COSC–111 Introduction to Computer Science I
Lab 05: Methods
Due: Friday, March 22, 8.59am

1 Writing methods to do unit conversions

1.1 The units

The core of this lab assignment will be *unit conversions*—translating a measurement in one unit into some other unit. In particular, we will use a few different measurements of *length*, and a few of *time*. Here are the units we will use:

- **Measurements of length:**

Unit name	Conversion
meter (m)	The canonical unit
foot (ft)	0.3048 m
inch (in)	$\frac{1}{12}$ ft
yard (yd)	3 ft
smoot	1.7018 m
barleycorn	$\frac{1}{3}$ in

- **Measurements of time:**

Unit name	Conversion
second (s)	The canonical unit
minute (m)	60 s
jiffy	$\frac{1}{60}$ s
helek	$\frac{10}{3}$ s

1.2 Getting started

Now that you have some units with which to work, it is time to start a new program for yourself. Specifically:

1. Create a directory for this project, change into it, and grab source code:

```
$ mkdir lab05
$ cd lab05
$ wget -nv -O Converter.java http://bit.ly/111s19L05F
```

2. Open the new source code file in *Emacs*:

```
        $ emacs Converter.java &
```

You will see, in this source code file, the beginnings of a program named `Converter`. It contains, for starters, two complete methods named `convertInchToFoot` and `convertFootToInch`. Given the conversion factor from one to the other, you should, as these methods show, easily be able also to use these to perform the inverse conversions.

Your first task: For each of the conversions listed above, write a method to perform that conversion **and** its inverse (e.g., inches to feet **and** feet to inches). Each method should follow the same form as `convertInchToFoot`. Its caller should provide a `double` value; the method should return the conversion as another `double` value; for converting from unit *Foo* to *Quux*, the method should be named `convertFooToQuux`, using **that exact format**: start the method name with the word "convert" in lowercase, then take the two unit names as they appear in the table and capitalize the first letter, separating the two unit names with the word "To" (capital T). (The choice of name matters for my testing code, which will call your methods. If your methods are not named correctly, my code will not work correctly. In general, having naming conventions for methods helps your code to be more readable to other humans—including yourself!)

After writing these methods, you should *test them.* It is always a good idea to determine, with pen and paper, what the results should be for a few different inputs, and these methods are no different. How do you test your methods? Write for yourself a `main` method—just as we have since the first day of class—and call on your various conversion methods. Print the results to the screen to see if they come out correctly.

2 Methods using other methods

Imagine writing your program to do the following:

```
Enter a number of meters: 1000
1000.000000 meters = 118110.236220 barleycorns

Enter a number of jiffies: 1000
1000.000000 jiffies = 5.000000 heleks
```

Your second task: Write `main` such that it performs these tasks. Specifically, it should prompt the user for a number of meters, and then convert those meters into barleycorns. Likewise, it should convert the user-provided number of jiffies into heleks. The output should look exactly as shown here (though if your output includes a different number of zeros or a slightly different rounding at the far-right decimal places, that is fine). Most importantly, **you must not write any new conversion methods**, but rather use the ones that you already have in perform these conversions.

3 Submitting your work

Submit your `Converter.java` file with the CS submission system, using one of the two methods:

- **Web-based:** Visit the submission system web page.
- **Command-line based:** Use the `cssubmit` command at your shell prompt.