ComS 207: Programming I

Instructor: Alexander Stoytchev
http://www.cs.iastate.edu/~alex/classes/2007_Fall_207/

Variables and Assignment
August 24, 2007

Quick review of last 2 lectures

Our First Program

```java
// comments about the class
public class MyProgram
{
   // comments about the method
   public static void main (String[] args)
   {
      System.out.println("Hello World");
   }
}
```

The same program in the C language

```c
// comments about my first program
#include <stdio.h>
int main()
{
   printf("Hello World!\n");
   return 0;
}
```

Program Development

- The mechanics of developing a program include several activities
  - writing the program in a specific programming language (such as Java)
  - translating the program into a form that the computer can execute
  - investigating and fixing various types of errors that can occur
- Software tools can be used to help with all parts of this process
Programming Languages

- Each type of CPU executes only a particular machine language
- A program must be translated into machine language before it can be executed
- A compiler is a software tool which translates source code into a specific target language
- Often, that target language is the machine language for a particular CPU type
- The Java approach is somewhat different

Java Translation

- The Java compiler translates Java source code into a special representation called bytecode
- Java bytecode is not the machine language for any traditional CPU
- Another software tool, called an interpreter, translates bytecode into machine language and executes it
- Therefore the Java compiler is not tied to any particular machine
- Java is considered to be architecture-neutral

Java Translation

Java source code → Java bytecode
Java compiler
Bytecode interpreter
Bytecode compiler
Machine code

Syntax and Semantics

- The syntax rules of a language define how we can put together symbols, reserved words, and identifiers to make a valid program
- The semantics of a program statement define what that statement means (its purpose or role in a program)
- A program that is syntactically correct is not necessarily logically (semantically) correct
- A program will always do what we tell it to do, not what we meant to tell it to do

Basic Program Development

- Errors
  - A program can have three types of errors
  - The compiler will find syntax errors and other basic problems (compile-time errors)
    - If compile-time errors exist, an executable version of the program is not created
  - A problem can occur during program execution, such as trying to divide by zero, which causes a program to terminate abnormally (run-time errors)
  - A program may run, but produce incorrect results, perhaps using an incorrect formula (logical errors)

Errors
Character Strings

A string of characters can be represented as a **string literal** by putting double quotes around the text:

- Examples:
  - "This is a string literal."
  - "123 Main Street"
  - "X"

Every character string is an object in Java, defined by the **String** class

Every string literal represents a **String** object

---

The print Method

- The **System.out** object provides another service as well

- The **print** method is similar to the **println** method, except that it does not advance to the next line

- Therefore anything printed after a **print** statement will appear on the same line

- See **Countdown.java** (page 63)

---

String Concatenation

- The **string concatenation operator (+)** is used to append one string to the end of another
  - "Peanut butter " + " and jelly"

- It can also be used to append a number to a string

- A string literal cannot be broken across two lines in a program

- See **Facts.java** (page 65)
### String Concatenation

- The `+` operator is also used for arithmetic addition.
- The function that it performs depends on the type of the information on which it operates.
- If both operands are strings, or if one is a string and one is a number, it performs string concatenation.
- If both operands are numeric, it adds them.
- The `+` operator is evaluated left to right, but parentheses can be used to force the order.

See *Addition.java* (page 67)

### Escape Sequences

- What if we wanted to print a the quote character?
- The following line would confuse the compiler because it would interpret the second quote as the end of the string:
  ```java
  System.out.println("I said "Hello" to you.");
  ```
- An escape sequence is a series of characters that represents a special character.
- An escape sequence begins with a backslash character (`\`).

```java
System.out.println("I said \\
Hello\\" to you.");
```

### Variables

- A variable is a name for a location in memory.
- A variable must be declared by specifying the variable's name and the type of information that it will hold.

```java
int total;
int count, temp, result;
```

### Rules for valid variable names

- The name can be made up of letters, digits, the underscore character (`_`), and the dollar sign.
- Variable names cannot begin with a digit.
- Java is case sensitive - `Total`, `total`, and `TOTAL` are different identifiers.
- By convention, programmers use different case styles for different types of names/identifiers, such as:
  - **title case** for class names - `Lincoln`
  - **upper case for constants** - `MAXIMUM`

### Variable Initialization

- A variable can be given an initial value in the declaration:
  ```java
  int sum = 0;
  int base = 32, max = 149;
  ```
- When a variable is referenced in a program, its current value is used.

See *PianoKeys.java* (page 70)
Assignment

- An assignment statement changes the value of a variable.
- The assignment operator is the = sign.
  
  \[
  \text{total} = 55; \\
  \]
  
  - The expression on the right is evaluated and the result is stored in the variable on the left.
  - The value that was in `total` is overwritten.
  - You can only assign a value to a variable that is consistent with the variable's declared type.
  - See `Geometry.java` (page 71).

Constants

- A constant is an identifier that is similar to a variable except that it holds the same value during its entire existence.
- As the name implies, it is constant, not variable.
- The compiler will issue an error if you try to change the value of a constant.
- In Java, we use the `final` modifier to declare a constant.
  
  ```java
  final int MIN_HEIGHT = 69;
  ```

- Constants are useful for three important reasons:
  
  1. First, they give meaning to otherwise unclear literal values.
     - For example, `MAX_LOAD` means more than the literal 250.
  2. Second, they facilitate program maintenance.
     - If a constant is used in multiple places, its value need only be updated in one place.
  3. Third, they formally establish that a value should not change, avoiding inadvertent errors by other programmers.

Run examples from the book

THE END