White Space

- Whitespace, Errors, Variables, Data Types, Assignment
- Lecture 4, Wed Jan 13, 2010
- borrowing from slides by Kurt Eiselt

Review: Comments

- Comments: help humans understand
- Ignored by compiler
- Comment out rest of line: //
- Comment start/end: /*   */

Review: High-Level Language

- Must be translated into machine language so the computer can understand.
- High-level instruction: \( A = B + C \)
- Comes at least four machine language instructions!

Review: Identifiers

- Identifier must
  - Start with a letter and be followed by
  - Digits are 0 through 9.
  - Letters are the 26 characters in English alphabet
  - But you don't need to memorize all 52 for exam
  - Digits are included
  - No special characters are included
  - Also special characters from other languages
  - Which of the following are not valid identifiers?

Review: Reserved Words

- Get familiar with these
- But you don't need to memorize all 52 for exam

Review: Java Does Itself

- Your Program.java
  - javac
  - Your Program.class
  - JVM interpreter (Win)

White Space

- Whitespace: breaks between identifiers, operators, and statements
- Tabs and newline characters included
- Does not affect how program runs
- Use whitespace to format programs for easier understanding

Review: Identifiers

- Java is case sensitive
- Oreo oreo OREO oreo
- are all different identifiers, so be careful
- Common source of errors in programming
- are these all valid identifiers?

Identifiers

- Creating identifiers in your Java programs
- Remember other people read what you create
- Make identifiers meaningful and descriptive for both you and them
- No limit to how many characters you can put in your identifiers
- but don't get carried away

White Space

- Java whitespace is ignored by the computer
- But helps humans understand
- Use whitespace to format programs for easier understanding

Review: Reserved Words

- Get familiar with these
- But you don't need to memorize all 52 for exam

Review: High-Level Language

- Must be translated into machine language so the computer can understand.
- High-level instruction: \( A = B + C \)
- Comes at least four machine language instructions!

Review: Identifiers

- Identifier must
  - Start with a letter and be followed by
  - Digits are 0 through 9.
  - Letters are the 26 characters in English alphabet
  - But you don't need to memorize all 52 for exam
  - Digits are included
  - No special characters are included
  - Also special characters from other languages
  - Which of the following are not valid identifiers?

Review: Comments

- Comments: help humans understand
- Ignored by compiler
- Comment out rest of line: //
- Comment start/end: /*   */

Review: High-Level Language

- Must be translated into machine language so the computer can understand.
- High-level instruction: \( A = B + C \)
- Comes at least four machine language instructions!

Review: Identifiers

- Identifier must
  - Start with a letter and be followed by
  - Digits are 0 through 9.
  - Letters are the 26 characters in English alphabet
  - But you don't need to memorize all 52 for exam
  - Digits are included
  - No special characters are included
  - Also special characters from other languages
  - Which of the following are not valid identifiers?

Review: Reserved Words

- Get familiar with these
- But you don't need to memorize all 52 for exam

Review: High-Level Language

- Must be translated into machine language so the computer can understand.
- High-level instruction: \( A = B + C \)
- Comes at least four machine language instructions!

Review: Identifiers

- Identifier must
  - Start with a letter and be followed by
  - Digits are 0 through 9.
  - Letters are the 26 characters in English alphabet
  - But you don't need to memorize all 52 for exam
  - Digits are included
  - No special characters are included
  - Also special characters from other languages
  - Which of the following are not valid identifiers?

Review: Comments

- Comments: help humans understand
- Ignored by compiler
- Comment out rest of line: //
- Comment start/end: /*   */

Review: High-Level Language

- Must be translated into machine language so the computer can understand.
- High-level instruction: \( A = B + C \)
- Comes at least four machine language instructions!

Review: Identifiers

- Identifier must
  - Start with a letter and be followed by
  - Digits are 0 through 9.
  - Letters are the 26 characters in English alphabet
  - But you don't need to memorize all 52 for exam
  - Digits are included
  - No special characters are included
  - Also special characters from other languages
  - Which of the following are not valid identifiers?

Review: Reserved Words

- Get familiar with these
- But you don't need to memorize all 52 for exam

Review: High-Level Language

- Must be translated into machine language so the computer can understand.
- High-level instruction: \( A = B + C \)
- Comes at least four machine language instructions!

Review: Identifiers

- Identifier must
  - Start with a letter and be followed by
  - Digits are 0 through 9.
  - Letters are the 26 characters in English alphabet
  - But you don't need to memorize all 52 for exam
  - Digits are included
  - No special characters are included
  - Also special characters from other languages
  - Which of the following are not valid identifiers?
Program Development

- Use an editor to create your Java program
- Often called source code
- Code used interchangeably with program or instructions in the computer world
- Another program, a compiler or an interpreter, translates source code into target language or object code, which is often machine language
- Finally, your computer can execute object code

Compiling and Running

- Let's try it!
- Command line for now
- Later we'll use Eclipse
- Integrated development environment (IDE)

Compiling and Running Java

- What I did at the command line
  - Create file HelloWorld.java in text editor
  - Containing class HelloWorld
- Compile it: "javac HelloWorld.java"
- Compiler makes file HelloWorld.class
- Run it in the interpreter: "java HelloWorld"
- Don't panic if this is mysterious!
- Hands-on practice in labs this week
- See detailed instructions on WebCT for how to download and configure your home desktop/laptop
- If you get stuck, bring laptop to lab or BLC for help
- A few weeks from now: Eclipse IDE

Syntax

- Rules to dictate how statements are constructed.
  - Example: open bracket needs matching close bracket
- If program is not syntactically correct, cannot be translated by compiler
- Different than humans dealing with natural languages like English. Consider statement with incorrect syntax (grammar)
- For weeks, rained in Vancouver it hasn't
- We still have pretty good shot at figuring out meaning

Semantics

- What will happen when statement is executed
- Programming languages have well-defined semantics, no ambiguity
- Different than natural languages like English. Consider statement:
  - Mary counted on her computer.
  - How could we interpret this?
- Programming languages cannot allow for such ambiguities or computer would not know which interpretation to execute

Errors

- Computers follow our instructions exactly
- If program produces the wrong result it's the programmer's fault
- Unless the user inputs incorrect data
- Then cannot expect program to output correct results: "Garbage in, garbage out" (GIGO)
- Debugging: process of finding and correcting errors
- Unfortunately can be very time consuming!

Errors

- Error at compile time (during translation)
- You did not follow syntax rules that say how Java elements must be combined to form valid Java statements

Errors

- Error at run time (during execution)
- Source code compiles
  - Syntactically (structurally) correct
  - But program tried something computers cannot do
  - Like divide a number by zero.
  - Typically program will crash: halt prematurely

Errors

- Logical error
  - Source code compiles
  - Object code runs
  - But program may still produce incorrect results because logic of your program is incorrect
  - Typically hardest problems to find

Errors

- Let's try it!
  - Usually errors happen by mistake, not on purpose...

Memory Recap

- Memory: series of locations, each having a unique address, used to store programs and data
- When data is stored in a memory location, previously stored data is overwritten and destroyed
- Each memory location stores one byte (8 bits) of data

Memory and Identifiers

- Example of a high-level instruction
  - \( A \equiv B + C \)
- Tells computer to
  - Go to main memory and find value stored in location called \( B \)
  - Go to main memory and find value stored in location called \( C \)
  - Add those two values together
  - Store result in memory in location called \( A \)
- Great! But... in reality, locations in memory are not actually called things like \( a, b, \) and \( c. \)

Memory and Identifiers

- So what's with the \( a, b, \) and \( c? \)
  - Machine language uses actual addresses for memory locations
  - High-level languages easier
  - Avoid having to remember actual addresses
  - Invent meaningful identifiers giving names to memory locations where important information is stored
  - \( pay \textunderscore \text{rate} \) and \( \text{hours} \textunderscore \text{worked} \) vs. 5802 and 5806
  - Easier to remember and a whole lot less confusing!

Memory and Identifiers: Variables

- Variable: name for location in memory where data is stored
  - Like variables in algebraic class
- \( \text{pay}\textunderscore \text{rate} \), \( \text{hours} \textunderscore \text{worked} \), \( a, b, \) and \( c \) are all variables
- Variable names begin with lower case letters
- Java convention, not compiler/syntax requirement
- \( \text{pay}\textunderscore \text{rate} \) and \( \text{hours} \textunderscore \text{worked} \) vs. 5802 and 5806
- Easier to remember and a whole lot less confusing!

Programming With Variables

```java
public class HelloWorld {
    public static void main (String[] args) {
        a = b + c;
        System.out.println("The answer is " + a);
    }
}
```

Programming With Variables

```java
public class HelloWorld {
    public static void main (String[] args) {
        a = b + c;
        System.out.println("The answer is " + a);
    }
}
```

Programming With Variables

```java
public class HelloWorld {
    public static void main (String[] args) {
        a = b + c;
        System.out.println("The answer is " + a);
    }
}
```
### Programming With Variables: Take 2

```java
// Test2.java  Author: Kurt
// Our second use of variables!
public class Test2
{
    public static void main (String[] args)
    {
        b = 3;
        c = 5;
        a = b + c;
        System.out.println("The answer is " + a);
    }
}
```

### Now What?

Java doesn't know how to interpret the contents of the memory location.

- Are they integers?
- Characters from the keyboard?
- Shades of gray? or....

Now what?

- Such a lazy computer, still can’t find symbols...

### Programming With Variables: Take 3

```java
// Test3.java  Author: Kurt
// Our third use of variables!
public class Test3
{
    public static void main (String[] args)
    {
        b = 3;
        c = 5;
        a = b + c;
        System.out.println("The answer is " + a);
    }
}
```

### Floating Point Numbers

**significant digits**
- 42 = $4.2 \times 10$
  - $42 = 4.2 \times 10^1$
- 4.2 = $4.2 \times 1$
  - $4.2 = 4.2 \times 10^0$
- 4200000 = $4.2 \times 1000000$
  - $4200000 = 4.2 \times 10^7$
- .000042 = $4.2 \times .00001$
  - $4.2 \times 10^{-5}$

**fixed size, finite capacity**

- nonzero digits
- where to put the decimal point
- floats around when multiply/divide by 10

### Assignment Statements

- Java first computes value on right side
- Then assigns value to variable given on left side
- x = 4 + 7; // what's in x?
- x = 2 + 1; // what's in x now?

Old value will be overwritten if variable was assigned before.

- Note that = is no longer a test for equality!
Assignment Statements

■ Here’s an occasional point of confusion:

```java
a = 7;       // what’s in a?
b = a;        // what’s in b?
System.out.println("a is " + a + " b is " + b);
```

■ Find out! Experiments are easy to do in CS

```java
a = 7;       // what’s in a?
b = a;        // what’s in b?
System.out.println("a is " + a + " b is " + b);
```

Variable values on left of = are clobbered
Variable values on right of = are unchanged
Copy of value assigned to a also assigned to b
but that doesn’t change value assigned to a

Memory locations a and b are distinct
Copy of value assigned to a also assigned to b
Changing a later does not affect previous copy
More later

Assignment Statements

■ Here’s an occasional point of confusion:

```java
a = 7;       // what’s in a?
b = a;        // what’s in b?
System.out.println("a is " + a + " b is " + b);
```

```java
System.out.println("a is " + a + " b is " + b);
```

```java
a = 8;
System.out.println("a is " + a + " b is " + b);
```

Variable Declaration and Assignment

■ Variable declaration is instruction to compiler
  ■ Reserve block of main memory large enough to store data type specified in declaration
  ■ Variable name is specified by identifier
  ■ Syntax:
    ■ `typeName variableName;`
    ■ `typeName variableName = value;`
    ■ Can declare and assign in one step