Name: (as it would appear on official course roster)	
Umail address: @umail.ucsb.edu	section 9:30am or 11am
Optional: name you wish to be called if different from name above.	
Optional: name of "homework buddy" (leaving this blank signifies "I worked alone"	

1

h06: Packages and JAR files (HFJ 17, pp. 581–595)

h06

CS56 M18

ready?	assigned	due	points
true	Thu 08/16 02:00PM	Mon 08/20 09:00PM	100

You may collaborate on this homework with AT MOST one person, an optional "homework buddy".

MAY ONLY BE TURNED IN IN THE LECTURE/LAB LISTED ABOVE AS THE DUE DATE,
OR IF APPLICABLE, SUBMITTED ON GRADESCOPE. There is NO MAKEUP for missed assignments;
in place of that, we drop the three lowest scores (if you have zeros, those are the three lowest scores.)

NOTE: This assignment is due by electronic upload to Gradescope.

See this link for additional information: https://ucsb-cs56-m18.github.io/static/gradescope.pdf

Reading Assignment: HFJ Chapter 17, ONLY pages 581 through 595. The material from p.596 to p.601 about **Java Web Start (JWS)** is not very important anymore (this technology really didn't catch on).

Similarly, we are not going to cover the material in Chapter 18, since there are better sources for learning about distributed computing in Java than this chapter. Pages 581 through 595 conclude our first pass through the material in the Head First Java Book; the next assignment starts our work in Head First Design Patterns.

We may also revisit chapters in HFJ we've already read through at least once to dig into some topics in more detail.

Finally: note that in Chapter 17, the name of the `classes` directory and the name of the `source` directory are only examples. In this assignment, we are calling those directories `build` and `src`, which is a more common name for them.

This is done not to confuse you, but to make sure you are paying attention and trying to understand the material.

You'll need to be able to switch easily between hand compiling, using `ant` and `maven. Each of those uses slightly different naming conventions for the directory that stores the` .java `and the` .class` files.
So we might as well start practicing that now.

Oh: and in case you think that the IDE (Eclipse, Netbeans, IntelliJ, etc.) is just going to take care of all of this for you: good luck with that.
That works for a while—right up to the point where it doesn't.
And at that point, you need to be able to look under the hood and figure things out without help from the IDE.

1. (10 pts) Please fill in the information at the top of this homework sheet, including your name and umail address. Put the time your discussion section starts (9:30am or 11am) in the space indicated (the one you are registered for—even if you usually attend a different one.) If the other two items apply, please fill them in as well. Please do this every single time you submit homework for this class.

2. (10 pts) What common naming convention is used to prevent package name conflicts (collisions)?

3. (10 pts) If the file `foo.jar` is an executable `.jar` file, how do you run it at the command line?

4. Suppose a java class `Bar` is in the package `edu.ucsb.cs56.foo`, and that your entire project lives in a directory called `~/cs56/Fum`. (Perhaps `Fum` is a regular directory, or perhaps its the name of a Github repo that you cloned, but either way, its the directory in which your project lives.)

   Further, suppose that you are keeping your source files under `~/cs56/Fum/src`, and your `.class` files under `~/cs56/Fum/build`.

   a. (10 pts) What should be the full path, starting from `~`, that of the `.java` source file for class `Bar`?

   b. (10 pts) What line of Java code must appear first, before any import statements in that source file?

   c. (10 pts) Suppose you wanted to compile without `ant`, using the techniques described in Chapter 17. Write down the full `cd` command you would use to get into the proper directory, and then the full `javac` command you would use to compile the `Bar` class.

   d. (10 pts) Give the full filename starting from `~` to the resulting `.class` file.

   e. (10 pts) Part of creating an executable jar is creating a manifest file. Suppose you want to create an executable jar that would run the `Bar` class from package `edu.ucsb.cs56.foo`, with all the same assumptions as above. What would the filename of your manifest file be, and in which directory should you put it?

   f. (10 pts) What would the contents of the manifest file be?

   g. (10 pts) Assuming you did all of the above, what directory should you be in, and what command you should you type to create a file `~/cs56/Fum/fum.jar` that is a an executable jar that would run the `Bar` class from package `edu.ucsb.cs56.foo`?