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?

Today

- Announcements
  - Still registering…
  - Labs / recitations – register for them
  - Join Piazza: piazza.com/unc/fall2015/comp110
  - Assignment1: **due Fri, Aug 28 at 11:55 PM**
  - **Check Sakai, Piazza and class webpage regularly**
- Your first program

Identifiers

- Names of things (variables, constants, methods) in your programs
- Can be composed of any combination of letters, digits, underscore (_), and dollar sign ($)
- Cannot begin with a digit
- May be any length
- Java is ***case-sensitive***
  - Total, total, and TOTAL are different identifiers

Illegal Identifiers

Identifier	Description
`employee Salary`	There can be no space between `employee` and `Salary`.
`Hello!`	The exclamation mark cannot be used in an identifier.
`one+two`	The symbol + cannot be used in an identifier.
`2nd`	An identifier cannot begin with a digit.

Questions

Classify the following as legal or illegal identifiers:

1. My First Program — illegal
2. my1stProgram — legal
3. 1stProgram — illegal
4. $money — legal
5. an_identifier — legal
6. Jane'sProgram — illegal

Primitive Data Types

What is a Data Type?

- *Primitive data* are fundamental values such as numbers and characters
- A set of values and the operations that can be performed on those values
- Operations are performed on primitive types using built-in operators

Primitive Data Types

- 8 primitive data types in Java
  - 4 represent integers
    - **byte, short, int, long**
  - 2 represent floating point numbers
    - **float, double**
  - 1 represents characters
    - **char**
  - 1 represents boolean values
    - **boolean**

Primitive Data Types (Numeric Types)

- The difference between the various numeric primitive types is their size, and therefore the values they can store:

Type	Storage	Min Value	Max Value
byte	8 bits	-128	127
short	16 bits	-32,768	32,767
int	32 bits	-2,147,483,648	2,147,483,647
long	64 bits	$< -9 \times 10^{18}$	$> 9 \times 10^{18}$
float	32 bits	$+/- 3.4 \times 10^{38}$ with 7 significant digits	
double	64 bits	$+/- 1.7 \times 10^{308}$ with 15 significant digits	

Integers

- Examples: -6728, -67, 0, 78, 36782
- Positive integers do not have a '+' sign in front of them (but they can)
- No commas are used in an integer
  - commas in Java are used to separate items in a list

Primitive Data Types (Characters)

- A **char** stores a single character from the *Unicode character set*
  - an ordered list of characters, and each character corresponds to a unique number
  - uses 16 bits per character, allowing for 65,536 unique characters
- Character literals are delimited by single quotes:

```
'a'   'X'   '7'   ' '   '$'   ','   '\n'
```

newline character
(we'll discuss later)

Primitive Data Types (Boolean)

- Only two valid values
  - true or false
  - uses 1 bit for storage
- Represent any situation that has 2 states
  - on - off
  - true - false
- **true** and **false** are reserved words

Arithmetic Expressions

- *Expression* - a combination of one or more operands and their operators
- *Arithmetic expressions* compute numeric results and make use of the arithmetic operators:

Addition	**+**
Subtraction	**-**
Multiplication	*****
Division	**/**
Remainder	**%**

- If either or both operands associated with an arithmetic operator are floating point, the result is a floating point

Division and Remainder

- If both operands to the division operator (/) are integers, the result is an integer (the fractional part is discarded)

```
14 / 3      equals?   4

8 / 12      equals?   0
```

- The remainder, or **mod**ulus, operator (%) returns the remainder after dividing the second operand into the first (only works with integer types)

```
14 % 3      equals?   2

8 % 12      equals?   8
```

Unary vs. Binary Operators

- Unary operators
  - has only one operand
  - example: - (negative, not subtraction)
    -5
- Binary operators
  - has two operands
  - example: - (subtraction)
    5 - 3

Operator Precedence

- Determines the order in which operators are evaluated:
  1. multiplication, division, and remainder
  2. addition, subtraction, and string concatenation
  3. arithmetic operators with the same precedence are evaluated from left to right
- Parentheses can be used to force the evaluation order (just like in math)

Operator Precedence (PEMDAS)

- Parentheses: 6 * (5 + 7) vs. 6 * 5 + 7
- Exponents (powers, roots – 2^5 $36^{1/2}$)
- Multiplication / Division / Mod
- Addition / Subtraction
- Left to right
- Which is these is correct?

  30 / 5 * 3 = 6 * 3 = 18 ← **this one!**

  30 / 5 * 3 = 30 / 15 = 2

Operator Precedence

- What is the order of evaluation in the following expressions?

```
a + b + c + d + e        a + b * c - d / e
  1   2   3   4            3   1   4   2

      a / (b + c) - d % e
        2     1     4   3

    a / (b * (c + (d - e)))
      4     3     2     1
```

Integral Expressions

- All operands are integers
- Result is an integer

- Examples:

  2 + 3 * 5

  3 + x – y / 7

  x + 2 * (y – z) + 18

Floating-point Expressions

- All operands are floating-point numbers
- Result is a floating-point

- Examples:

  12.8 * 17.5 – 34.50

  x * 10.5 + y - 16.2

  7.0 / 3.5

Mixed Expressions

- Operands of different types
- Examples:
  2 + 3.5
  6 / 4 + 3.9
- Integer operands yield an integer result
- Floating-point operands yield a floating-point result
- If both types of operands are present, the result is a floating-point number
  - **implicit type coercion**
- Precedence rules are followed

Type Conversion (Casting)

- Used to avoid implicit type coercion
- Syntax
  (*dataTypeName*) *expression*
- *Expression* evaluated first, then type converted to *dataTypeName*
- Examples:
  (**int**) (7.9 + 6.7) = 14
  (**int**) (7.9) + (**int**)(6.7) = 13

Questions

1. (5 + 4) % 6 — 9 % 6, 3
2. (5 + 6) % 3.5 — 11 % 3.5, not possible
3. (double) (13) / 2 — 13.0 / 2, 6.5
4. (double) (13 / 2) — (double) (6), 6.0

Our First Program

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

Next class (Thu, Aug 27)

- Binary representation
- Program in class: Adding two numbers
- Assignment1 DUE Fri, Aug 28

→ Reading Assignment: Chapter 1

Teaching Assistants

- Yenchun Chen
- Junpyo Hong (JP)
- Ben Newton

Ben Newton

Junpyo Hong (JP)

Teaching Assistants

Dana Elhertani

Spencer Byers

Sarah White

Camden Link

Jeffrey Young

Max Daum