Designing Programs & Primitive Types

COMP 110
Summer II 2012

Peter Lincoln
6/22/2012
Announcements

- We’ve moved
  - FB009 is the new room for the remainder of the class

- Extra Office Hours Today
  - After Class
  - Come if you don’t have Java & Eclipse working
Questions?

Yesterday:

- Introduction to Programming
  - Class Syllabus

- Computer Basics
  - Hardware & Memory
  - Programs & Compiling
  - Your First Program
Today in COMP 110

- Lecture
  - Writing Algorithms as Pseudocode
  - Variables & Primitive Types
  - Operators

- Lab 0
Algorithm

- A set of instructions for solving a problem
Fun Exercise Time

- PB&J Sandwich Algorithm
  - The class-developed algorithm is posted to the course website.
Pseudocode

- A combination of code and English, used to express an algorithm
  - Write before coding to design the algorithm
  - Write after coding to explain the code
import java.util.*;

public class FirstProgram
{
    public static void main(String[] args)
    {
        System.out.println("Hello out there.");
        System.out.println("I will add two numbers for you.");
        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 those two numbers is");
        System.out.println(n1 + n2);
    }
}
Pseudocode Example: Algorithm

1. Prompt the user to enter two whole numbers
2. Read in two whole numbers from the keyboard and store them as n1 and n2
3. Output the value of n1 + n2
Pseudocode Example 2

- There are many writing styles for pseudocode, this one is more code-like:

```plaintext
CONTAINS-DUPLICATES(X[1..n]: sequence<T>, H: T ➞ integer) : boolean
  define shared tbl: hashtable<T> of capacity c·n
  define shared k: boolean
  define local e: hashtable<T>.cell, b: integer
  forall i in 1..n do
    while not CONTAINS(tbl, X[i], i) do
      INSERT(tbl, X[i], i)
    od
  od
  k := false
  forall i in 1..c·n do
    e := tbl[i]
    if CONTAINS-DUPLICATES-SCANS(e) then
      k := true
    fi
  od
  return k
```

- Do not worry about the details of this particular algorithm, it is just an example.
Variables

- Used to store data in a program
  - Variable: a storage location in memory
  - Value: the data currently in the variable
  - Identifier: the name of a variable

Declare Variable: \texttt{n1}

Assign \texttt{5} to \texttt{n1}

Main Memory
Variables

- Used to store data in a program
  - Variable: a storage location in memory
  - Value: the data currently in the variable
  - Identifier: the name of a variable

- Values can be changed throughout a program
- Choose variable names that are meaningful!
Variable Declarations

- Syntax:
  - Type variable_1, variable_2, ...

- Examples
  - int count, score, total;
  - char gradeLetter;
  - double totalCost, ratio;
  - int zipgok; // Not a useful name
Variable Identifiers (Names)

- Composed of letters, digits (0-9), and underscores (_)
  - First character cannot be a digit

- Legal Names
  - pinkFloyd, the_coup, b3atles

- Illegal Names
  - michael.bolton, kenny-G, 1CP

- Java is case sensitive
  - pinkFloyd and pinkfloyd are two different variables
Keywords

• Keyword: a reserved word with a predefined meaning

• Cannot use keywords for variable names
  • Examples: if, else, return, new
  • A complete list is in the front inside cover of the textbook
Variable Types

- The kind of value the variable can hold

- Two main forms of types:
  - Primitive Types: indecomposable values
    - Names begin with lowercase letters and are keywords
    - Examples: byte, short, int, long, float, double, char, boolean
  - Class Types: composed of other variables and/or methods
    - Names begin with uppercase letters
    - Examples: Scanner, String
Primitive Types

- Integers (byte, short, int, long)
  - 0, -3, 5, 43

- Floating-Point Numbers (float, double)
  - 0.5, 12.4863, -4.3

- Characters (char)
  - A, r, %, T

- Boolean (boolean)
  - true, false
Variables and Memory

- Different variable types
  - Have different ranges (min, max)
  - Have different precision levels (decimal places)
  - Consume different amounts of memory when declared
  - Full details on the front inside cover of the textbook

```java
int age;
double length;
char letter;
```
Assignment Statements

- Changes a variable’s value

- Syntax:
  
  ```
  variable = expression;
  ```

- Examples:
  
  ```
  sleepNeeded = 8;
  sleepDesired = sleepNeeded * 2;
  sleepObtained = sleepNeeded / 2;
  ```
Behind the Assignment Statement

- \texttt{variable} = \texttt{expression};
  - CPU computes the value of \texttt{expression}
  - Stores the value in the memory location of \texttt{variable}

- \texttt{sleepDesired} = \texttt{sleepNeeded} \times 2;
  - Calculate \texttt{sleepNeeded} \times 2
    - Loads the value of \texttt{sleepNeeded} from its memory location
  - Assigns the value to the location of \texttt{sleepDesired}
Combining Declaration & Assignment

- It is possible to combine declaring a new variable with assigning in a value
  - It often improves clarity
  - Until a value is assigned, the variable has no readable value

```
int count;
count = 55;
```

Equivalent

```
int count = 55;
```
Operators

- Computes a result based on 1, 2 or 3 values

- Common Arithmetic Operators (integer and floating-point values)
  - Unary Operators
    - +, -, ++, --
    - (Type)
  - Binary Operators
    - *, /, %, +, -
    - =, *=, /=, %=, +=, -=
  - Ternary Operators will be covered later

- A complete list is presented in the back-cover of the textbook
Basic Math Operators

- **Unary**: +, -
- **Binary**: *, /, +, -
  - These behave like their math counterparts

^  
- A binary operator, but not for arithmetic
- Will be covered in a later class
Basic Math Operator: %

- Modular (mod) or clock arithmetic
  - Minutes on a clock are mod 60

- Remainder
  - \( 7 \mod 3 = 1 \) (\( 7 / 3 = 2 \), remainder 1)
  - \( 8 \mod 3 = 2 \) (\( 8 / 3 = 2 \), remainder 2)
  - \( 9 \mod 3 = 0 \) (\( 9 / 3 = 3 \), remainder 0)
Specialized Assignment Operators

- **Unary**: `++`, `--`
- **Binary**: `*=`, `/=`, `%=`, `+=`, `-=`

Shorthand for commonly performed operations

**Examples:**

- `length = length * 5;`
  - `length *= 5;`

- `age = age + 1;`
  - `age += 1;`
  - `age++;`
Type Casting

- Usually values of a particular type are put into variables of the same type
- If not, then in some cases the value will be automatically converted (implicit type cast)

Example:
```java
int age;
age = 10;
double length;
length = age;
```

- Auto-cast: byte → short → int → long → float → double
Type Casting Operator

- **Auto-cast:** byte ➔ short ➔ int ➔ long ➔ float ➔ double

- If a cast in the opposite direction is required, then the casting operator is required

- **Illegal (compiler error):**
  - myFloat = myDouble;
  - myByte = myInt;
  - myShort = myFloat;

- **Legal:**
  - myFloat = (float)myDouble;
  - myByte = (byte)myInt;
  - myShort = (short)myFloat;
Parentheses & Precedence

- Expressions within parentheses are evaluated first
  - $(\text{cost} + \text{tax}) \times \text{discount}$
  - $\text{cost} + (\text{tax} \times \text{discount})$

- Operator Precedence Order
  1. (unary) $+, -, ++, --$
  2. (unary) $(\text{type})$
  3. (binary) $\times, /, \%$
  4. (binary) $+, -$

- Complete list on the back cover of the textbook

Different Results
Testing & Debugging

- Sometimes designing and implementing that design is not enough
  - Test the program to verify it behaves correctly
  - Debug the program to resolve any problems
Why “Debugging?”

Photo # NH 96566-KN  First Computer "Bug", 1945

Image Source: http://sections.asme.org/jerseyshore/bug.html
Types of Bugs

- Syntax Error: A violation of the programming language’s grammar
  - In this case, the program will not compile or run

- Examples:
  - `int a = 5,
  - `int b = 0.5;
  - `c+ +
Types of Bugs

- Run-Time Error: an error that is detected during program execution
  - The program will compile, run to the point of failure, and then crash

Example:

```java
int n1 = 5;
int n2 = 0;
int n3 = 2 + n1 / n2;
```
Types of Bugs

- Logic Error: a mistake in the program caused by the underlying algorithm
  - The program will compile and run, but behave unexpectedly

Examples:

- double unitCost = 2.50;
  int quantity = 5;
  double totalCost = unitCost / quantity;

- double a = 1.0, b = 2.0, c = 3.0;
  double x1 =
    (-b + Math.sqrt(b * b - 4 * a * c)) / 2.0
Questions?
Logistics

- Homework 0 is due tonight by 11:55p
  - Talk to me if you still don’t have Sakai access

- Program 1 is now assigned
  - Due next Friday (June 29) at 9:45a

- Next: Installing Java & Eclipse (Lab 0)

- Monday: Strings, Console I/O, & Lab 1