Class Notes 6

eLearning

Current web browsers like Chrome, Safari, Firefox, and Edge all have a behavior called *caching*. A *cache* is a memory or disk copy of an existing website that has been previously visited. In order to speed up the performance of the browser, it is not necessary to re-download every single element of a web site every time it is visited, especially if the pages are mostly static (non-changing).

However, with a site like eLearning, which can change almost any minute, your web browser may not be set up to record those changes on a regular basis, which results in your seeing outdated versions of the site. In every browser, there is a *Refresh* button or menu selection (sometimes assigned the hotkey F5) which you can select to force your browser to download the current web page again, updating it if any changes have been made.

Methods

As a Java programmer, your primary job is to write *methods*. Methods are also called procedures, routines, or functions. They define the *behavior* of a computer program.

Methods provide **encapsulation**, which is a fancy way of saying *information hiding.* The concept of information hiding is that we try to conceal the details of implementation from the callers of the method. This is not a bad thing! By encapsulating the more complex behavior necessary to accomplish a task within a method, we allow the method callers to put together programs that are more abstract and easier to understand.

```
// this is self-explanatory
bankAccount.deposit( 150.00 );
bankAccount.withdraw( 20.00 );

// versus this:
account2213091412312
unit 123124, subsection 2189412B1312AFE
```

```
place in memory location 232145r12BSAEWfew23
amount !@#21421#$#@%@#@!34123242
.
.
.
(a couple dozen more lines of code)
```

--

More to the point, as part of program design, writing methods gives us the ability to break down one large complex task that takes hundreds of lines of code, down into many simpler tasks, that probably take fewer lines of code to complete. From there, each individual method can be tested or verified for functionality, and when those are demonstrated to work properly, they can be combined to form the larger program.

This is the *divide and conquer* principle you may have discussed previously in world history (see *The Art of War*).

--

Method Declaration

```
public static void main( String[] args )
```

Access Level - public or private

Static/non-static (optional) - you use static when you have written methods that are shared in the same class as a main() method, so that the main() can access those methods directly.

Return Type - a primitive or abstract data type

Name - should be a *verb*

Parameters - data that you want to give a method so it can use it in calculations

Methods can accept any number and type of parameters.
Methods can only return ONE data element.

Naming Conventions

In Java, we follow some "rules" so that it makes things easier for other programmers to read our code. We dictate certain styles for writing code for uniformity in group programming situations.

Variable names should start with a lower-case letter. Multiple word variable names should appear like this: `thisIsSomeVariable`. This is called *camel notation*.

C and Python use underscores instead:
`this_is_some_variable`

Methods Again

The use of methods in programming heralds a different paradigm (or way of doing things) than traditional line-by-line structured programming. We call this paradigm **procedural programming**. The availability of functions transformed the way that programmers looked at solving problems.

Solving problems by taking complex tasks and breaking them up into smaller subtasks is called **functional decomposition**.

Programming Exercise

Write a program that determines if an array of integers contains more numbers that are perfect squares or not. Write a method that performs the check for perfect square.

To determine if a number is a perfect square:
1. Take the square root of the number.
2. Truncate the decimal.
3. Square the number.
4. Compare the result to the original number.

Hints:

java.lang.Math has some useful methods:
`sqrt()`

How does one truncate a double?
Typecast it as an integer.

```
double num1 = 6.23;
int num2 = (int) num1;
System.out.println( num2 );
output: 6
```

Homework Project

Write a program that displays a calendar in textmode for a given month of the year. The project should be named **Calendar**. Here is a sample run of the program:

```
Enter full year: 2006
Enter month: 6

         June 2006
---------------------------
Sun Mon Tue Wed Thu Fri Sat
                  1   2   3
  4   5   6   7   8   9  10
 11  12  13  14  15  16  17
 18  19  20  21  22  23  24
 25  26  27  28  29  30
```

Your program will prompt the user for the full year and month, then display the calendar as show above.

This a somewhat complex task with lots of calculation involved, and it is unlikely that we would be able to code it all in a single main function.

The first thing we will do is create a **program framework**. This is a "shell" of a program that lets us identify WHAT the program intends to do without hasving to figure out exactly HOW it will accomplish it yet. In software development, we call this looking at the "big picture" or **top-down design**.

Once we can determine all the parts that we need to put the program together, after performing functional decomposition to get all of the subtasks, we will then engage in **bottom-up implementation**, where we code and test all the individual methods, and combine them together to form a complete program. We start with a top-level stub, then move down to the individual program elements.

The framework code provides us a way to see how all of the data is moving through the system. We have a clear understanding of what the program is trying to accomplish. Our next task is to break up the rest of the program's functionality into smaller subtasks that are simpler and easier to code. Once those are complete, we can test them individualy, then integrate them into the complete program.

Your job is to code, test, and integrate all these small parts through bottom-up implementation. Do not move on to the next component until you hae completed and tested the current one. In order, from smallest to largest:

METHODS TO IMPLEMENT:

isLeapYear()
Determines whether a year is a leap year or not. Returns a boolean value.

getDaysInMonth()
Determines how many days are in a given month of a given year.

getTotalNumberOfDays()
Determines how many days have elapsed since January 1, 2000, which is a Saturday. We call this our *epoch* date.

getStartDay()
Figures out what day of the week we will start printing the days at.

printDay()
Prints out a single day on the calendar--watch out! Some dates require two spaces and a number, while some require one space and two numbers.

printMonthBody()
Combine the necessary methods above (helper methods) to complete this method.

printMonthTitle()
This can be written at any time, as it does not require any helper methods.