First name (color-in initial)	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	section (9,10,or 11)	first name initial	last name initial
Last name (color-in initial)	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z			

H02: Due Monday 04.09.2012 in Lecture. Total Points: 50

MAY ONLY BE TURNED IN DURING Lecture ON Monday 04.09.2012, or offered in person, for in person grading, during instructor or TAs office hours.

See the course syllabus at https://foo.cs.ucsb.edu/56wiki/index.php/S12:Syllabus for more details.

---

(1) (10 pts) Fill in the information below. Also, fill in the A-Z header by

- **coloring in** the first letter of your first and last name (as it would appears in Gauchospace),
- writing **either 9,10,11** to indicate your **discussion section** meeting time
- writing your **first and last initial** in large capital letters.

All of this helps us to manage the avalanche of paper that results from the daily homework.

name:	
umail address:	@umail.ucsb.edu

Reading Assignment:

Throughout the quarter, when I refer to **HFJ**, this means your Head First Java, 2nd Edition textbook. The other textbook for the course is the Java Pocket Guide, which I'll refer to as **JPG**. Each of these has its own page on the wiki with reading notes.

- Read HFJ, Chapter 3 (especially pages 59-62) and reading notes at HFJ:Chapter_3
- Read Chapter 4 and reading notes at HFJ:Chapter_4
- Read HFJ, page **560** (just that one page)
- Read HFJ, Chapter 17 , ONLY pages 581 through 595, and reading notes for those pages at HFJ:Chapter_17

Why are we skipping ahead?—I know that students sometimes get nervous when we "skip ahead" in the textbook, so I try to do it only when there's a really good reason. This week, there is.

The material on p. 560 really should have come right next to p. 86 in my opinion—it is about the difference between == and .equals(), which is a very common source of errors among beginning (and advanced) Java programmers. So we are going to move it where it should have been all along (IMHO).

The material in Chapter 17 is all about "Releasing your Code"—i.e. making it available to the rest of the world. Pages 581-595 describe how to put code into packages, and JAR files. This material doesn't really depend on any of the material in Chapters 5-16 (at least not very much.) And since the emphasis in this course is on producing "open source" software, I want to be sure that we are comfortable with making JAR files and putting our code in packages. Chapter 17 should help make some of what we are doing in lab with those topics a bit more clear.

The rest of the chapter is about Java Web Start—we'll cover that once we start talking more about GUIs.

(2) Based on your reading in HFJ Chapter 3:

- (1 pts) If I write 3.4, is that of type double, or float?
- (2 pts) Declare x as a double and assign it the value 3.4 (as a double)
- (2 pts) Declare y as a float and assign it the value 3.4 (as a float)

(3) 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.

- (2 pts) What do the bits that represent `int y;` represent?
  Assume that y is assigned the value 13

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

(4) Consider these questions about memory—answers are in Chapter 3 of HFJ.

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

- (1 pts) Does the amount of memory taken up by the object itself differ for different kinds of objects?

- (1 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?

(5) (3 pts) Based on your reading in HFJ Chapter 17:

To be able to run a jar file directly with `java -jar blah.jar` it is necessary for the jar file to contain a "manifest". What **is** a manifest, and **what information** does it contain?

(6) (3 pts) From Chapter 17: Briefly, what is the purpose of having multiple "packages" in Java?

(7) (3 pts) From Chapter 17: How do you indicate in the source code that a class is part of a package?

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

- (2 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?

- (2 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.

- (1 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?

(9) (10 pts) Based on your reading in HFJ Chapter 4:

Consider the following Java code.

- Will this code produce an error message, when compiled with `javac *.java` and if so what? (I don't need a detailed character by character account of the error messsage—just a general description of what the error is will be sufficient.)

- If it does compile: will this code produce an error message, when run with `java StudentTestDrive` and if so what? (same as the previous question—just a general description of the error is sufficient.)

- If this code does NOT produce an error message when compiled or run, what will be the resulting output when this code is run?

Contents of `Student.java`

```
class Student {
    private int perm;
    private String name;

    public int getPerm() {
        return perm;
    }
    public String getName() {
        return name;
    }
}
```

Contents of `StudentTestDrive.java`

```
public class StudentTestDrive {
    public static void main (String[] args) {
        Student s = new Student() ;
        System.out.println("Student's perm is " + s.getPerm() ) ;
        System.out.println("Student's name is " + s.getName() ) ;
    }
}
```

(10) (4 pts) p. 560 discusses two kind of equality: reference equality and object equality. For each of the statements below, indicate whether the statement is true of reference equality or object equality.

Statement	Circle one
This type of equality refers to "meaningful equality" of two objects: e.g. two book objects that have the same values for title, author, and publication year.	Reference equality Object Equality
This type of equality is what you get when you use the == operator.	Reference equality Object Equality
This type of equality checks whether two variables refer to the same object on the heap	Reference equality Object Equality
Making this type of equality work properly involves overriding the .equals() method and the .hashcode() method.	Reference equality Object Equality