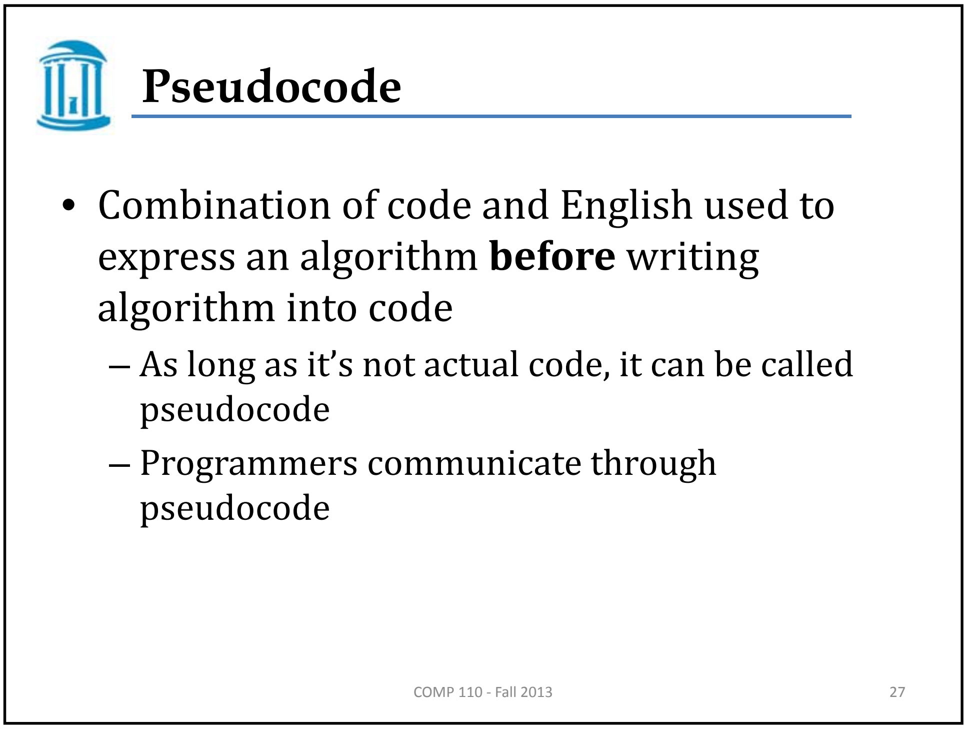
Algorithm

- A set of instructions for solving a problem
- By designing methods, programmers provide actions for objects to perform.
- An algorithm describes a means of performing an action.
- Once an algorithm is defined, expressing it in Java (or in another programming language) is usually easy



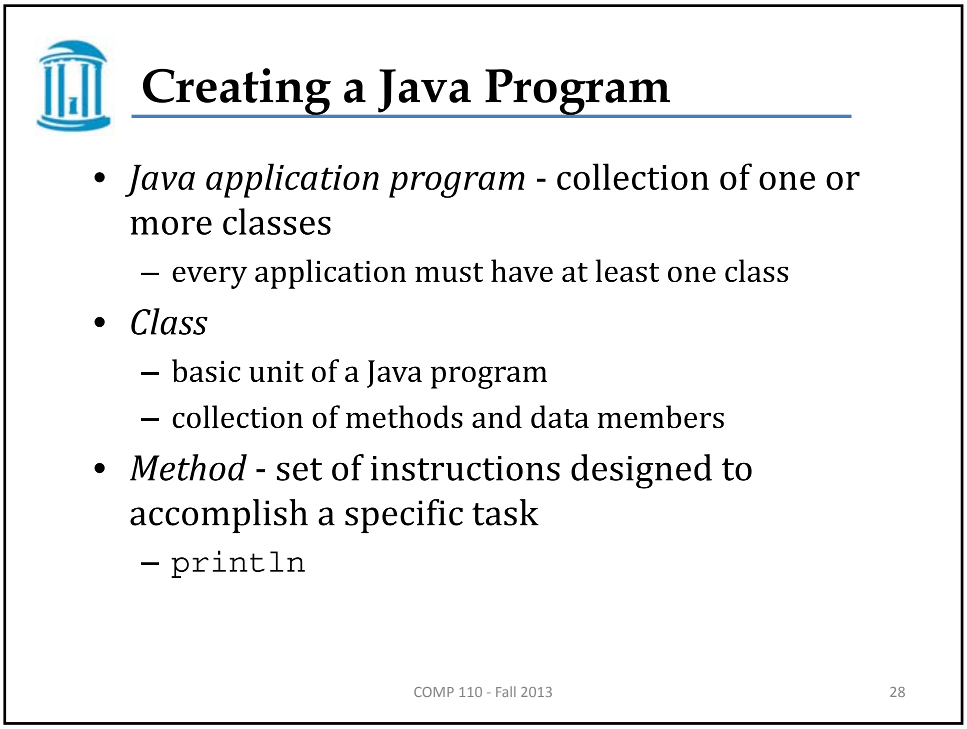
PBJS Algorithm

- Get a slice of bread from loaf and put it on plate
- Repeat following two steps until you get enough peanut butter
  - Put knife into peanut butter jar and get peanut butter
  - Transfer peanut butter from knife to slice of bread
- Transfer other slice of bread from loaf to plate
- Repeat following to steps until you get enough jelly
  - Put knife into jelly jar and get jelly
  - Transfer jelly from knife to other slice of bread
- Put one slice of bread (pb side down) on other slice of bread
- Enjoy!



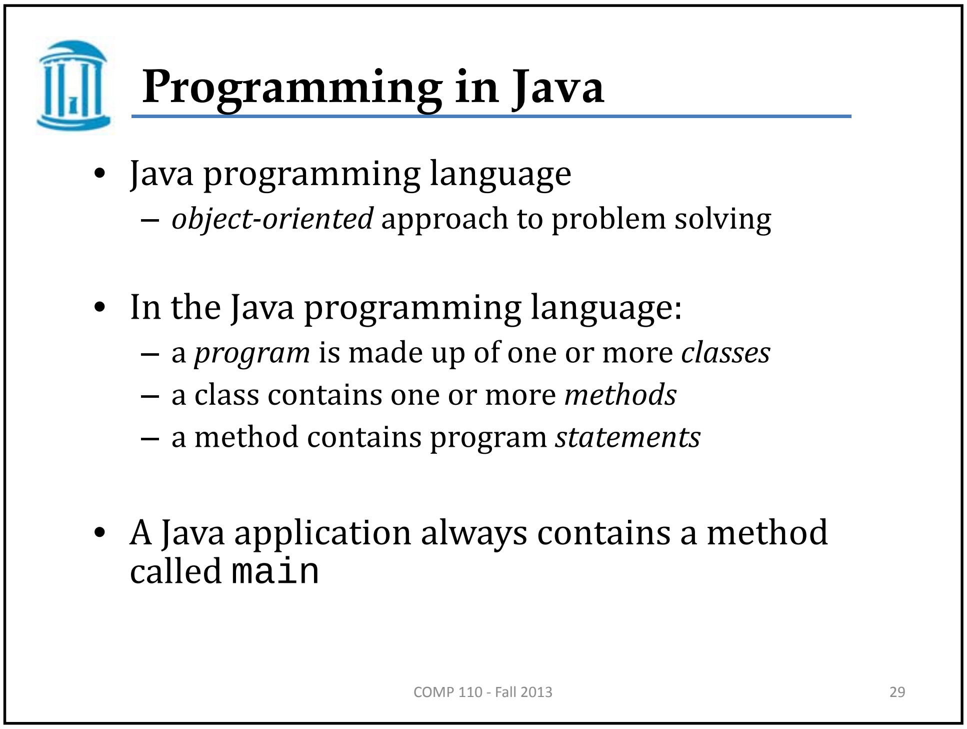
Pseudocode

- Combination of code and English used to express an algorithm **before** writing algorithm into code
  - As long as it’s not actual code, it can be called pseudocode
  - Programmers communicate through pseudocode



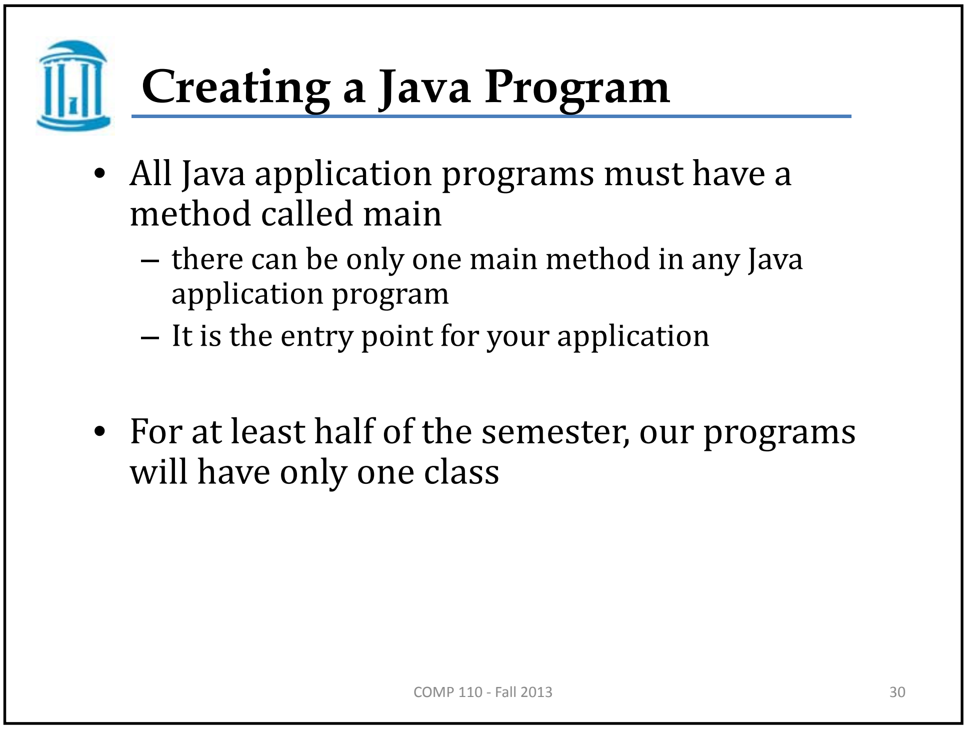
Creating a Java Program

- *Java application program* - collection of one or more classes
  - every application must have at least one class
- *Class*
  - basic unit of a Java program
  - collection of methods and data members
- *Method* - set of instructions designed to accomplish a specific task
  - `println`



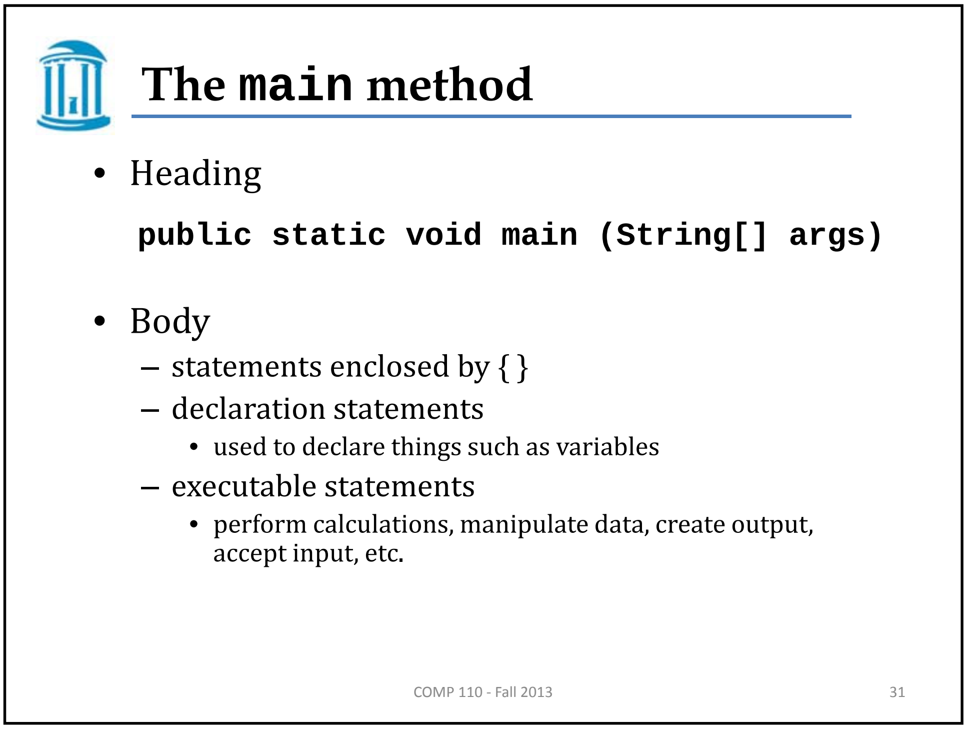
Programming in Java

- Java programming language
  - *object-oriented* approach to problem solving
- In the Java programming language:
  - a *program* is made up of one or more *classes*
  - a class contains one or more *methods*
  - a method contains program *statements*
- A Java application always contains a method called `main`



Creating a Java Program

- All Java application programs must have a method called main
  - there can be only one main method in any Java application program
  - It is the entry point for your application
- For at least half of the semester, our programs will have only one class

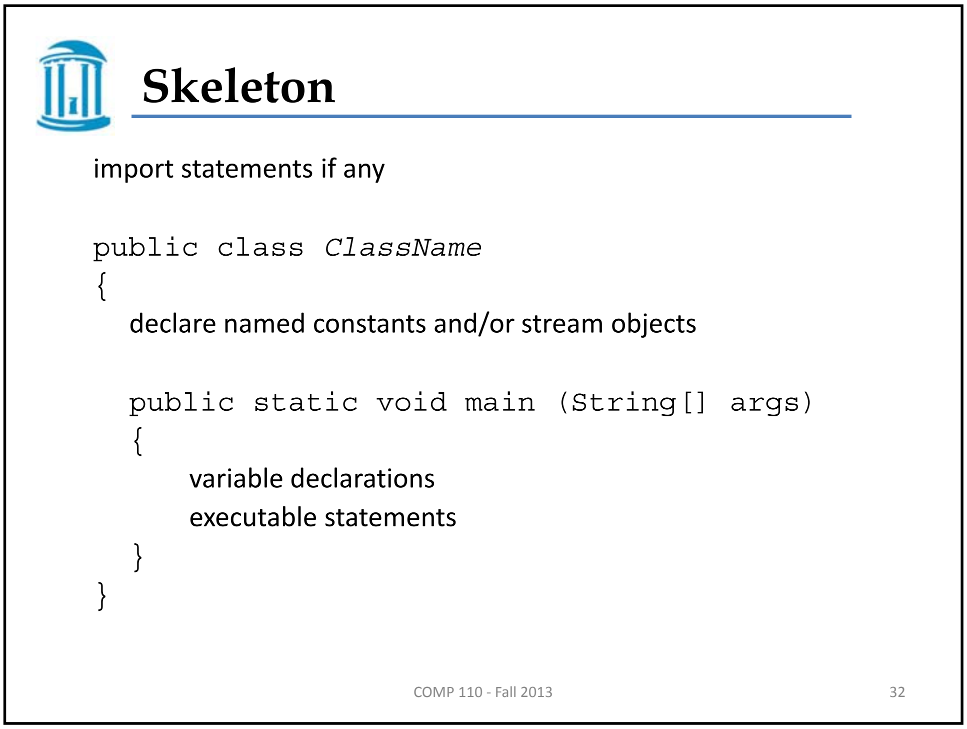


The `main` method

- Heading

  `public static void main (String[] args)`

- Body
  - statements enclosed by { }
  - declaration statements
    - used to declare things such as variables
  - executable statements
    - perform calculations, manipulate data, create output, accept input, etc.



Skeleton

```
import statements if any

public class ClassName
{
   declare named constants and/or stream objects

   public static void main (String[] args)
   {
        variable declarations
        executable statements
   }
}
```

Import Statement

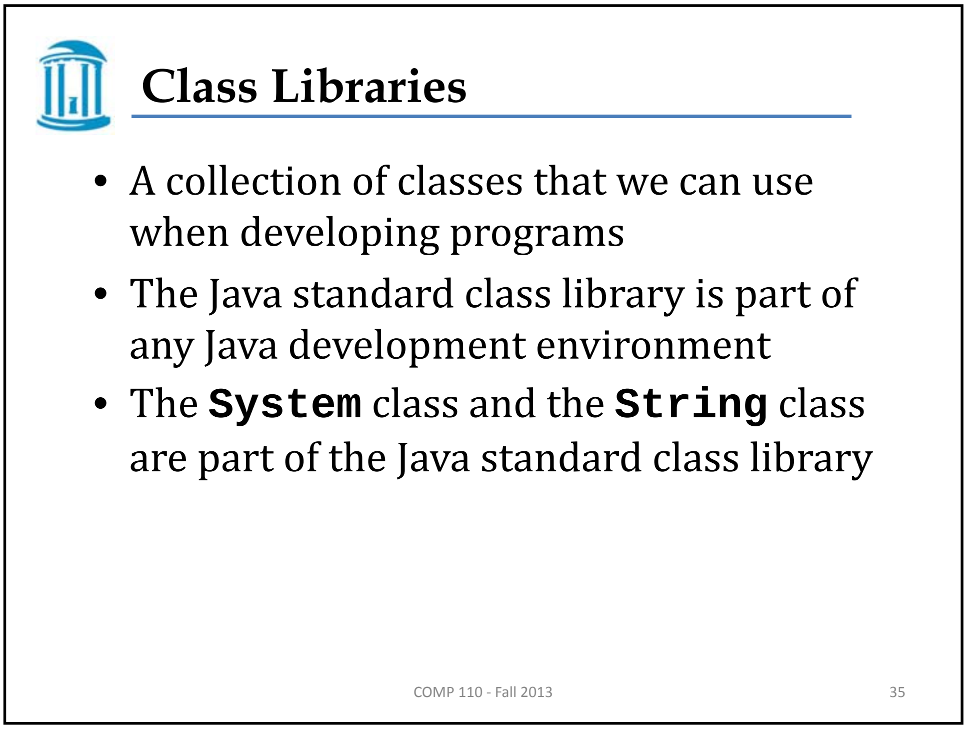
- Tell the compiler which packages are used in the program
- Import statements and program statements constitute the *source code*

- Source code saved in a file with the extension .java
- Source code file must have the same name as the class with the `main` method

Import Packages

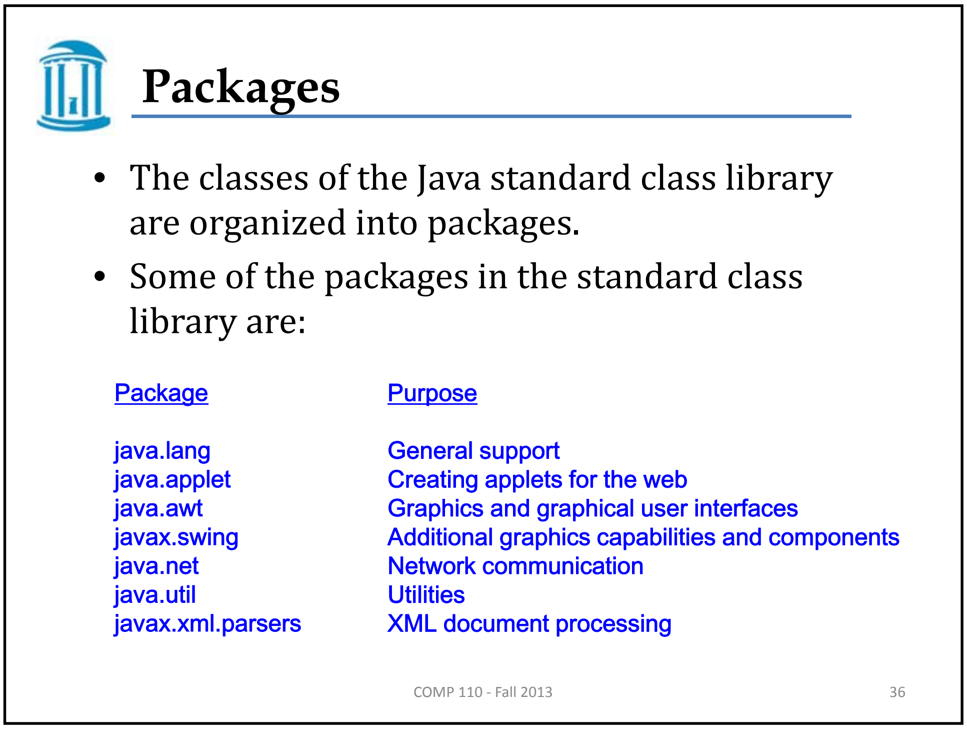
```
import java.util.Scanner;
```

- **Import = borrow something from somewhere else**
- Package = Library of tools
  - *java.util* is a package that contains useful standard tools
    - ***java.math*** contains mathematical tools
    - ***java.net*** contains network connection tools
  - Scanner is one standard tool about keyboard input in this package



Class Libraries

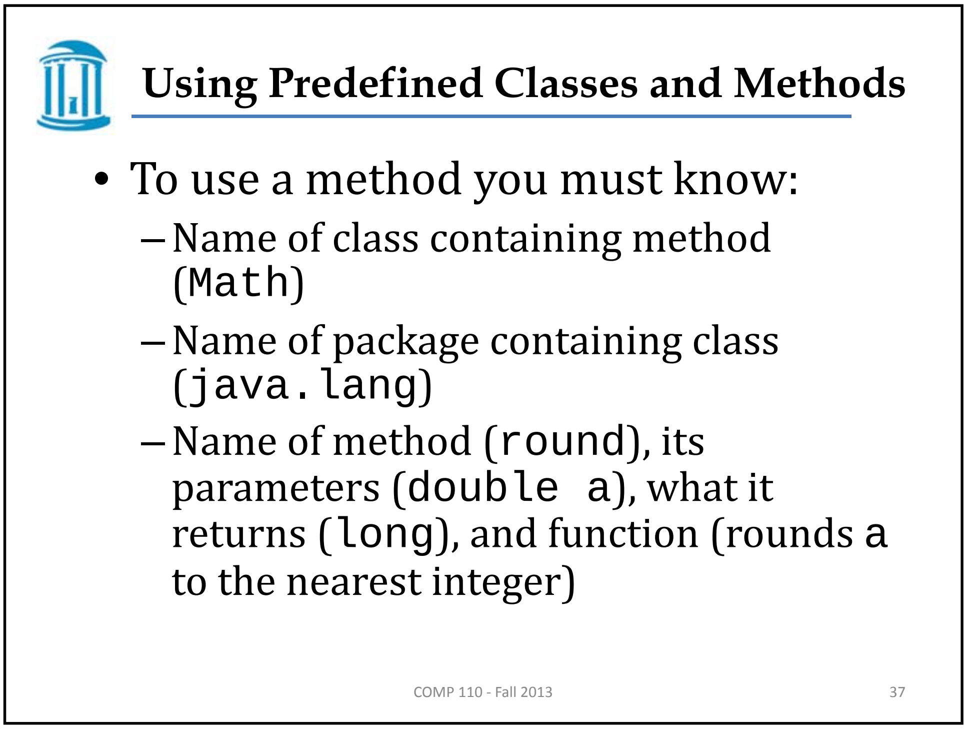
- A collection of classes that we can use when developing programs
- The Java standard class library is part of any Java development environment
- The **`System`** class and the **`String`** class are part of the Java standard class library



Packages

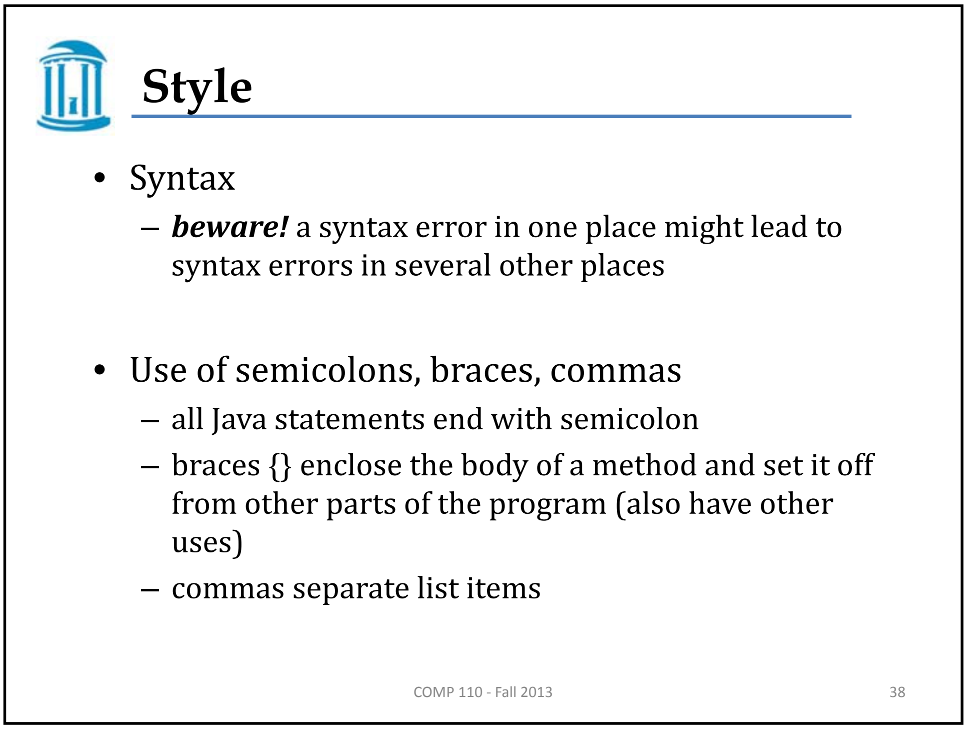
- The classes of the Java standard class library are organized into packages.
- Some of the packages in the standard class library are:

Package	Purpose
java.lang	General support
java.applet	Creating applets for the web
java.awt	Graphics and graphical user interfaces
javax.swing	Additional graphics capabilities and components
java.net	Network communication
java.util	Utilities
javax.xml.parsers	XML document processing



Using Predefined Classes and Methods

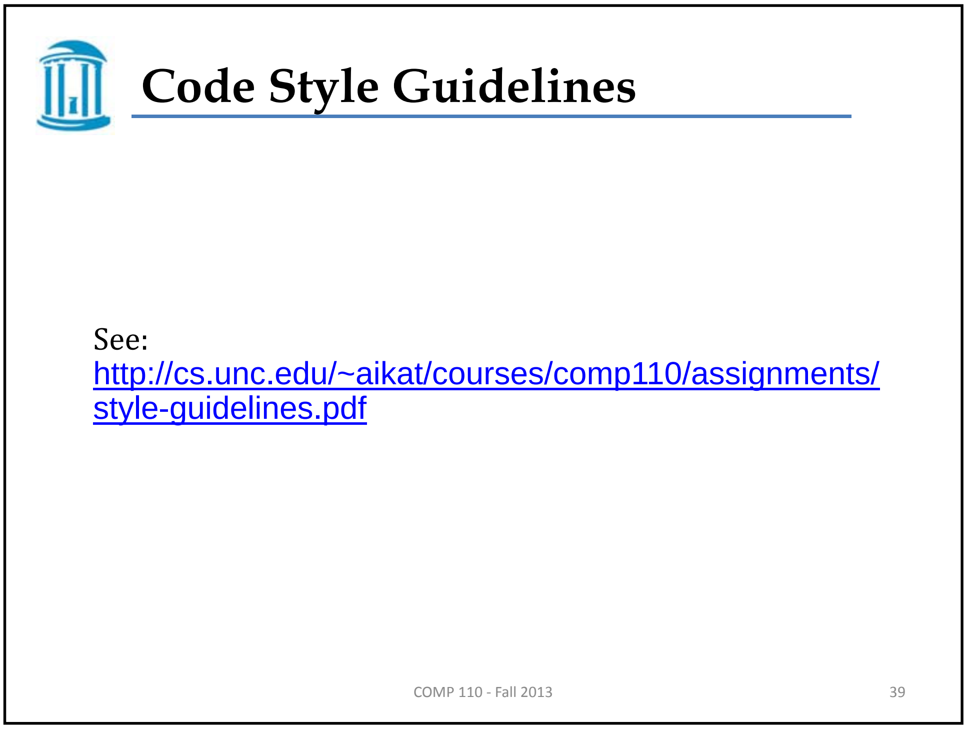
- To use a method you must know:
  - Name of class containing method (`Math`)
  - Name of package containing class (`java.lang`)
  - Name of method (`round`), its parameters (`double a`), what it returns (`long`), and function (rounds `a` to the nearest integer)



Style

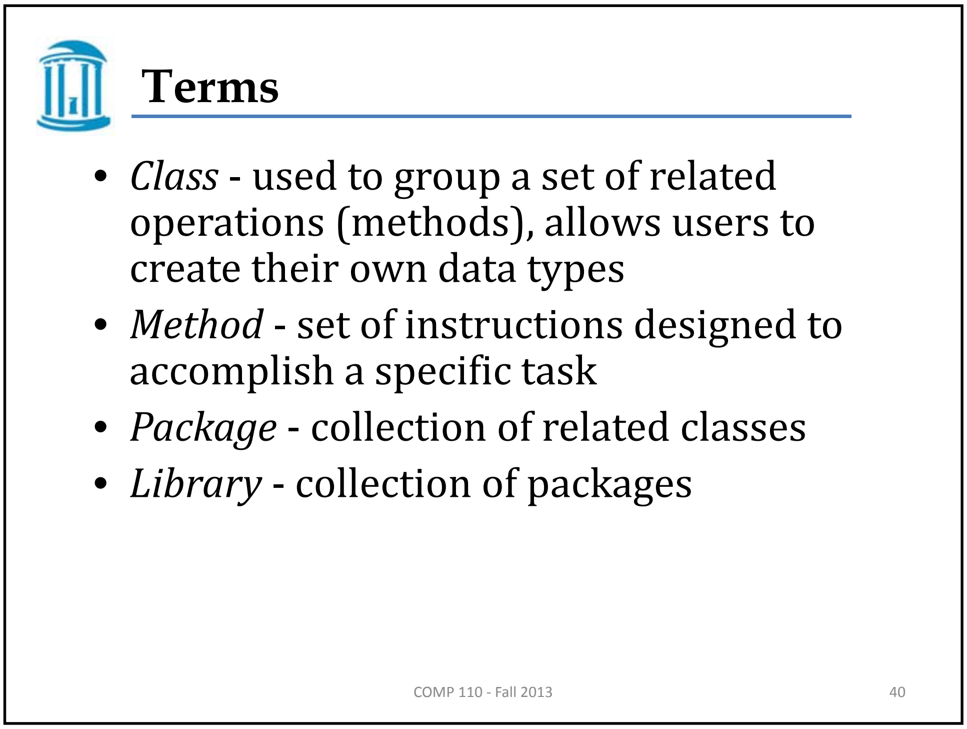
- Syntax
  - ***beware!*** a syntax error in one place might lead to syntax errors in several other places

- Use of semicolons, braces, commas
  - all Java statements end with semicolon
  - braces {} enclose the body of a method and set it off from other parts of the program (also have other uses)
  - commas separate list items



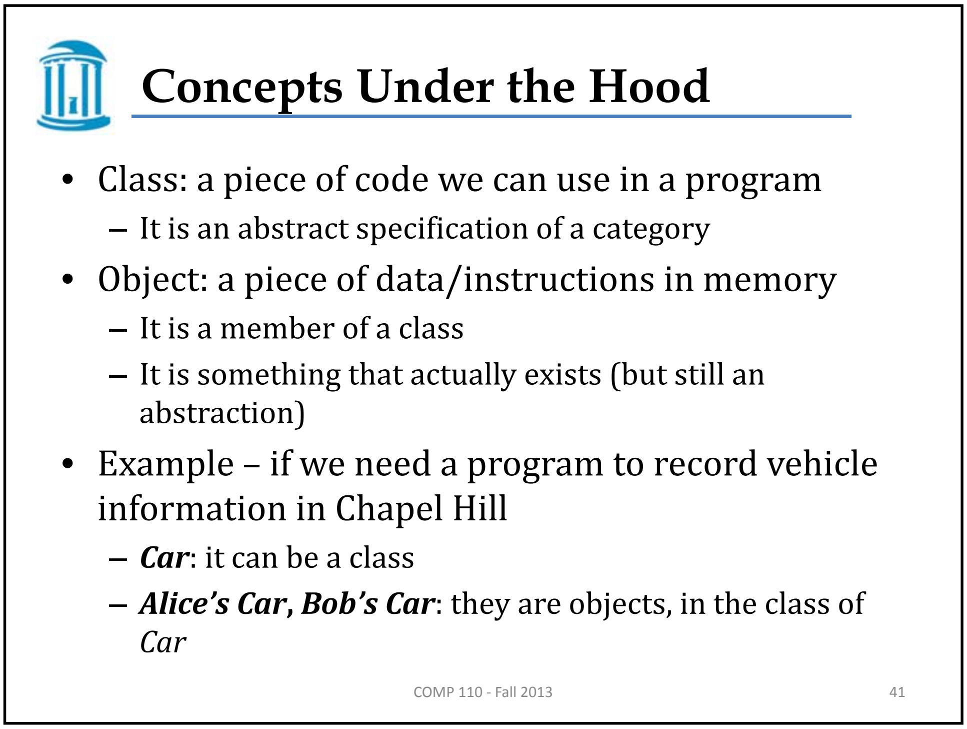
Code Style Guidelines

See:
http://cs.unc.edu/~aikat/courses/comp110/assignments/style-guidelines.pdf



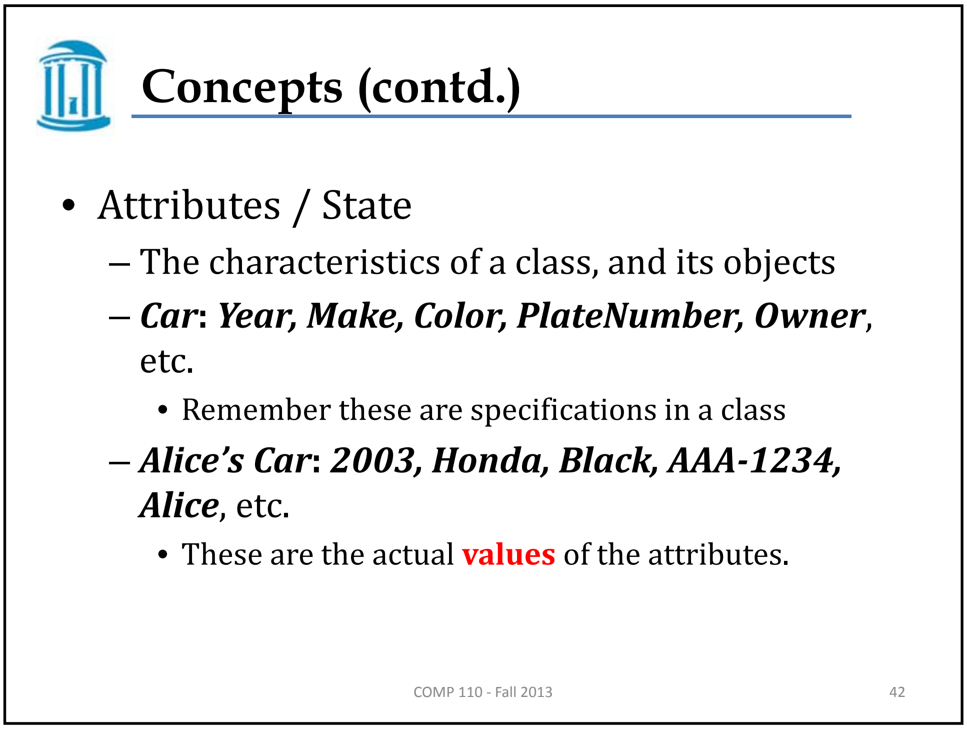
Terms

- *Class* - used to group a set of related operations (methods), allows users to create their own data types
- *Method* - set of instructions designed to accomplish a specific task
- *Package* - collection of related classes
- *Library* - collection of packages



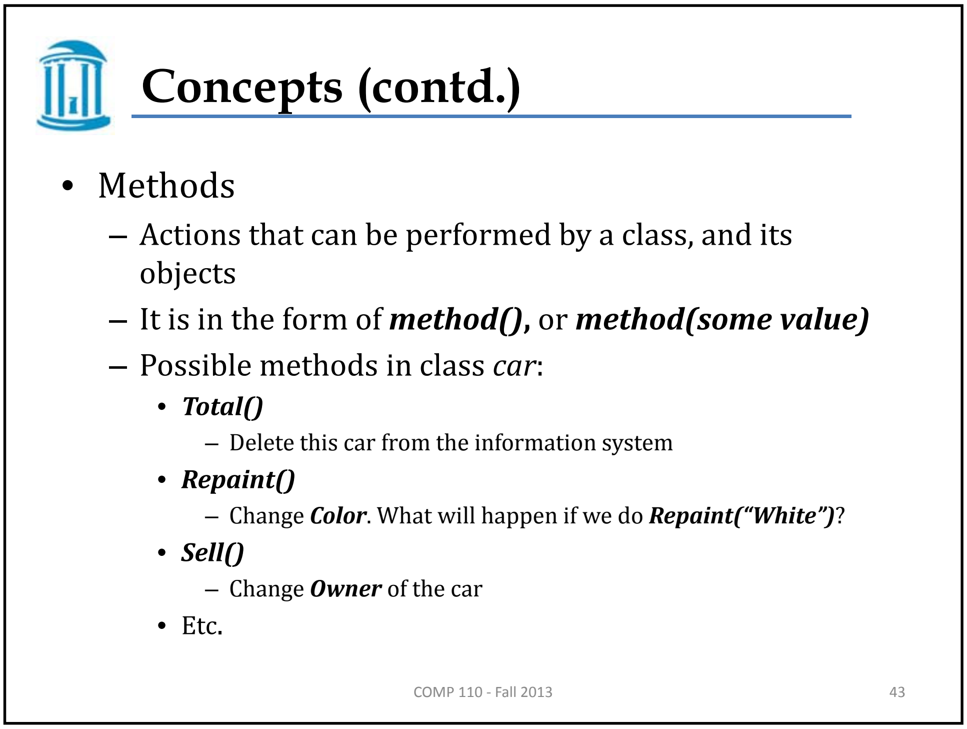
Concepts Under the Hood

- Class: a piece of code we can use in a program
  - It is an abstract specification of a category
- Object: a piece of data/instructions in memory
  - It is a member of a class
  - It is something that actually exists (but still an abstraction)
- Example – if we need a program to record vehicle information in Chapel Hill
  - ***Car***: it can be a class
  - ***Alice's Car***, ***Bob's Car***: they are objects, in the class of *Car*



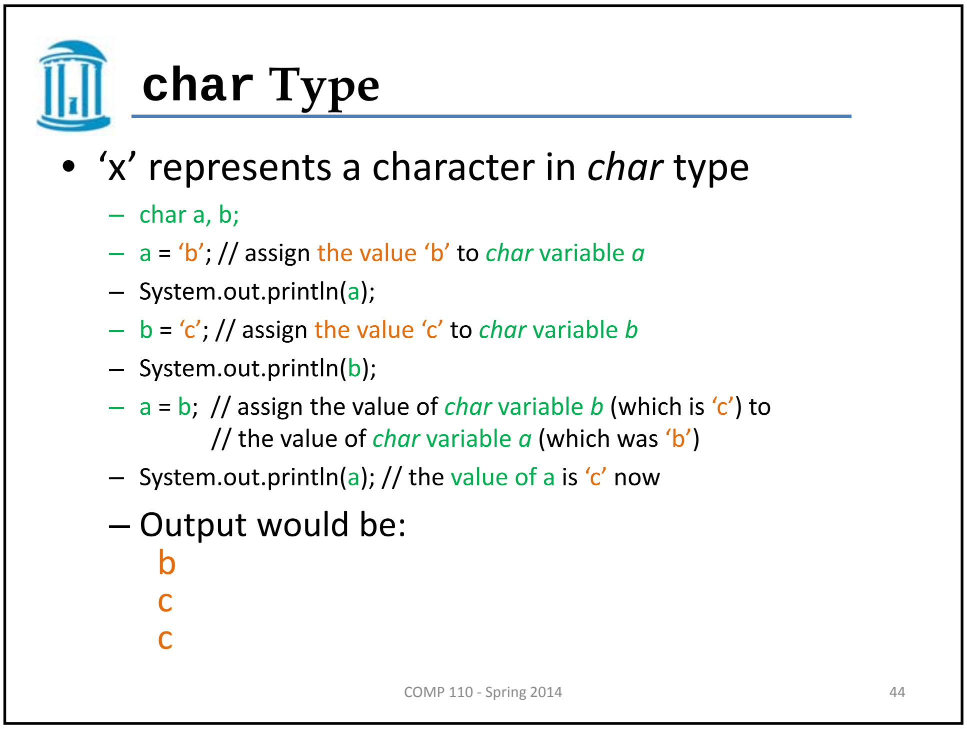
Concepts (contd.)

- Attributes / State
  - The characteristics of a class, and its objects
  - ***Car***: ***Year, Make, Color, PlateNumber, Owner***, etc.
    - Remember these are specifications in a class
  - ***Alice's Car***: ***2003, Honda, Black, AAA-1234, Alice***, etc.
    - These are the actual **values** of the attributes.



Concepts (contd.)

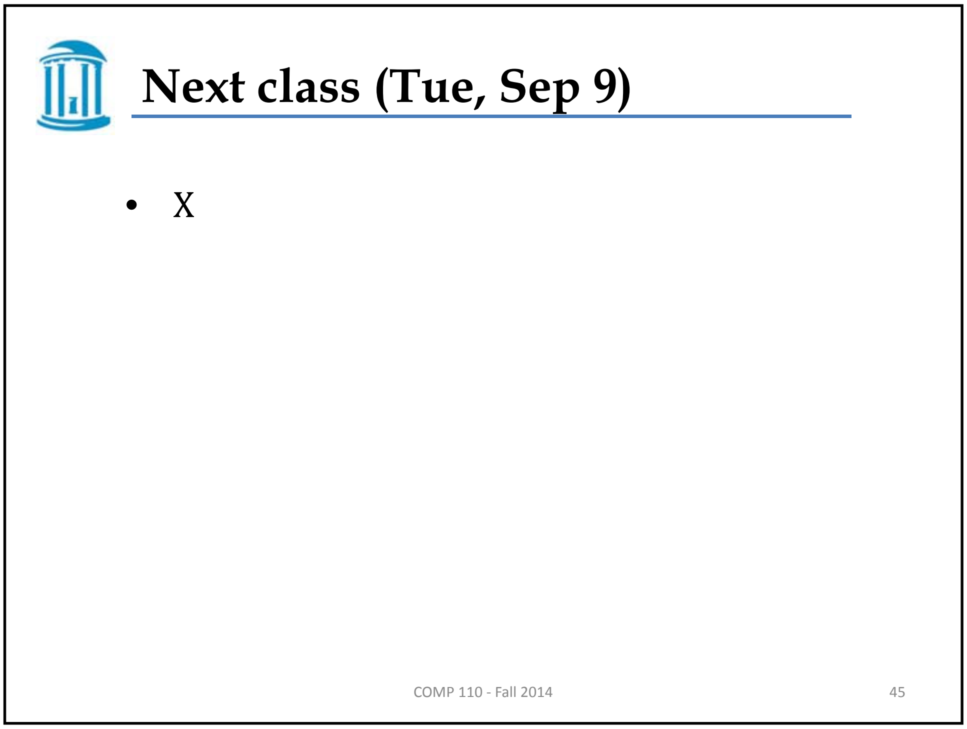
- Methods
  - Actions that can be performed by a class, and its objects
  - It is in the form of ***method()*,** or ***method(some value)***
  - Possible methods in class *car*:
    - ***Total()***
      - Delete this car from the information system
    - ***Repaint()***
      - Change ***Color***. What will happen if we do ***Repaint("White")***?
    - ***Sell()***
      - Change ***Owner*** of the car
    - Etc.



char Type

- 'x' represents a character in *char* type
  - char a, b;
  - a = 'b'; // assign the value 'b' to *char* variable *a*
  - System.out.println(a);
  - b = 'c'; // assign the value 'c' to *char* variable *b*
  - System.out.println(b);
  - a = b; // assign the value of *char* variable *b* (which is 'c') to
    // the value of *char* variable *a* (which was 'b')
  - System.out.println(a); // the value of a is 'c' now
  - Output would be:

```
b
c
c
```



Next class (Tue, Sep 9)

- X