THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

COMP 110
Introduction to Programming

Fall 2015

Time: TR 9:30 – 10:45

Room: AR 121 (Hanes Art Center)

Jay Aikat

FB 314, *aikat@cs.unc.edu*

Previous Class

- What did we discuss?

COMP SCI OPEN HOUSE - 5 PM TODAY

- Are you wondering about what computer science might offer you? What careers it might open?
- Come to our open house this Thursday, August 27 5-7 PM in Sitterson 014. We'll introduce you to the opportunities in the department, the careers computer science opens, and let you in on what makes our department so much fun. After our presentations, there will be pizza and a presentation sponsored by Credit Suisse.
- Come and discover why the class you are in now should not be your last computer science class!

Today

- Announcements
  - Labs / recitations
  - Assignment0: **due Fri, Aug 28 at 11:55 PM**
- Your first program

Documentation and Style

- Meaningful Names
- Comments
- Indentation
- Named Constants

Documentation and Style

- Most programs are modified over time to respond to new requirements.
- Programs which are easy to read and understand are easy to modify.
- Even if it will be used only once, you have to read it in order to debug it .

Meaningful Variable Names

- A variable's name should suggest its use.
- Observe conventions in choosing names for variables.
  - Use only letters and digits.
  - "Punctuate" using uppercase letters at word boundaries (e.g. **taxRate**).
  - Start variables with lowercase letters.
  - Start class names with uppercase letters.

Comments

- The best programs are self-documenting.
  - Clean style
  - Well-chosen names
- Comments are written into a program as needed explain the program.
  - They are useful to the programmer, but they are ignored by the compiler.

Comments

- A comment can begin with //.
- Everything after these symbols and to the end of the line is treated as a comment and is ignored by the compiler.

```
double radius; //in centimeters
```


Comments

- A comment can begin with /* and end with */
- Everything between these symbols is treated as a comment and is ignored by the compiler.

```
/*
This program should only
be used on alternate Thursdays,
except during leap years, when it should
only be used on alternate Tuesdays.
*/
```

Comments

- A *javadoc* comment, begins with `/**` and ends with `*/`
- It can be extracted automatically from Java software.

```
/** this is how a comment is
  written in Java; it can run into
  multiple lines like this! */
```


When to Use Comments

- Begin each program file with an explanatory comment
  - What the program does
  - The name of the author
  - Contact information for the author
  - Date of the last modification.
- Provide only those comments which the expected reader of the program file will need in order to understand it.

Indentation

- Indentation should communicate nesting clearly.
- A good choice is four spaces for each level of indentation.
- Indentation should be consistent.
- Indentation should be used for second and subsequent lines of statements which do not fit on a single line.

Indentation

- Indentation does not change the behavior of the program.
- Proper indentation helps communicate to the human reader the nested structures of the program

Storing Data in Memory

1. Instruct the computer to allocate memory (*declaration*)
   - *how much?* based on data type
   - *how to access?* name the memory location

2. Include statements in the program to put data into the allocated memory
   - *assign* a particular value to a particular memory location
   - *initialization* - putting the first (initial) value into the memory location

Types of Storage

- Named Constant
  - once the value is stored in memory, it can't be changed

- Variable
  - the value stored in memory can be changed (can *vary*)

Named Constant

```
static final dataType IDENTIFIER = value;
```

- Declared by using the reserved word final
- Always *initialized* when it is declared
- Identifier should be in ALL CAPS, separate words with underscore (_)
- Example:
  - 1 inch is always 2.54 centimeters

```
final double CM_PER_INCH = 2.54;
```


Variable

```
dataType identifier;
```

- Must be declared before it can be used
- Can be (but doesn't have to be) initialized when declared
- Identifier should start in lowercase, indicate separate words with uppercase
- Example:
  - number of students in class

```
int numStudents;
```

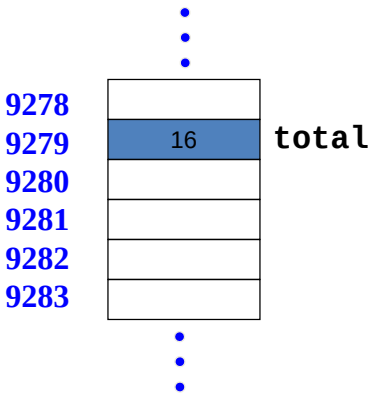
- Multiple variables (of the same data type) can be declared on a single line

```
int numStudents, numGrades, total;
```

Variable

- A variable can be given an initial value in the declaration
- When a variable is referenced in a program, its current value is used

```
int total = 16;
int base = 32, max = 149;
```



Naming Variables

- Variables should be descriptively named
- We should be able to tell what type of information the variable holds by its name
- ***Remember Java is case-sensitive!***

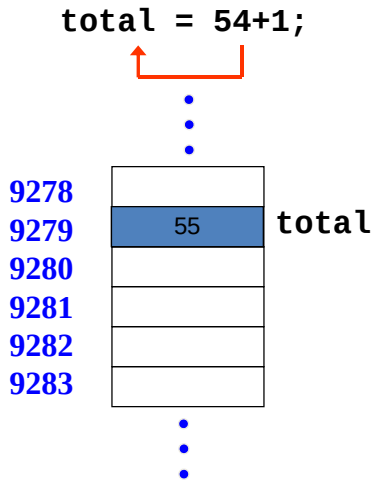
Assignment

```
variable = expresssion;
```

- Assignment Operator (=)
- *expression* can be a value (3) or a mathematical expression (2 + 1)
- The *expression* must evaluate to the same data type as the variable was declared

Assignment

- assignment statement
  - changes the value of a variable
- Expression on right is evaluated and result is stored in variable on the left
- Value that was in total is overwritten



Assignment

- The assignment operator has a lower precedence than the arithmetic operators

First the expression on the right hand side of the = operator is evaluated

```
answer  =  sum / 4 + MAX * lowest;
           4     1   3     2
```

Then the result is stored in the variable on the left hand side

Assignment

- The right and left hand sides of an assignment statement can contain the same variable

First, one is added to the original value of count

```
count  =  count + 1;
```

Then the result is stored back into count (overwriting the original value)

Questions

What is stored in the memory location referred to by the identifier num after each of the following statements is executed in order?

```
int num;
num = 3;
num = 5 + 4 - 2;
num = num * 2;
num = 3.4 + 5;
```

nothing
3
7
14
14

would give an error since 3.4 + 5 would not result in an int

Questions

Write Java statements to accomplish the following:

1. Declare `int` variables `x` and `y`

```
int x, y;
```

2. Update the value of an `int` variable x by adding 5 to it

```
x = x + 5;
```

3. Declare and initialize an `int` variable `x` to 10 and a `char` variable `ch` to 'B'

```
int x = 10;
char ch = 'B';
```

4. Declare `int` variables to store four integers

```
int num1, num2, num3, num4;
```

The Value of Variables

- We can use variables to save the result of expressions for use later in our programs.

```
int sum, count;
double average;

sum = 3 + 4 + 5;
count = 3;
average = (double) sum / count;
```


Questions

Which of the following are valid Java assignment statements? Assume that `i`, `x`, and `percent` have been declared as `double` variables and properly initialized.

```
1. i = i + 5;
2. x + 2 = x;
3. x = 2.5 * x;
4. percent = 10%
```

valid

invalid

valid

invalid

Exercise – Add two numbers

```
public class addTwoNumbers {
    public static void main(String[] args) {

    int num1, num2, total;

    num1 = 10;
    num2 = 20;

    total = num1 + num2;

    System.out.println("The sum of those two numbers is");
    System.out.println(total);
    }
}
```


Next class (Tue, Sep 1)

- Binary representation
- More programming in class

→ Reading Assignment: Chapter 1