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".

H11: Due Friday, 01.22 in GradeScope

H11

CS56 W16

Serialization and File I/O (HFJ Ch 14, JPG Ch 11)
Assigned: Wed 01.13 Total Points: 50

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:

- HFJ:Chapter_14, starting on p. **429** Serialization and File-IO: Saving Objects
- In addition to HFJ (Head First Java). The other textbook for the course is the Java Pocket Guide, which I'll refer to as **JPG**. Both HFJ and JPG have their own pages on the wiki with reading notes.
- In JPG, read Chapter 11, which is about the same topics as Chapter 14 in HFJ, i.e. Input and Output (especially File IO and Serialization).
  - You may skip the Section titled "Socket Reading and Writing" on pages 101-103. But read all of the rest.

---

1. (5 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.

Based on your reading in HFJ Chapter 14 and JPG Chapter 11:

2. (5 pts) How do you indicate that part of an object that implements the Serializable interface should NOT be saved?

3. (10 pts) Write a brief main method that writes the string "Hello Disk!" to a file called "myFirstFile.txt".

4. Figure 11-1 on p. 98 of JPG shows an inheritance chart with four base classes: Reader, Writer, InputStream and OutputStream, and various kinds of derived classes that inherit from these.

   a. (8 pts) Fill in the blanks: According to the text, Reader and Writer (and implicitly, the classes that inherit from them)

      are used mainly for _____, while

      InputStream and OutputStream are used mainly for _____

   b. (6 pts) According to JPG, what is the situation where a Buffered version of an InputStream or OutputStream is appropriate?

5. (6 pts) The File class in Java provides methods that correspond to several Unix commands. What is the equivalent java code to the Unix command: mv foo.txt bar.txt Note: You may need more than just the information in the chapters—you might also need to consult the Javadoc for the File class online, and maybe even try out your code to see if it works. I dont need the whole class—just write the code that would appear inside a main() method:

6. (10 pts) For this question, consider the reading in HFJ about serialization. Your friend B. C. Dull says: "I don't get why object serialization is such a big deal. You have a pointer to each object, and you know how many bytes it takes up in memory, and you know the objects type. Just write those bytes to a file, along with a few extra bytes indicating the type—problem solved. Then you read those bits back in and restore the objects. What's the big deal". You however, see more deeply into the situation, and say: "Well, B.C., it isn't quite that simple. You are forgetting a few subtle issues—things you should realize from your previous study of how data structures work—things you should have learned in CS16 and CS24. Even though those were C/C++ courses, the same problems are going to arise in Java." B.C. says: "I don't see what you are getting at. Can you explain it to me?" What do you say to B.C. to help him/her realize why Object Serialization is more subtle than just writing out all the bits exactly as they are, and reading them back in exactly as they are? HINT: Think about things like linked lists, pointers, and references, and how they are implemented in both C++ and Java.