Name: *(as it would appear on official course roster)*		
Umail address:	**@umail.ucsb.edu**	section 4, 5, or 6
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

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

H01: Due Thursday, 01.07 in Lab

H01

CS56 W16

Variables, Types (double vs. float, primitive vs. reference etc.) Instance Variables, Methods (HFJ Ch3,4)
Assigned: Mon 01.04 Total Points: 46

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 five lowest scores (if you have zeros, those are the five lowest scores.)

Reading Assignment: Throughout the quarter, when I refer to **HFJ**, this means your Head First Java, 2nd Edition textbook.

- Read HFJ:Chapter_3 (especially pages 59-62) and reading notes on the wiki
- Read HFJ:Chapter_4 and reading notes on the wiki

Please:

- **No Staples**.
- **No Paperclips**.
- **No folded down corners**.

1. (6 pts) Fill in the homework header properly—this helps us keep the grading pipeline flowing so that you get credit for your work and get feedback more quickly.
   - writing **either 4, 5, or 6** to indicate your **discussion section (lab)** meeting time
   - entering BOTH your name AND your umail address EVERY time.

   Paper submissions: One sheet of 8.5x11 paper double sided, or two DISCONNECTED SHEETS with your name on EACH. Please: **NO STAPLES, NO PAPERCLIPS, NO TAPE, NO ATTACHMENT OF ANY KIND**. These damage the document scanner.

   Scanned submission: When submitting by PDF upload: scan your pages legibly and **SCAN IN THE CORRECT ORDER**. Page 1 first, then Page 2, in the correct orientation. Failure to scan properly may result in zero credit, meaning you "use up" one of your five "drop the lowest grade" slots.

2. Based on your reading in HFJ Chapter 3:

   a. (4 pts) If I write 3.4, is that of type double, or float?

   b. (4 pts) Declare x as a double and assign it the value 3.4 (as a double)

   c. (4 pts) Declare y as a float and assign it the value 3.4 (as a float)

3. (5 pts) In C++, the name of a plain old array of `Student` objects is not an object, but is rather a pointer to a `Student` (i.e. it is of type `Student *`. What about in Java—is an array an object, yes or no?

4. Variables that represent a primitive type (e.g. `boolean x;` or `int y;`) and variables containing object references (`String w;` or `Student z;`) have this in common—they are both composed of bits in memory. But—as explained in HFJ Chapter 3—they differ in what the bits *actually* represent. You won't get this one by just guessing—you really have to read the book.

   a. (4 pts) What do the bits that represent `int y;` represent?
   Assume that y is assigned the value 13

   b. (4 pts) What do the bits that represent `String w;` represent?
   Assume that w is assigned the value "foo".

5. Consider these questions about memory—answers are in Chapter 3 of HFJ.

   a. (2 pts) Does the amount of memory taken up by an object reference differ for different kinds of objects (say String vs. ArrayList<String>?)

   b. (2 pts) Does the amount of memory taken up by the object itself differ for different kinds of objects (assuming the same JVM)

   c. (2 pts) Can the amount of memory taking up for an object reference for a object particular type (say `String`) differ from one JVM to another?

6. Based on your reading in HFJ Chapter 3, p. 59-62 and HFJ Chapter 4 p. 84:

   a. (4 pts) Suppose I have a class called Student. How do I declare and allocate space for a plain old Java array called `students` that can hold 5 references to Student objects?

   b. (5 pts) Java `for` loops look pretty much just like C++ `for` loops (see HFJ page **10** if you really need to check.) Given that, assuming there is a default constructor `Student()` that you can call to create a new `Student` object, write a `for` loop that initializes all of the elements of the array `students` (from the previous problem) to be instances of the `Student` class.