

ComS 207: Programming I

Instructor: Alexander Stoytchev

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

© 2004 Pearson Addison-Wesley. All rights reserved

Variables and Assignment

August 24, 2007

ComS 207: Programming I (in Java)
Iowa State University, FALL 2007
Instructor: Alexander Stoytchev

© 2004 Pearson Addison-Wesley. All rights reserved

Quick review of last 2 lectures

© 2004 Pearson Addison-Wesley. All rights reserved

Our First Program

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

© 2004 Pearson Addison-Wesley. All rights reserved

The same program in the C language

```
// comments about my first program
#include <stdio.h>

int main()
{
    printf("Hello World!\n");
    return 0;
}
```

© 2004 Pearson Addison-Wesley. All rights reserved

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

© 2004 Pearson Addison-Wesley. All rights reserved

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

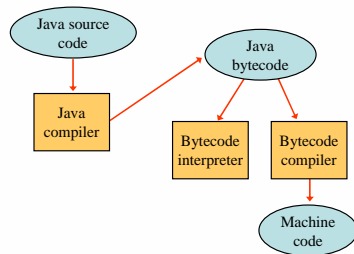
© 2004 Pearson Addison-Wesley. All rights reserved.

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*

© 2004 Pearson Addison-Wesley. All rights reserved.

Java Translation



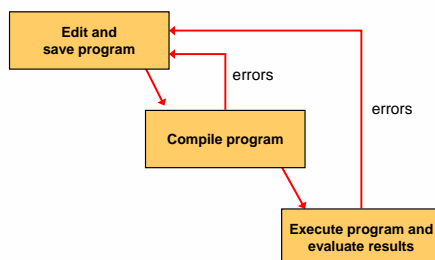
© 2004 Pearson Addison-Wesley. All rights reserved.

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

© 2004 Pearson Addison-Wesley. All rights reserved.

Basic Program Development



© 2004 Pearson Addison-Wesley. All rights reserved.

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*)

© 2004 Pearson Addison-Wesley. All rights reserved.

The first bug!

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

1945
9/9
0800 Action started
1000 stopped - action
1500 1000 MP one
2000 800 = 2 13000000
2100 800 = 2 13000000
2200 800 = 2 13000000
2300 800 = 2 13000000
2400 800 = 2 13000000
2500 800 = 2 13000000
2600 800 = 2 13000000
2700 800 = 2 13000000
2800 800 = 2 13000000
2900 800 = 2 13000000
3000 800 = 2 13000000
3100 800 = 2 13000000
3200 800 = 2 13000000
3300 800 = 2 13000000
3400 800 = 2 13000000
3500 800 = 2 13000000
3600 800 = 2 13000000
3700 800 = 2 13000000
3800 800 = 2 13000000
3900 800 = 2 13000000
4000 800 = 2 13000000
4100 800 = 2 13000000
4200 800 = 2 13000000
4300 800 = 2 13000000
4400 800 = 2 13000000
4500 800 = 2 13000000
4600 800 = 2 13000000
4700 800 = 2 13000000
4800 800 = 2 13000000
4900 800 = 2 13000000
5000 800 = 2 13000000
5100 800 = 2 13000000
5200 800 = 2 13000000
5300 800 = 2 13000000
5400 800 = 2 13000000
5500 800 = 2 13000000
5600 800 = 2 13000000
5700 800 = 2 13000000
5800 800 = 2 13000000
5900 800 = 2 13000000
6000 800 = 2 13000000
6100 800 = 2 13000000
6200 800 = 2 13000000
6300 800 = 2 13000000
6400 800 = 2 13000000
6500 800 = 2 13000000
6600 800 = 2 13000000
6700 800 = 2 13000000
6800 800 = 2 13000000
6900 800 = 2 13000000
7000 800 = 2 13000000
7100 800 = 2 13000000
7200 800 = 2 13000000
7300 800 = 2 13000000
7400 800 = 2 13000000
7500 800 = 2 13000000
7600 800 = 2 13000000
7700 800 = 2 13000000
7800 800 = 2 13000000
7900 800 = 2 13000000
8000 800 = 2 13000000
8100 800 = 2 13000000
8200 800 = 2 13000000
8300 800 = 2 13000000
8400 800 = 2 13000000
8500 800 = 2 13000000
8600 800 = 2 13000000
8700 800 = 2 13000000
8800 800 = 2 13000000
8900 800 = 2 13000000
9000 800 = 2 13000000
9100 800 = 2 13000000
9200 800 = 2 13000000
9300 800 = 2 13000000
9400 800 = 2 13000000
9500 800 = 2 13000000
9600 800 = 2 13000000
9700 800 = 2 13000000
9800 800 = 2 13000000
9900 800 = 2 13000000
10000 800 = 2 13000000

11/11 Started Sine tape (Sine check)
15/25 Started Multiplier Test
15/45 Relay #70 Panel F (moth) in relay.

First actual case of bug being found.
changed stack.
changed dem.

Read more at:
<http://www.history.navy.mil/photos/pers-us/uspers-h/g-hoppr.htm>

© 2004 F

Chapter 2

Data and Expressions

PEARSON Addison-Wesley

© 2004 Pearson Addison-Wesley. All rights reserved.

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

© 2004 Pearson Addison-Wesley. All rights reserved.

The println Method

- In the `Lincoln` program from Chapter 1, we invoked the `println` method to print a character string
- The `System.out` object represents a destination (the monitor screen) to which we can send output

```
System.out.println("Whatever you are, be a good one.");
```

object method name information provided to the method (parameters)

© 2004 Pearson Addison-Wesley. All rights reserved.

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)

© 2004 Pearson Addison-Wesley. All rights reserved.

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)

© 2004 Pearson Addison-Wesley. All rights reserved.

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)

© 2004 Pearson Addison-Wesley. All rights reserved.

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

```
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 (\)

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

© 2004 Pearson Addison-Wesley. All rights reserved.

Escape Sequences

- Some Java escape sequences:

Escape Sequence	Meaning
<code>\b</code>	backspace
<code>\t</code>	tab
<code>\n</code>	newline
<code>\r</code>	carriage return
<code>\"</code>	double quote
<code>'</code>	single quote
<code>\\</code>	backslash

- See [Roses.java](#) (page 68)

© 2004 Pearson Addison-Wesley. All rights reserved.

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

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

Multiple variables can be created in one declaration

© 2004 Pearson Addison-Wesley. All rights reserved.

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

© 2004 Pearson Addison-Wesley. All rights reserved.

Variable Initialization

- A variable can be given an initial value in the declaration

```
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)

© 2004 Pearson Addison-Wesley. All rights reserved.

Assignment

- An *assignment statement* changes the value of a variable
- The assignment operator is the = sign

```
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)

© 2004 Pearson Addison-Wesley. All rights reserved

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

```
final int MIN_HEIGHT = 69;
```

© 2004 Pearson Addison-Wesley. All rights reserved

Constants

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

© 2004 Pearson Addison-Wesley. All rights reserved

Run examples from the book

© 2004 Pearson Addison-Wesley. All rights reserved

THE END

© 2004 Pearson Addison-Wesley. All rights reserved