Homework 1

Due Date: Saturday, September 14th, 11:59 pm

To install: `cs0150_install hw1`
To submit: `cs0150_handin hw1`

Note: All of the above commands should be run in a terminal.

Silly Premise

It's that time of year again—Kevin is making his famous chili with a hint of secret crab sauce for the Office today. This time though, he's making it in the Dunder-Mifflin office kitchen to make sure he doesn't drop the ball (or the chili) again. He needs your help to make this feast a success!

For this homework, you are going to be writing your first Java program! If you are confused about Linux or navigating the Terminal for the following problems, Lab 0 will cover those concepts, but you can get started beforehand.

Half of this assignment will be coding, and the other half will be written. We have provided a `README` template for you to put your written answers inside. It will be turned in along with the rest of your files.

Getting Started

Install this homework by typing **`cs0150_install hw1`** into a terminal. This will create an empty directory in `/home/<your login>/course/cs0150/hw1/`. **Please do not include any identifying information on your handin (name, login, Banner ID) as we grade anonymously. Including identifying information will result in a deduction from your assignment.**

Collaboration Policy and Important Documents

Before you get started, however, please first read our Collaboration Policy. **You will not be graded or receive credit for any assignment**, including this one, **unless you have read the Collaboration Policy and signed the form linked below.**

Question 0 [Write your answers in `README`—see instructions below]

a. Fill out and sign the Collaboration Policy Quiz and Agreement. The quiz is not graded. **Additionally, you are bound by this contract for this and all future assignments.**
b. In a paragraph, summarize the CS15 collaboration policy as you understand it, making sure to cover the policies on homework, lab and design sections, conceptual hours, and course material. In addition, give your own rationale for this CS15 specific collaboration policy.

Next, read through the Style Guide found on the website. You are responsible for all information in this document. Points will be deducted from any project or homework for not following these guidelines.

Installation, compiling, and running your code

- Type **`cd ~/course/cs0150/hw1`** to navigate to the folder where this homework is stored.
- Type **`ls`** in the terminal to see the files in the folder. You should see three files: `App.java`, `Chef.java`, and `README`.
- Type **`atom *`** to open all the files. There's a few lines of code already written in each file, and the first thing you should do is familiarize yourself with them. We've added comments in the stencil to walk you through. Look at the syntax carefully, starting with `App.java`, and be sure to reference it later as you start adding to the stencil.
- To compile your code, type **`javac *.java`** in your terminal
    - If your code compiled successfully, nothing will print in your terminal. A new file called `App.class` and `Chef.class` will appear in your directory. You can confirm this by again typing `ls` in your terminal.
- After you compile your code successfully, run your code by typing **`java hw1.App`**.

Assignment

Question 1 [Write your answers in `README`]

You should see three lines of text printed in your console:

```
Hello world!
A new Chef named Kevin has been made.
Washing hands, putting on an apron, and prepping the kitchen...
```

For each line, search through App.java and Chef.java to **write down the name of the method that prints each of the above lines out**. (If it's a constructor, simply write "<class> constructor").

> The `System.out.println()` command prints out text into your terminal. This is a great way of confirming your program has reached a certain point in the code, and will become one of your best friends throughout the course.

Question 2 [Write your answers in `README`]

Head over to the `Chef` class. Right now, a `Chef` can only really `getReady()`, which isn't too helpful for us. We also want them to be able to make food! We'll write a method to do that soon, but first let's recap method definitions.

Examine the following method:

```
public int add(int firstNum, int secondNum) {
    return firstNum + secondNum;
}
```

In the `README`, **match the colored terms from the above code** with the terms *method name, method definition/body,* and *parameter name*, according to their purpose.

Note: you can ignore "public int" and "return" for now. We'll see them in lecture later.

Question 3 [Code your answers in `Chef.java`]

Great! Onwards to writing our first Java method!

Write a public method in `Chef.java` named `makeFood`. It should take in a single *parameter*, a String named `foodToMake`. When it's called, it should **print out the following line**:

```
Making <foodToMake>!
```

> Note: Remember that <> is just a placeholder (see Lecture 2, slide 11) and `String` is a variable type that is just text.
>
> **[Hint]**: Take a look at how we use it in both the `Chef` constructor and when we create `kevin`.

To clarify, here we just ask you to define the method, not execute it. (That comes later!) When you run the program, nothing should change in your console output.

Speaking of which, now is actually a good time to compile your program again (`javac *.java`), and make sure you don't get any errors that came from your recent code addition. This is called incremental programming, and it's very important to do this as much as possible to make sure you catch bugs early on.

Question 4 [Write your answers in `README`]

Now let's go over this code from lecture:

```
// samBot is a Robot
samBot.moveForward(3);
```

Regarding what this code is doing, answer these three questions in the README:

a) What is `samBot`?
b) What is `moveForward`?
c) What is `3`?

Yay! Now you're ready to call the method you just wrote.

Question 5 [Code your answers in `App.java`]

Earlier we made a Chef named `kevin` and he hasn't really done much yet. **In `App.java`, call our `makeFood` method on `kevin`, and have him make chili.** In other words, when you compile and run your program, the following lines should print out:

```
Hello world!
A new Chef named Kevin has been made.
Washing hands, putting on an apron, and prepping the kitchen...
Making chili!
```

What if we want more than one `Chef`? **After `kevin` makes chili, create another `Chef` named Jim, and save it in a variable called `jim`.**

Remember that variable names always start with lowercase!

Question 6 [Write your answers in `README`]
At this point, **what lines print out in the console**? *Hint: you should have 6 lines total.*

Question 7 [Code your answers in `App.java`]

Jim wants to make some food for his "friend" Pam. You can decide what he makes, but call the `makeFood()` method on `jim`!

Compile and run your program now. Does the console say what you expect it to say? If not, look over the code again and follow it from method to method to see where each line is coming from.

Handin Information

This assignment must be submitted no later than **11:59 pm** on **Saturday, September 14th.** There is no late handin for this assignment. Please remove any identifying information from your handin. When you've finished the assignment, run **`cs0150_handin hw1`** in a terminal to submit. This will turn in all of your **`.java`** files and your **`README`**. When you have successfully handed in the assignment, a confirmation email will be sent to your CS department account (`<yourlogin>@cs.brown.edu`), which forwards automatically to your regular Brown email.

You must submit this electronically through the handin script! If you do not handin via `cs0150_handin` and instead email your handin to the TAs (even before the handin deadline), you will lose 25% of the total available points.