Today’s Topics

• Questions / Comments?
• Questionnaire
• Continue first program discussion
• Escape sequences
• String concatenation
• Variables & Types
1 credit honors add-on course

• do not have to be an honors forum student to enroll
• part of Project Extinction. Our “cluster” is called Obsolescence.
• work with students from other disciplines on the topic of "Obsolescence."
• the work you do over the course of the semester will be largely student driven, with a final collaborative project at the end.
• HF-200.003.Fall2015: Obsolescence - Instructors: Michael Eckmann; Evan Halstead; Catherine Hill.
• meet from 12:40-2pm on Thursdays. Students can add the class any time before the add deadline.
Hello, World! program

/* CS106 first program
   written by Michael Eckmann
   purpose: print the text “Hello, World!”
   to the screen */
public class HelloWorld
{
    public static void main(String args[])
    {
        System.out.println("Hello, World!");
    }  // end of main method

}  // end of HelloWorld class
Discussion of “Hello, World!”

• { and } – are called braces, I call them curly braces.
• ( and ) – are called parentheses, I sometimes call them parens.
• [ and ] – are called brackets, I call them square brackets.
Some common programming mistakes (bugs)

• Case matters --- upper case letters in class names etc. are different than lower case letters --- e.g. String and string are understood differently by the compiler. Additionally the capitalized String is necessary in the main method argument list declaration.

• System must be capitalized when referring to something like: System.out because System is the name of the class/object, system with a lower case s is not.

• Spaces are not allowed in a class name or any identifier / variable name e.g. HelloWorld is legal, Hello World is not.
Some common programming mistakes (bugs) (continued)

- Double quotes must be paired up when giving the value of a String --- can’t forget the second double quote.
- Curly braces must be paired correctly. An extra one dangling somewhere in the code will cause errors, same if one is forgotten.
- Class name must be the same as the main part of the file name. e.g. a public class named HelloWorld must reside in a file named HelloWorld.java --- not in something like HW.java or any other file.
- Java source code can only be in plain text files. If a Java source code file is stored as a Word document or any other file type besides plain text, the code will not be able to be compiled.

Michael Eckmann  -  Skidmore College  -  CS 106 -  Fall 2015
println vs. print

System.out.println("I love java.");
System.out.println("It tastes good.");

If these two statements were inside the main method's curly braces, then the program would print:

I love java.
It tastes good.
println vs. print

System.out.print("I love java.");
System.out.println("It tastes good.");

Will print:

I love java. It tastes good.
println vs. print

System.out.print("I love java.");
System.out.print("It tastes good.");

How will this print?
Phases of Java Programs

• create / compile the program
  – Create / Edit the source code of the program
  – Compile / Build the source code into bytecodes
  – After the .java file is compiled we get a .class file
    • E.g. if we compile a program called HelloWorld.java the result is HelloWorld.class

• When we execute the program (via the java interpreter), the interpreter does the following:
  – Loads the Classes
  – Verifies the Bytecodes
  – Interprets the Bytecodes

• All of these “execution” steps are done on the .class file(s)
Phases of Java Programs

- terminology you need to understand
  - source code
  - Compile / build
  - bytecode
  - Execute / run
Phases of Java Programs

- terminology you need to understand
  - source code (= the .java files)
  - Compile / build (the source code into bytecode)
  - bytecode (= the .class files)
  - Execute / run (the .class file(s) using the Java bytecode interpreter)
Eclipse

Eclipse is the integrated development environment that we will use for this course.

You are free to use others but I will give instructions for Eclipse and show examples during labs, etc. in Eclipse.
Escape sequences

• A String's value is specified between double quotes.
• Within the double quotes the backslash character \ is special.
• The backslash character followed by another character is called an escape sequence.
• Escape sequences are handled differently than regular characters.
• \n is a common escape sequence to embed a newline
• \t is a way to add a tab to a string
• E.g. System.out.println(“Hello\nWorld”); prints: Hello World
Escape sequences

• \b is the backspace character
• There are only a total of eight valid escape sequences.
• Let’s say you wanted to print a sarcastic message using double quotes.
• If we do
  System.out.println("Hamilton has a "great" lacrosse team");
• there will be a syntax error at compile time.
Escape sequences

System.out.println("Hamilton has a "great" lacrosse team");

• The string will be
  – Hamilton has a
• and then the compiler will see a g and will generate an error because the second double quote (the one before great) causes the string to end.
• What do we do?
Escape sequences

• There is an escape sequence to represent a ". It is \\
System.out.println("Hamilton has a "great" lacrosse team");

• Now, the string literal contains the double quotes around great. The first double quote and last double quote mark the beginning and ending of the String literal.

• The above line of code will do what we want now.
Escape sequences

• The \ character is special inside a String. Java will interpret the \ and the next character as an escape sequence (even if it is not one of the valid ones.)
• How might we add a backslash character to a String?
• Let’s say we want a String like:
  – C:\My Documents
• Will this work?
  – System.out.println("C:\My Documents");
Let’s try this problem

• Write an application that displays the numbers 1 to 4 on the same line, with each pair of adjacent numbers separated by one space. Write the program using the following methods:
  – A) Using one System.out statement
  – B) Using four System.out statements

• Now let's do the same thing but on 4 different lines.
  – A) Using one System.out statement
  – B) Using four System.out statements
String concatenation

• To concatenate two Strings together, Java provides the operator +

  e.g.
  System.out.println("Hey " + "now.");
  prints
  Hey now.
String concatenation

- Technically, Strings can only be concatenated to other Strings, but numbers are converted to Strings by Java automatically if they are being concatenated to a String.

e.g.
System.out.println("I am " + 21 + " years old.");
Will work the same as:
System.out.println("I am " + "21" + " years old.");
Will work the same as:
System.out.println("I am 21 years old.");
Variables and Type

• A program often needs to be able to store and change data as it is running. For example, a program might need to store a user's name, or the user's age, or the air temperature, or whatever.

• Variables allow names to be given to data that can be stored and changed.
Variables and Type

• A variable is a location in memory (RAM) with the following properties.

• name --- given by the programmer --- try to be descriptive of the data the variable will hold

• type --- decided by the programmer from all available types, depending on what kind of values the programmer wants to store

• size --- amount of memory used, this is determined by the type

• value --- self-explanatory
  – value can be stored / changed / read by a program.
Primitive Types for Variables

- **byte, short, int, long** --- these are integer types ---
  they hold *whole* numbers like $-31$, $4256$, $56$, $2632$, $755$, $-901$ (they hold different ranges)

- **char** --- this is a type that holds *one* character like a letter, digit, *, &, or an escape sequence …

- **boolean** --- holds one of *true* or *false*

- **float, double** --- numeric types that hold numbers with decimal values like $-5.5$, $98.6$, $1002.99995$ etc. (they hold different ranges and to different precisions)

- We can look up the ranges/precision of the numeric types.
To be able to specify to our program that we want to use a variable, **before we can use it we must declare the variable.** That is, we state the desired type as well as a descriptive name that we make up (must start with a letter and may contain letters, digits, and _).

Example: If we wanted a place to store a person's age (years only e.g. 18, 21, 55 etc.), we might create a variable like this:

- **int age;** // declares the variable age of type int

- age is a name we made up, what does int do?
Variables and Type

- `int age;` // declares the variable age of type int

- age is a name we made up, what does int do?
- We chose int because we knew we wanted to only store whole numbers and the int type stores whole numbers. We could have chosen byte, short or long which also hold whole numbers but we typically use int for whole numbers unless we have a good reason to use one of those other three.
Variables and Type

```java
int area = 50; // This line of code not only
// declares a variable but also initializes its value.
```

- **Name of variable** is `area`
- **Type of variable** is `int`
- **Size of variable** is 32 bits (because an int has a size of 32 bits.)
- **Value of variable** is currently **50**

Note the same thing can be done with 2 lines of code:

```java
int area;
area = 50; // this is an assignment statement (it assigns
// the value 50 to the variable area
```
Program Flow

- Java programs start executing in the main method.
- They execute sequentially, in order (unless something changes the program flow --- more on that later today and throughout the semester) until the end of the main method when the program ends.
public class VariableExample
{
    public static void main(String args[])
    {
        int result; // declares variable with name result and type int
        // allocates enough memory for an int to be stored
        int some_number; // declares variable with name some_number and type
        // int and allocates enough memory for an int to be stored
        result = 35; // an assignment statement that stores the value 35 in the
        // variable result.
        some_number = 24; // an assignment statement that stores the value 24
        // in the variable some_number.
        result = some_number + 5;
        // uses the + operator to add the value in some_number to 5
        // and stores (assigns) the sum to the variable result, thereby
        // changing result’s value.
    } // end main method body
} // end class VariableExample body
<table>
<thead>
<tr>
<th>Memory after line 1</th>
<th>result</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory after line 2</td>
<td>result</td>
<td>some_number</td>
</tr>
<tr>
<td>Memory after line 3</td>
<td>result</td>
<td>some_number</td>
</tr>
<tr>
<td>Memory after line 4</td>
<td>result</td>
<td>some_number</td>
</tr>
<tr>
<td>Memory after line 5</td>
<td>result</td>
<td>some_number</td>
</tr>
</tbody>
</table>

Michael Eckmann - Skidmore College - CS 106 - Fall 2015
Types for Variables

- Variables can be of the primitive types (int, char, double, etc.) as well as predefined classes provided by Java or a class that a programmer creates.
- For example a variable can be of type `String` which is a class provided by Java.
- We’ll see more of variables and types shortly.