Lecture 3
topics:
from Ch 1 – classes and objects
ch.2 Java fundamentals (sections 1 – 8)
System class, system.out object,
println() ,print()
keywords
escape sequences
primitive datatypes
Unicode
Variables
Math operators
Datatype conversions(promotion and casting)
discuss lab 1 assignment – last step is modified because lab is closed

1. Classes and objects – analogy of blueprint and house

2. The System Class

System is a class that is part of the java.lang package. Java packages are collections of classes. All the packages, taken together, make up the Java Class Libraries, also called the Java API. Java.lang is automatically imported into every Java program, so its classes are available to every program. Every time you write a java program, it automatically has access to the members of the System class. Members are either data declarations(also called "fields") and methods. So it is not necessary to import System to use its methods such as println() and print(). We do this by using the dot operator to link together the Class name, System, with an object of the System class called out, and with a method such as print() that the out object provides us as a service.
System.out.print("hello");
Class.object.method
System.out.println(" , how are you?");

Output to monitor, the default “out” object:
Hello, how are you?

I use this method for debugging my code. This works for both applications and applets:

System.out.println("nTotal = " + nTotal);

In the debug code I just listed, I used string concatenation, using the + operator.
The plus operator is overloaded, which means it takes it meaning from the context of the expression where it is used. If its operands are numbers, it performs addition. If it has a string as one or more operands (or terms), whether literal as in this example, or a string variable, then it will first convert the numeric operand to a string, and then concatenate them.

Examples:
"table" + 4 -> "table4"
"table" + 4 + 5 -> "table45"
but "table" + (4+5) -> "table9" because the parens forces the addition to take place first.

But if we write 4 + 5 + “table” -> “9table” because Java evaluates left to right when all operators are of equal precedence. So first the arithmetic addition is done 4 + 5, then Java sees a string is the next operand, and treats the second + as a string operator. So the 9 is converted to a string and concatenated.
Likewise, \((4 + 5) + \text{"Mickey Mouse"} + 4 + 5\) --> "9Mickey Mouse45"

3. **Keywords**, also called reserved words, do not require memorization, but you should read over the list a few times to start to get familiar with the keywords. Every other word in a program is either an identifier or part of a comment.

4. Writing Escape Sequences: the Escape character - \backslash backslash

following the escape character with a special character, such as "n", creates an escape sequence

a partial list:

\n  newline
\t  tab
\r  carriage return(no new line)
\\  prints backslash
\"  prints quote character

You will find the \n (newline) escape sequence useful in lab 1 to display text over more than one line:

\begin{verbatim}
System.out.println("Here is the first line of text.\n" + "and here is the second line.");
\end{verbatim}

// notice we can write a string expression over several lines of code as long as we break outside a set of quotes. This keeps your program listing from printing off the right edge of the page.
5. **Java Primitive Data Types**

8 in Java:

4 integer types: byte, short, int, long

- byte: -128 to 127
- short: $-2^{15}$ to $2^{15}$ (-32,768 to 32,767)
- int: $-2^{31}$ to $2^{31} - 1$ (~ +2 billion)
- long: $-2^{63}$ to $2^{63} - 1$

2 floating-point: float, double

- float: ~ $10^{38}$, 7 significant digits
- double: ~ $10^{308}$, 15 significant digits

character: char (16 bits)

- Unicode 0 to Unicode $2^{16} - 1$

boolean: boolean (1 bit) true and false, not 1 and 0

Discuss Unicode and Ascii

6. Variables and Literals

The simplest data structure is the variable, a container that can hold only 1 value at a time. Later in the semester we will look at a complex data structure, the array.

Variables are a way for programmers to refer to a memory location without remembering the numerical address of the memory cell.

Naming rules in Java: no spaces or special characters, except for the underscore and $(though$ $should$ $be$ $avoided);$ can be any combination of letters and numbers, but should start with a letter. A variable name is one type of *identifier*, which means a programmer-defined name. Other identifiers are class names, method names, and object references.
Always invent meaningful variable names, so that another person can understand the kind of value you intend the variable to contain. Do not use single letter variable names, with the exception of these for loop counter variables:

x, y, z, i, j, k

Personally, I like to follow a naming convention called Hungarian notation, but I don’t require that you do. Hungarian notation begins variables names with a character that indicates their type, for example,

```java
int nValue;
String sName;
Boolean bDone;
```

Literal values: Unlike variables, which are containers that hold values, literal values are actual values. Some examples:

- “hello” a String literal
- 4 an integer literal (Java assumes an int type)
- 2.5 a floating-point literal (Java assumes a double type)
- true a boolean literal
- ‘h’ a char literal

Demo using envelopes to represent memory and variables.
7. Arithmetic operators

operators and precedence
highest to lowest:
( )
+, - unary
*, /, %
+, - binary
= assignment

give examples of expressions and their evaluation

8. Math class methods

4. The Methods of the Math class (all static)

ex. - int nNewNum = Math.abs(nOldNum);

ex. - // Math.ceil rounds to smallest integer not less than argument
    // (but still typed as a double)
    Math.ceil( 3.2) -> 4.0
    Math.ceil( -3.7) -> -3.0

ex. - // Math.floor round to largest integer not greater than argument
    Math.floor(3.2) -> 3.0
    Math.floor(-3.2) -> -4.0
9. Conversion between Primitive datatypes – **widening** conversion is automatic – to a type with more storage or greater precision – and is called promotion

<table>
<thead>
<tr>
<th>type(bits)</th>
<th>promotion</th>
</tr>
</thead>
<tbody>
<tr>
<td>double(64)</td>
<td>none</td>
</tr>
<tr>
<td>float(32)</td>
<td>double</td>
</tr>
<tr>
<td>long(64)</td>
<td>float or double</td>
</tr>
<tr>
<td>int(32)</td>
<td>long, float, or double</td>
</tr>
<tr>
<td>short(16)</td>
<td>int, long, float, or double</td>
</tr>
<tr>
<td>byte(8)</td>
<td>short, int, long, float, or double</td>
</tr>
</tbody>
</table>

**narrowing** must be done explicitly with a cast

example - widening:
if a variable of type int is copied into a variable of type double, the int ‘s value is temporarily converted to a double

```java
int nNum = 3;
double mySquare = nNum; // 3 will be promoted to 3.0
```

narrowing conversion can only by done with an explicit (cast):

```java
double dNum = 3.4;
int mySquare = (int)dNum;
// 3.4 will be narrowed to 3, with loss of precision
```

Be careful where in the expression you perform the cast. These two expressions yield different evaluations:

```java
(double)( 5 / 2) -> 2.0 because integer division is performed first
```
(double)5 / 2 -> 2.5 because 5 is converted to the double 5.0 before floating-point division is performed

**Mixed Integer operations**

When operands of different integer types are combined in an expression, the evaluated type is int (or long if one or more operands are longs)

When operands of different floating-type and integer values are combined in an expression, the evaluated type is the same as the operand with the Highest rank (largest storage requirement or greatest precision)

Show how to test your ideas about Java on computer – create a test program in jGrasp.

Discuss Lab 1 – Haiku – Because the lab won’t be open next week, you will have to print out Lab 1 at home. Here are changes to the description on the last page of the Syllabus:

Use the print feature of SSH Secure Shell to print out a Unix session on hills, including a listing, compilation, and output of your program. To hand in: a completed lab cover sheet, with the **SSH printout** stapled to the top left corner.

Upload your Haiku.java file to your student account, or create it in the account using the pico editor.

Next display the contents of the Haiku.java file:
$ cat Haiku.java
Next compile it:
$ javac Haiku.java

Next display the long list of files:
$ ll

Next run your Haiku.class file by giving it to the compiler:
$ java Haiku

Finally print a copy:   File/Print