COMP 110-003
Introduction to Programming
Primitive Types, Strings and Console I/O

January 22, 2013

Haohan Li

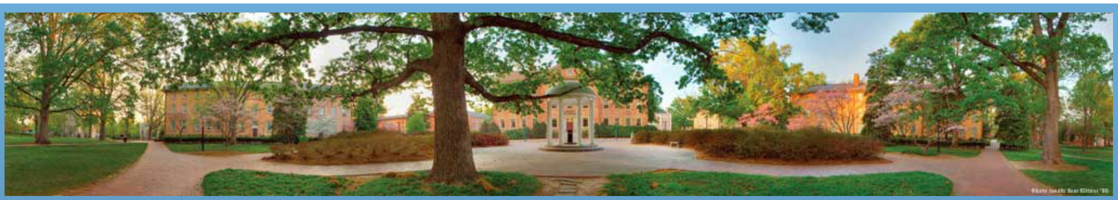
TR 11:00 – 12:15, SN 011

Spring 2013

THE UNIVERSITY
of NORTH CAROLINA
at CHAPEL HILL

Daily Joke

- Q: How did the programmer die in the shower?
  A: He read the shampoo bottle instructions:
  Lather, Rinse, Repeat.



Daily Joke

- Q: How did the programmer die in the shower?
  A: He read the shampoo bottle instructions:
  Lather, Rinse, Repeat.

Use one word to tell:
what is this in the world of a programmer?

Daily Joke

- Q: How did the programmer die in the shower?
  A: He read the shampoo bottle instructions:
  Lather, Rinse, Repeat.

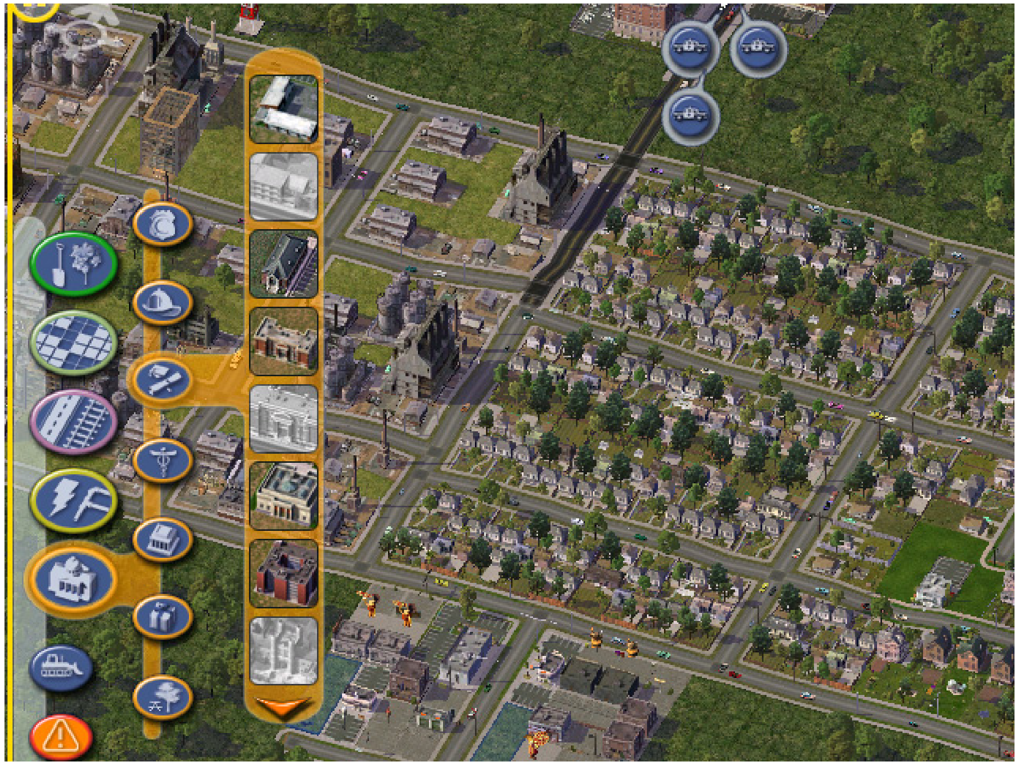
  Use one word to tell:
  what is this in the world of a programmer?
  - Algorithm: A set of instructions for solving a problem
  - What's the problem with this algorithm?

Review

- Classes/Objects
- Attributes
- Methods
- Object-Oriented Programming/ Procedural Programming

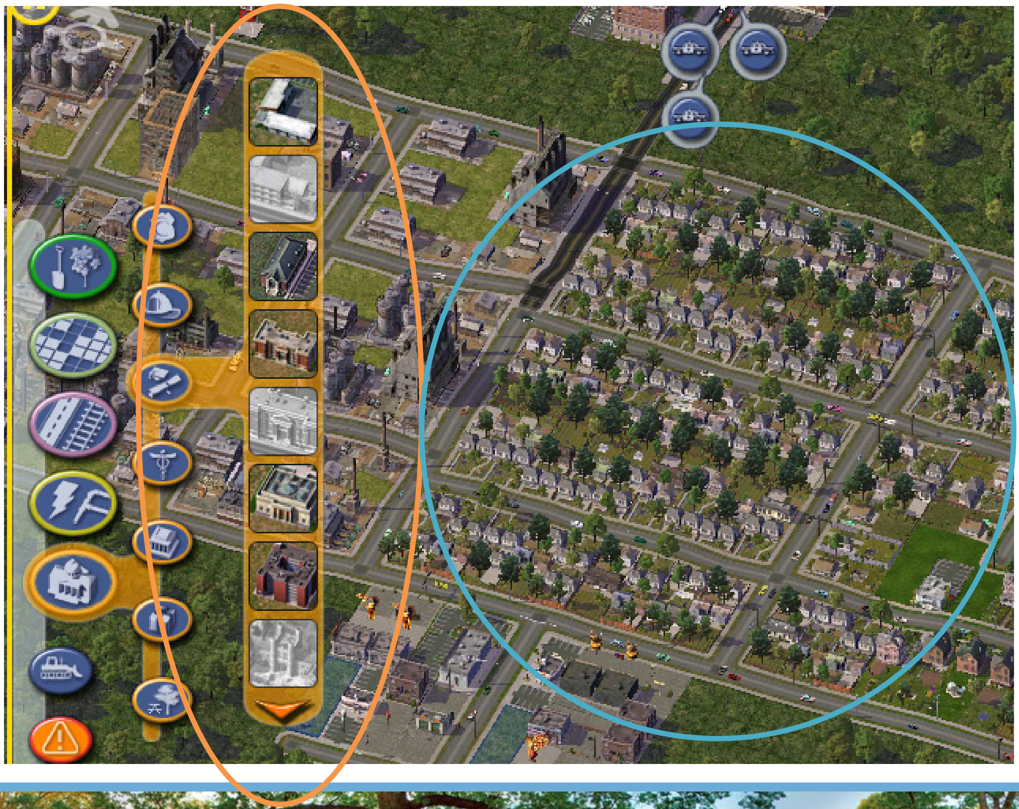
SimCity 4

- If we want to simulate such a city in computer
  - What can be the
    - Classes
    - Objects
    - Attributes
    - Methods
  - Start with class/object
  - We'll brainstorm



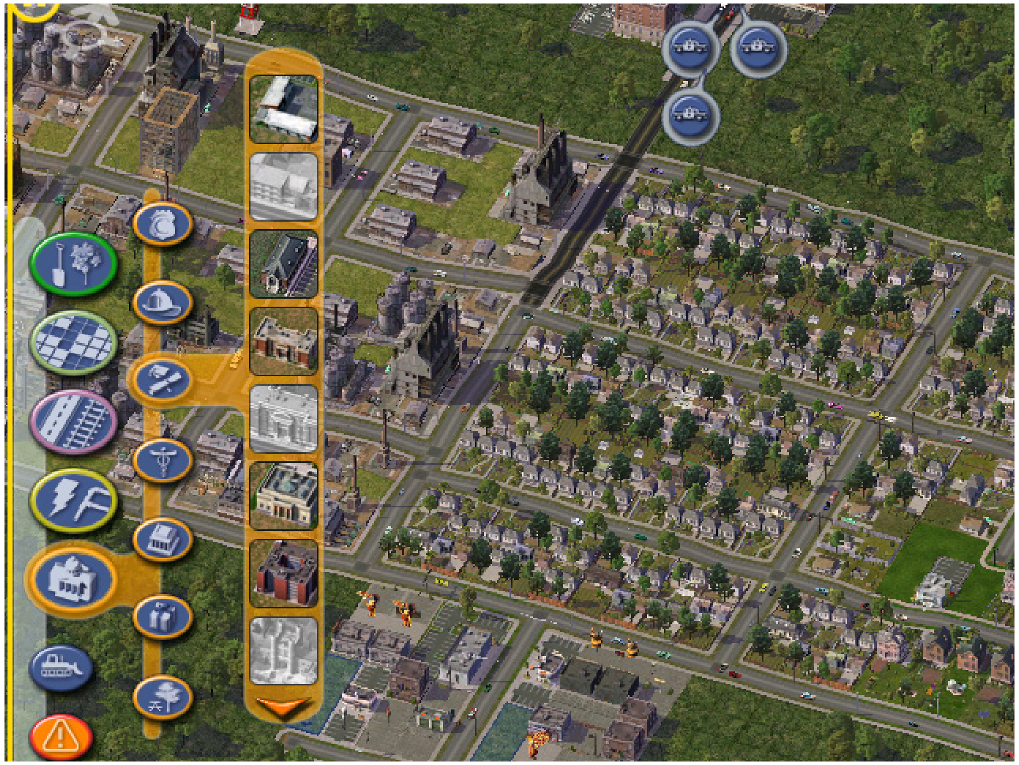
Classes vs. Objects

- **Classes:**
  - What we can create
- **Objects**:
  - What have been created



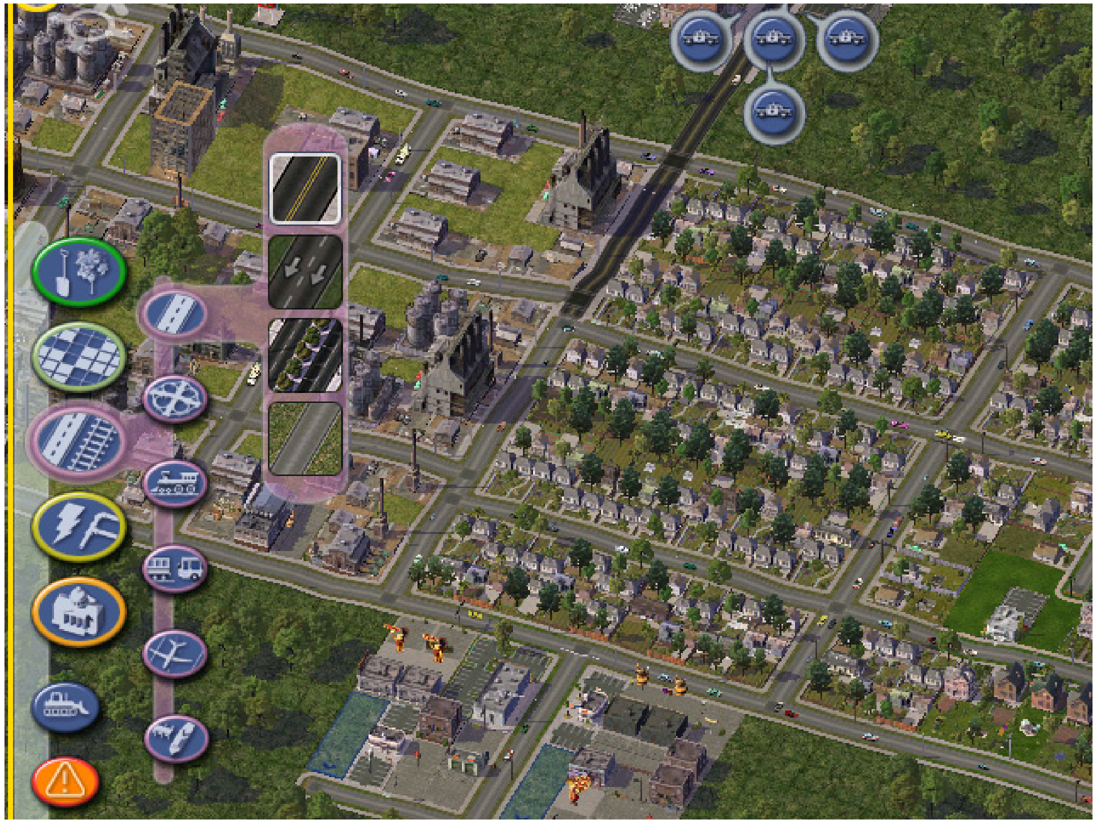
Classes and Objects

- Classes
  - House
  - Market
  - Power plant
  - Hospital
- Objects?
  - Those on the ground



More Classes

- More classes
  - Road
  - Rail
  - Bridge



More Classes

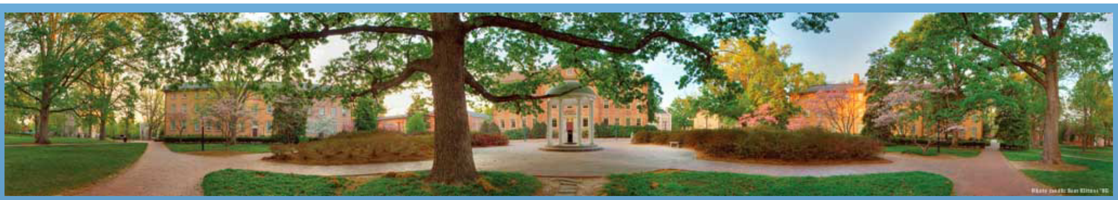
- More Classes
  - Tree
  - Grass
  - Animal
  - Car
  - People

More Class?

- Ground!
  - Ground class can be big or small
- If it's big
  - One object with many attributes
- If it's small
  - Many objects with several attributes

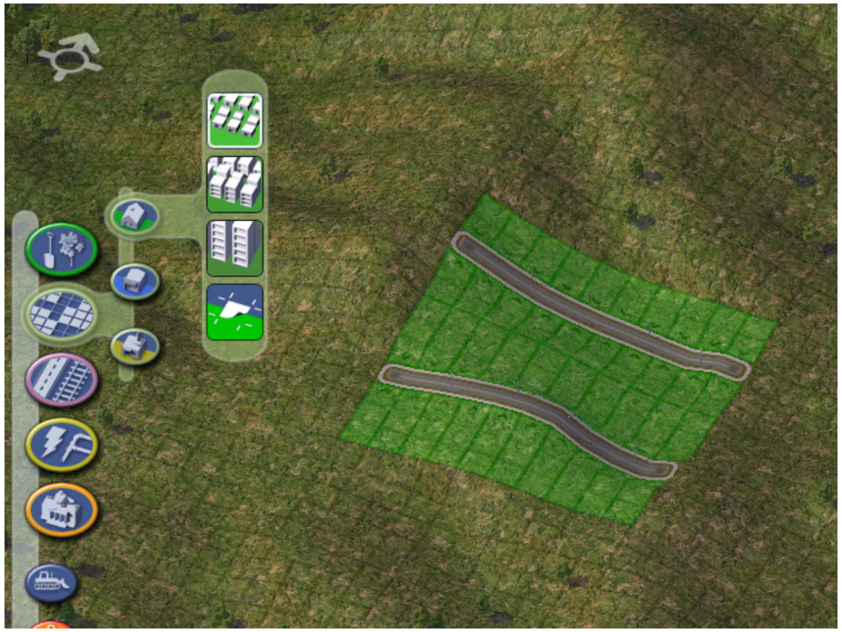
Attributes?

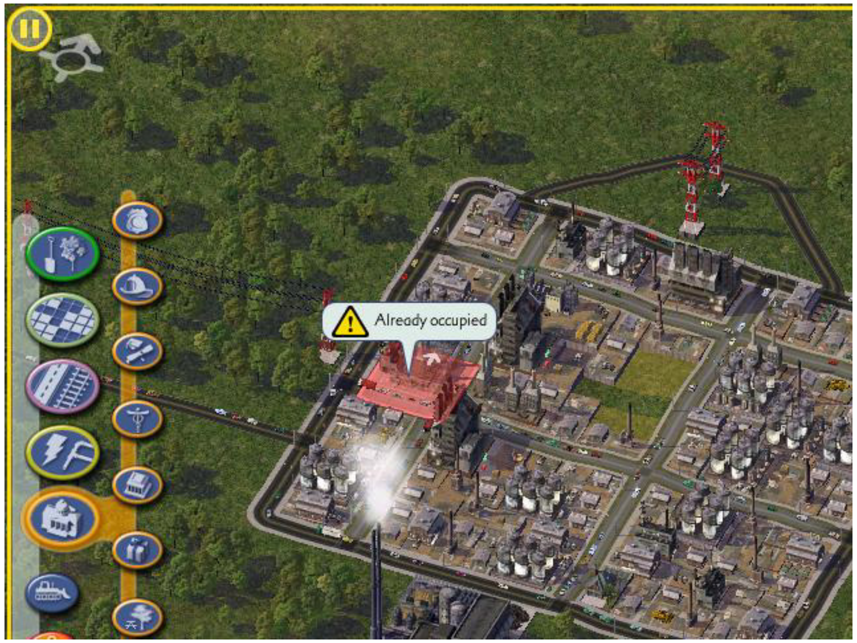
- There can be a lot of them
  - Name
  - Style
  - Area size
  - Location
  - etc.

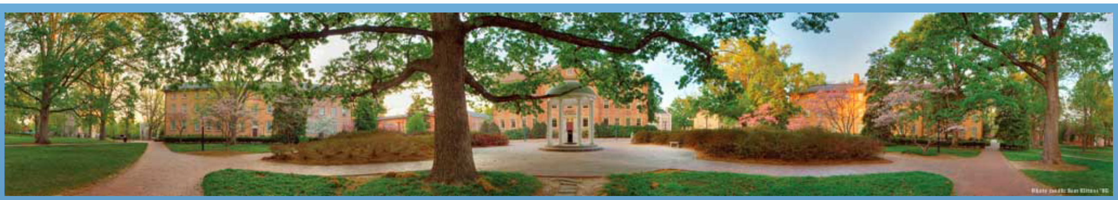


Methods?

- Display()
- PutOnGround()

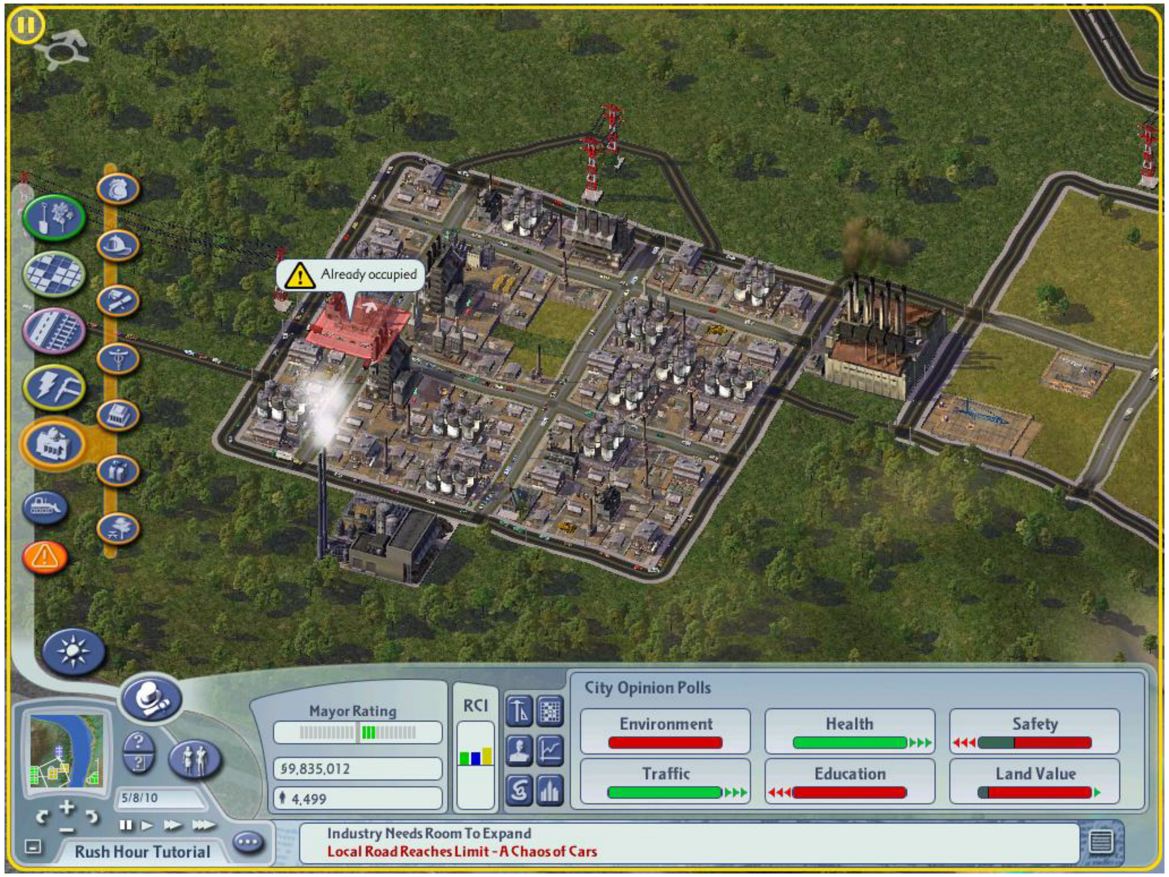






More Methods?

- AttractPeople()
- AffectEnviron()
- SupplyWater()
- SupplyPower()
- EarnMoney()
- PayTax()
- etc.

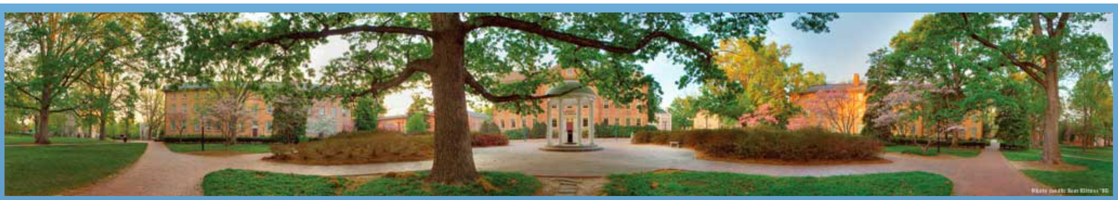


OOP vs Procedural Programming?

- Hard to perform simulation of a city using procedural programming
- We will use SimCity as an OOP example in future

Today

- Primitive Types and Expressions
- Strings
- Console I/O

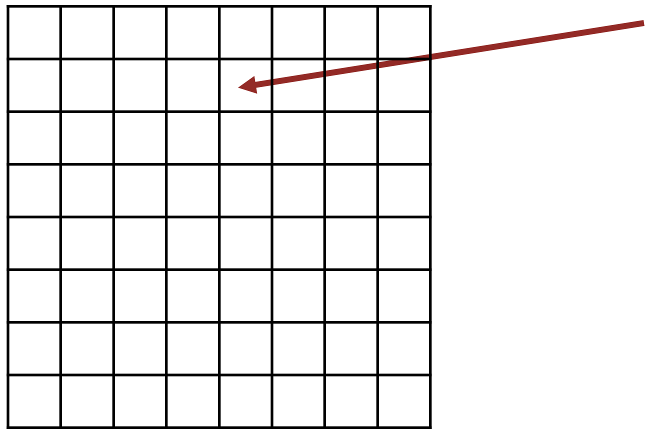


Variables

- Used to store data in program
- The data currently in a variable is its **value**
- Name of variable is an **identifier**
- Can change value throughout program
- Choose variable names that are helpful
  - amount, quarters, dimes, nickels

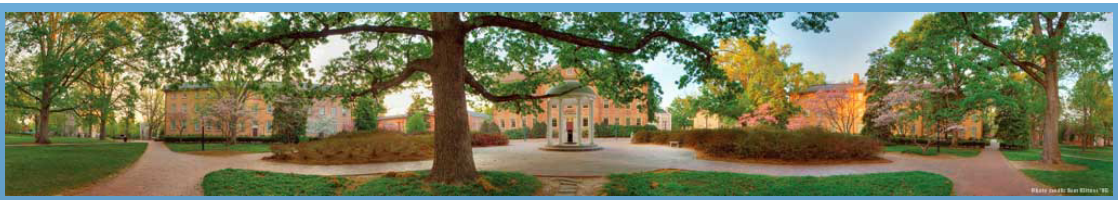
Variables and Memory

- A variable corresponds to a location in memory



variable: amount

- Use this cell to store the value of total money amount
- Prevent this cell from being used by other variables later



How to use variables

- **Declare** a variable
- **Assign** a value to the variable
- **Change** the value of the variable

Variable Declaration

- **Syntax:**
  - *type variable_1, variable_2, …;*
- **Examples:**
  - *int count, score, myInt;*
  - *char letter;*
  - *double totalCost, ratio;*

How to name an identifier

- Naming rules:
  - Letters, digits(0-9)
  - First character *cannot* be a digit
  - No spaces
- Java is case sensitive
- Legal names
  - pinkFloyd, b3atles, eyeColor
- Illegal names
  - michael.bolton, kenny-G, 1CP

Keywords

- Reserved words with predefined meanings
- You *cannot* name your variables keywords
- Inside cover of the textbook
- if, else, return, new

Type

- What kind of value the variable can hold
  - Primitive type - indecomposable values
    - Names usually begin with lowercase letters
    - int, double, char, boolean
    - See inside cover of the textbook
  - Class type - objects with both data and methods
    - Names usually begin with uppercase letter
    - Scanner, String

Primitive Types

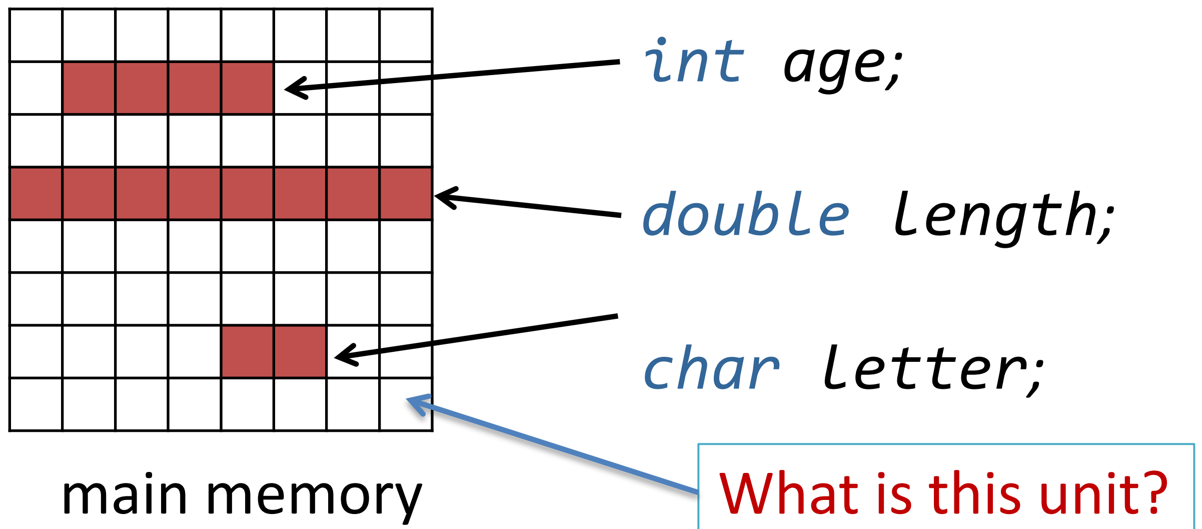
- Integer (byte, short, int, long)
  - 0, -3, 5, 43
- Floating-point number (float, double)
  - 0.5, 12.4863, -4.3
- Characters (char)
  - A, r, %, T
- Boolean (boolean)
  - true, false

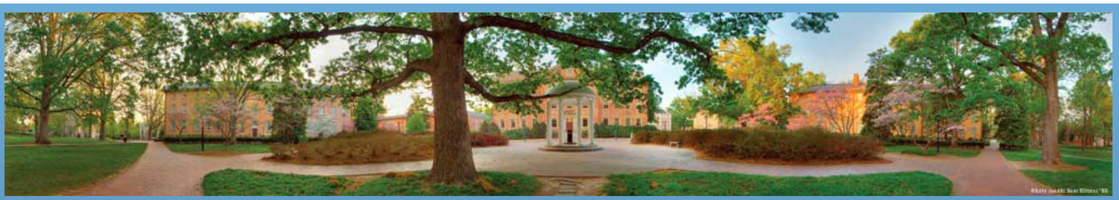
Integer and Floating-Point

- Floating-point vs fixed-point
  - 120000000000, 0.0000000000325 are fixed-point
  - 1.2 x 10^{11}, 3.25 x 10^{-11} are floating-point
    - In computer, you only have to save 12 and 11, or 325 and -11
- Integer is an exact value
- Floating-point is an approximation value
  - Consider: 1/3 = 0.333333333333333333333333……
  - No way to save this exact number in finite memory

Variables and Memory

- When declaring a variable, a certain amount of memory is assigned/allocated based on the declared primitive type



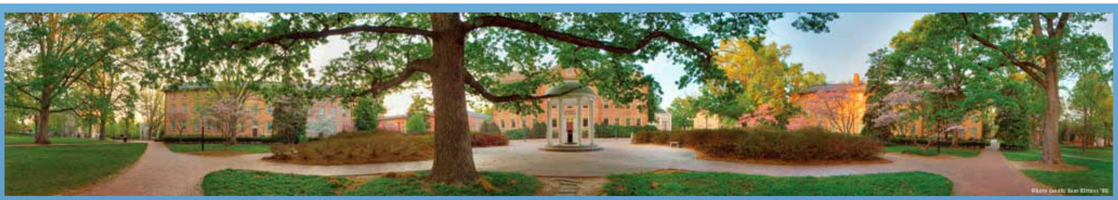


Assign and Change Variables

- *int changingVar = 0;*
  - Declare and assign value
  - *type variable = value;*
- *changingVar = 5;*
  - Assign/change value, variable must be declared before
  - *variable = value;*
- *changingVar = changingVar + 4;*
  - Can refer to itself
  - It means newValue = oldValue + 4. Now changingVar = ?

Assignment Statements

- Change a variable's value
  Syntax:
  - variable = expression;
- Example:
  - *sleepNeeded = 8;*
  - *sleepDesired = sleepNeeded * 2;*



Behind the Statement

- variable = expression;
  - CPU calculates the value of the expression.
  - Send the value to the location of variable.
- *sleepDesired = sleepNeeded * 2;*
  - Calculate sleepNeeded * 2
    - Get the current value of sleepNeeded from its memory location
  - Assign the value to the location of sleepDesired

Special Assignment Operators

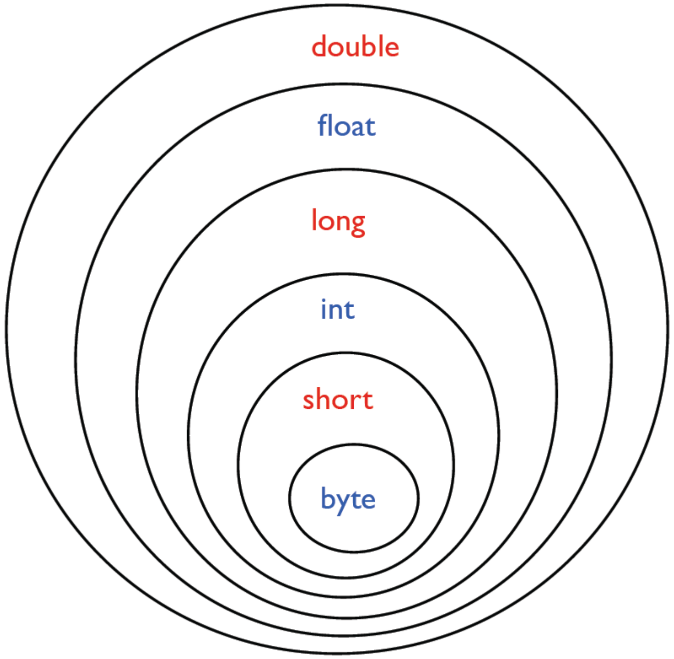
- Some operators are new to you
  - *total += 5; // is the same as*
  - *total = total + 5;*
  - *count++; // is the same as*
  - *count = count + 1;*
- They are created because
  - It's shorter
  - Less possibility of making mistakes

Assignment Compatibilities

- Usually, we need to put values of a certain type into variables of the same type.
  - Put integer into int, floating-point into double, etc.
- However, in some cases, the value will automatically be converted when types are different
  - *int age= 10;*
  - *double average = age;*

Assignment Compatibilities

- You can only put small things into bigger things
  - byte->short->int->long->float->double
- Some examples
  - ~~*myShort = myInt*~~*;* Wrong
  - ~~*myByte = myLong*~~*;* Wrong
  - *myFloat = mybyte;* Right
  - *myLong = myInt;* Right



Type Casting

- You can ask the computer to change the type of values which are against the compatibility.
  - ~~*myFloat = myDouble;*~~
  - ~~*myByte = myInt*~~*;*
  - ~~*myShort = myFloat*~~*;*
  - *myFloat = (float)myDouble;*
  - *myByte = (byte)myInt;*
  - *myShort = (short)myFloat;*
- It means you know the risk but you still want to change
- You may lose information

Arithmetic Operators

- Unary operators (more info later)
  - +, -, ++, --, !
- Binary arithmetic operators
  - *, /, %, +, -
    - *rate*rate + delta*
    - *1/(time + 3*mass)*
    - *(a - 7)/(t + 9*v)*

Modular Arithmetic - %

- Remainder
  - 7 % 3 = 1   (7 / 3 = 2, remainder 1)
  - 8 % 3 = 2   (8 / 3 = 2, remainder 2)
  - 9 % 3 = 0   (9 / 3 = 3, remainder 0)
- "clock arithmetic"
  - Minutes on a clock are mod 60

Parentheses and Precedence

- Expressions inside parentheses evaluated first
  - `(cost + tax) * discount`
  - `cost + (tax * discount)`
- Precedence order:
  - First: the unary operators: ++, --, !
  - Second: the binary arithmetic operators: *, /, %
  - Third: the binary arithmetic operators: +, -
- In the same level, from left to right

Parentheses and Precedence

- These are the same:
  - *total = cost + tax * discount;*
  - *total = cost + (tax * discount);*
    - The highest precedence level is marked in red
- Probably we wanted:
  - *total = (cost + tax) * discount;*
- Full operator precedence table on back cover

Errors

- Syntax error – grammatical mistake in your program
  - *int n3 = 10, //* use a ‘;’ instead of a ‘,’
  - Eclipse can only detect this level of error
- Run-time error – an error that is detected during program execution
  - *int n3 = n1 / n2; //* But n2 == 0
- Logic error – a mistake in a program caused by the underlying algorithm
  - *int n3 = n1 - n2; //* But we meant to sum

Strings

- No primitive type for strings in Java
  - Instead, Java provides a class called String
- “Text” is a value. You can declare String variables
  - *String month = “May”;*
    - Similar to: *int n1 = 10;*
  - *System.out.println(month);*
    - month is a variable. Its value is “May”
- So it prints: May

String Concatenation

- We use “+” to connect multiple strings
  - *String month = “May”;*
  - *String sentence = “This month is ” **+** month;*
  - *System.out.println(sentence);*
  - It will print: This month is May
- Moreover, “+” can be used to connect String and other types
  - *int quarters = 3;*
  - *System.out.println(quarters + " quarters");*

String (Class type)

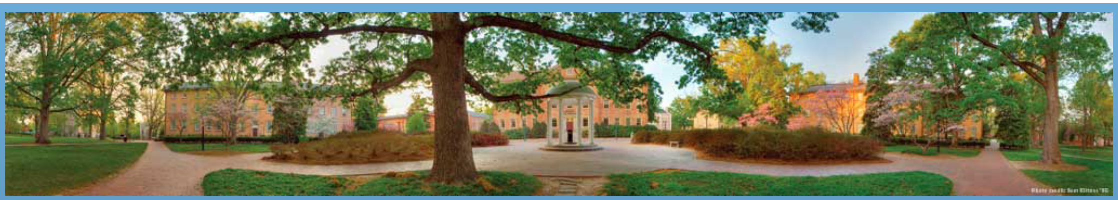
- Class types have methods

Object

```
String myString = “COMP110”;
int len = myString.length();
```

Method

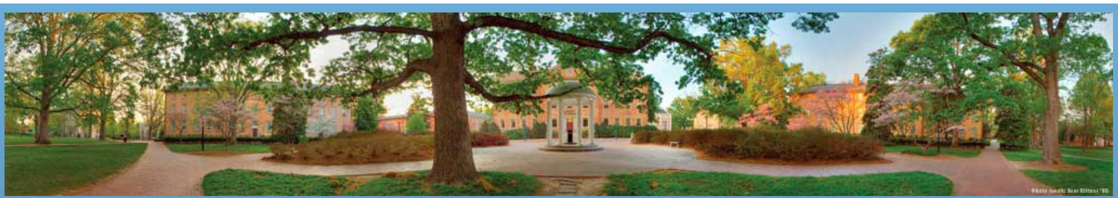
- *len* will be equal to 7



Strings Methods (Figure 2.5)

- *myString.length();*
- *myString.equals("a string");*
- *myString.toLowerCase();*
- *myString.trim();*

- You will see these in Lab 2



String Indices

U	N	C		i	s		G	r	e	a	t
0	1	2	3	4	5	6	7	8	9	10	11

```
String output = myString.substring(1, 8);
System.out.println(output);
```

It will print: NC is G

String Indices

U	N	C		i	s		G	r	e	a	t
0	1	2	3	4	5	6	7	8	9	10	11

```
String output = myString.substring(1, 8);
System.out.println(output);
```

It will print: NC is G **WHY?**

String Indices

U	N	C		i	s		G	r	e	a	t
0	1	2	3	4	5	6	7	8	9	10	11

It's easy to output a specified length.

```
String output = myString.substring(1, 8);
System.out.println(output);
```

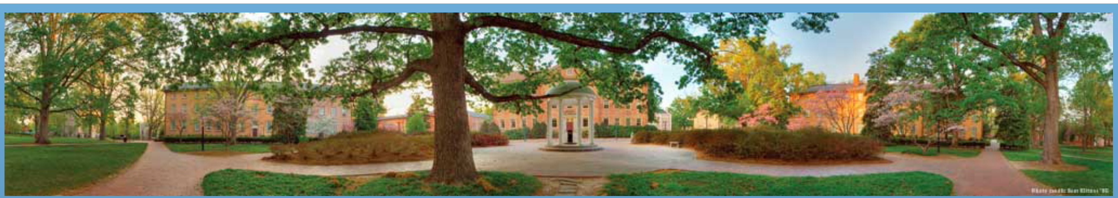
It will print: NC is G **WHY?**

Put Quotes in a String

- What to do if you want to output
  - How do I put “quotes” in my string?
- You will have trouble!
  - *System.out.println(“How do I put ”quotes“ in my string?”);*
- You have to let computer know that you want the quote marks to be in the String
  - *System.out.println(“How do I put \“quotes\” in my string?”);*

But what about 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

I/O (Input/Output)

- *System.out.print("this is a string");*
- *System.out.println("this is a string");*
- What is the difference?
  - println() method advances to a new line after it displays its output, whereas the print() method does not
  - Instead: *System.out.print("this is a string\n");*

Keyboard Input

- Use the Scanner class
  - Scanner *Scanner_object_name* = new Scanner(System.in);
  - *Scanner_object_name*.nextLine();
  - *Scanner_object_name*.nextInt();
  - *Scanner_object_name*.nextDouble();
- Make sure to read Chapter 2.3, and the **Gotcha** before Figure 2.7

Program 1

- No collaboration privilege
  - You are allowed to
    - Talk about textbook, notes and Java features
    - Talk to understand the program requirement
    - Let others see your program's problem only if
      - You've written a complete section of code but it's not working
      - The one who helps has finished his/her own code
      - The one who helps only tells you where the problem is and you will fix it all by yourself
      - Key point: **it must be your idea and your code**
      - If you get help on how to do something you don't have an idea, it's very easy to produce similar codes