COMP 110

Primitive Types, Strings, and Console I/O

Darrell Bethea
May 12, 2011
Announcements

- Homework 0 due tonight
- Grades will be posted on Blackboard
  - If you do not have a grade and thought you submitted the hw, please email me.
- Program 1 due in 6 days
  - Follow submission instructions
  - .jar tutorial
Questions?
Today in COMP 110

- 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!!!
A variable corresponds to a location in memory

- Use this memory cell to store the value of n1
- 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 Declarations

- **Syntax:**
  - `Type Variable_1, Variable_2, ...;

- **Examples:**
  - `int count, score, myInt;
  - `char letter;
  - `double totalCost, ratio;
How to name an identifier

- Letters, digits(0–9), underscore (_)
- First character cannot be a digit
- No spaces or other characters
- Java is case sensitive

- Legal names
  ◦ pinkFloyd, the_coup, b3atles

- Illegal names
  ◦ michael.bolton, kenny–G, 1CP
Keywords

- Reserved words with predefined meanings
- You cannot name your variables keywords
- Appendix 1
- **if, else, return, new, ...**
  - See inside front book cover for full list
What kind of value the variable can hold

- **Primitive type** – indecomposable values
  - Names begin with lowercase letters
  - int, double, char, boolean
  - See inside front book cover for full list

- **Class type** – objects with both data and methods
  - Name begins with uppercase letter
  - Scanner, String
Integers (\texttt{byte, short, int, long})
- Some possible values: 0, -3, 5, 43

Floating-point numbers (\texttt{float, double})
- Some possible values: 0.5, 12.4863, -4.3

Characters (\texttt{char})
- Some possible values: A, r, %, T

Booleans (\texttt{boolean})
- Only possible values: true, false
Primitive Types: small to big

double
float
long
int
short
byte
When declaring a variable, a certain amount of memory is assigned based on the declared type.

- `int age;`
- `double length;`
- `char letter;`

Main memory
Variables Change

- `int changingVar = 0;`
- `changingVar = 5;`
- `changingVar = changingVar + 4;`
Assignment Statements

- Change a variable’s value

  Syntax:
  - variable = expression;

  Example:
  - sleepNeeded = 8;
  - sleepDesired = sleepNeeded * 2;
variable = expression;
- CPU calculates the value of the expression.
- Stores the value in the memory location used by the variable.

sleepDesired = sleepNeeded * 2;
- Calculate sleepNeeded * 2
  - Get the current value of sleepNeeded from its memory location
  - Multiply it by 2
- Assign the value to the location of sleepDesired
- `total += 5; // is the same as total = total + 5;`
- `count++; // is the same as count = count + 1;`
Assignment compatibilities

- Usually, we need to put values of a certain type into variables of the same type.
- However, in some cases, the value will automatically be converted when types are different.

- `int` `age;`
- `age = 10;`
- `double` `length;`
- `length = age;`
Assignment Compatibilities

- You can only put small things into bigger things

- `byte -> short -> int -> long -> float -> double`

  - `myShort != myInt;`
  - `myByte != myLong;`
  - `myFloat = mybyte;`
  - `myLong = myInt;`
Type Casting

- You can ask Java to change the type of values which would violate the compatibility rule.

- `myFloat = myDouble;`
- `myByte = myInt;`
- `myShort = myFloat;`

- `myFloat = (float)myDouble;`
- `myByte = (byte)myInt;`
- `myShort = (short)myFloat;`
Arithmetic Operators

- **Unary operators (more info later)**
  - +, -, ++, --, !

- **Binary arithmetic operators**
  - *, /, %, +, -
    - rate*rate + delta
    - 1/(time + 3*mass)
    - (a – 7)/(t + 9*v)
Modular Arithmetic – %

- “clock arithmetic”
  - Minutes on a clock are mod 60
- 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)
Parentheses and Precedence

- Expressions inside parentheses evaluated first
  - (cost + tax) * discount
  - cost + (tax * discount)

- Highest precedence
  - First: the unary operators:
    - +, -, ++, --, !
  - Second: the binary arithmetic operators:
    - *, /, %
  - Third: the binary arithmetic operators:
    - +, -
Parentheses and Precedence

\[ \text{total} = \text{cost} + \text{tax} \times \text{discount}; \]

Same as:

\[ \text{total} = \text{cost} + (\text{tax} \times \text{discount}); \]

Probably we wanted:

\[ \text{total} = (\text{cost} + \text{tax}) \times \text{discount}; \]

Full operator precedence table on back cover
Errors

- Syntax error – grammatical mistake in your program
  - `int n3 = n1 + n2, // Need a ‘;’, not a ‘,’`

- 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.`
String

String month = “May”;

System.out.println(month);

Prints:

May
String Concatenation

- String month = “May”;
- String sentence;
- Sentence = “This month is ” + month;

This month is May
String (Class type)

Class types have methods

Object

String `myString` = “COMP110”;

Method

```
int len = myString.length();
```
Strings Methods (pp. 80–82)

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

- You will see these in the lab tomorrow
String output = myString.substring(1, 8);
System.out.println(“How do I put \“quotes\” in my string?”);
System.out.println("How do I put a \\ in my string?");
## Escape Characters

<table>
<thead>
<tr>
<th>Escape</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>\&quot;</code></td>
<td>Double quote</td>
</tr>
<tr>
<td><code>'</code></td>
<td>Single quote</td>
</tr>
<tr>
<td><code>\</code></td>
<td>Backslash</td>
</tr>
<tr>
<td><code>\n</code></td>
<td>New line</td>
</tr>
<tr>
<td><code>\r</code></td>
<td>Carriage return</td>
</tr>
<tr>
<td><code>\t</code></td>
<td>Tab</td>
</tr>
</tbody>
</table>
I/O (Input/Output)

- System.out.print("this is a string");
- System.out.println("this is a string");

- What is the difference?
Keyboard Input

- Scanner `Scanner_object_name` = new Scanner(System.in);
- `Scanner_object_name`.nextLine();
- `Scanner_object_name`.nextInt();
- `Scanner_object_name`.nextDouble();
- See p. 86
- Make sure to read Gotcha on p. 89
Documentation and Style

- Meaningful names
- Indenting
- Documentation (comments)
- Defined Constants
public static final Type Variable = Constant;
Named in ALL_CAPS

public class NamedConstant
{
    public static final double PI = 3.14159;
    public static void main(String[] args)
    {
        ...
    }
}
Tomorrow

- Console I/O
- Read Sections 2.3–2.5
- Lab 2