Name:	Seat:
(as it would appear on official course roster)	

Umail address:	
	@umail.ucsb.edu

EXAM: e02: Midterm 2

ready?	date	points
true	Thu 11/16 09:30AM	100

You may not collaborate on this exam with anyone. If you need to use the restroom, you must leave your cell phone with the exam proctor before leaving the room.

- Write your name at the top of this page **AND EVERY ODD NUMBERED PAGE.**
- Double check that you turned in ALL pages; look for "End of Exam" on the last page.
- This exam is **closed book, closed notes**, **closed mouth, cell phone off.**
- You are permitted **one sheet of paper** (max size 8.5x11") on which to write notes.
- This sheet will be collected with the exam, and might not be returned.
- Please write your name on your notes sheet.

The first two questions are essay/short answer type of question, with an emphasis on "short answer". For this and *all other similar questions on this exam*, answer as if you were in a *job interview*. Your answer will be graded on the basis of whether it would be likely to help you or hurt you in that context. Answers gain or lose points based on whether they are accurate, precise, concise, relevant, and whether they address the specific question asked.

1. (10 pts) What is a "Design Pattern"? (See the note above about keeping essay/short answer questions short and to the point.)

2. (10 pts) A common misconception about Java threads is that when you invoke the `.start()` method on them, they begin running immediately. Explain what happens instead.

3. For this question, you need the additional handout A with code for these files: `Beverage.java`, `Edible.java`, `Food.java`, `FreeCandy.java` and `Product.java`. These are classes used by a grocery store known as "Partial Foods".

You may assume that all of the code for these files on the handout compiles— I've checked that this is true. Now consider the code for the `PartialFoods` class listed below.

Please do these things with this "broken" code for the `PartialFoods` class:

a. (15 pts) Several lines (more than four, but fewer than sixteen) need to be eliminated from this file in order to make it compile.

Find the lines that are bogus, and for each, * draw a line through each of them in the code listing below. * ALSO put an X in the `/* [ ] */` comment, like this: `/* [✗] */`

For example, if you decide that 4 is bogus, your paper should look like this:

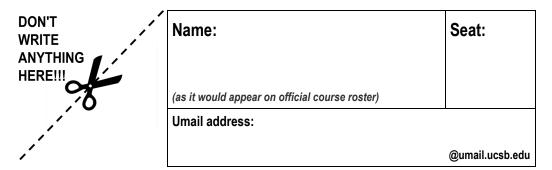`/* [✗] */` ~~Beverage a = new Beverage(99,"Coke",150,12.0);~~

Start by determining which, if any, of the constructors are bogus. Then, eliminate any lines that refer to the variables created on those lines. Finally, check all of the remaining method calls.

You will lose two points each time you striking a line that is not bogus, and you will lose two points for failing to strike any line that IS bogus. So, choose wisely.

```
1 public class PartialFoods {
2   public static void main (String [] args) {
3
4     /* [  ] */ Beverage a = new Beverage(99,"Coke",150,12.0);
5     /* [  ] */ Edible b = new Beverage(89,"Diet Coke",0,12.0);
6     /* [  ] */ Food c = new Food(249,"Kind Bar",200,1.4);
7     /* [  ] */ Edible d = new Edible(199,"Gummi Bears",520,5);
8     /* [  ] */ FreeCandy e = new FreeCandy(25);
9     /* [  ] */ Edible f = new FreeCandy(30);
10    /* [  ] */ Product g = new Product(299,"Ziploc Bags");
11    /* [  ] */ Product h = new FreeCandy(30);
12    /* [  ] */ Product i = new Product(299,"Ziploc Bags");
13    /* [  ] */ Product j = new Beverage(79,"Pepsi",150,12.0);
14    /* [  ] */ Product k = new Food(125,"Doritos",260,1.7);
15
16    /* [  ] */ System.out.println("a: " + a.getCalories());
17    /* [  ] */ System.out.println("b: " + b.getPrice());
18    /* [  ] */ System.out.println("c: " + c.getFluidOunces());
19    /* [  ] */ System.out.println("d: " + d.getCalories());
20    /* [  ] */ System.out.println("e: " + e.getCalories());
21    /* [  ] */ System.out.println("f: " + f.getName());
22    /* [  ] */ System.out.println("g: " + g.getName());
23    /* [  ] */ System.out.println("h: " + h.getPrice());
24    /* [  ] */ System.out.println("i: " + i.getName());
25    /* [  ] */ System.out.println("j: " + j.getCalories());
26    /* [  ] */ System.out.println("k: " + k.getWeight());
27  }
28 }
```

b. (5 pts) After striking through the bogus lines, the remaining code should compile and run. So, indicate what the output will be (if any) below.

For full credit, be precise about exact formatting, including line breaks, and the constant string parts such as `"a: "`, `"b: "` and so forth. If there will no output, write "no output". There is no partial credit for this problem.

Name:

Seat:

(as it would appear on official course roster)

Umail address:

@umail.ucsb.edu

4. (10 pts) Suppose you have a Java program that implements a server for a networked game. Game players have to connect to a server to play; your code is implementing that server.

You start up the program, and you see this exception: `java.net.BindException: Address already in use`

What, specifically, do you look for in the code to determine how to fix the problem?

5. (20 pts) Using only the space provided below, describe as clearly as possible (as if you were asked in a job interview) to describe the legacy code project you are working on. Include:

- a description of what the software does
- what, specifically, you have done already to make the code or product better
- what, you hope to have accomplished by the end of the quarter

You will be graded not only on content, but also on the clarity of your writing.

4

e02

CS56 f17

DON'T
WRITE
ANYTHING
HERE!!!

6. (10 pts) Please refer to the following handouts for this problem, and all of the
   remaining problems on this exam:

   ○ p. 1 of <u>Handout B</u> which has several reminders about `Comparable`,
   `Comparator`, etc.
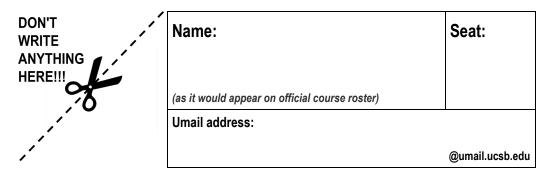   ○ p. 1 of <u>Handout C</u> which has incomplete code for a `Student` class.

   Consider the following class that uses `Student`, called `StudentSort`.

```java
1  // StudentSort.java
2  import java.util.ArrayList;
3
4  class StudentSort {
5
6      public static void main(String [] args) {
7          ArrayList<Student> roster = new ArrayList<Student>();
8          roster.add(new Student(6234567,"Bob",3.25,"CMPSC"));
9          roster.add(new Student(7152353,"Charlie",3.96,"CMPSC"));
10         roster.add(new Student(2352353,"Alice",3.97,"CMPEN"));
11         roster.add(new Student(2888888,"Danielle",4.00,"CMPSC"));
12         roster.add(new Student(1152353,"Ed",2.90,"CMPEN"));
13
14         java.util.Collections.sort(roster);
15
16         for (Student s: roster) {
17             System.out.println(s);
18         }
19     }
20 }
```

Suppose we want line 14 of this code, `java.util.Collections.sort(roster);` to sort in lexicographic order by name (Alice, Bob,
Charlie, Danielle, Ed). In order for this code to work properly, and indeed for the `Student.java` class to even compile, at least one extra method
must be added.

Please write the additional method that would be added Student.java in the space below.
Please try to avoid writing too close to the end of the page (it doesn't scan well.)

DON'T WRITE ANYTHING HERE!!!

Name:

(as it would appear on official course roster)

Umail address:

@umail.ucsb.edu

Seat:

5

e02

CS56 f17

7. (10 pts) Please continue to refer to:

- p. 1 of Handout B
- p. 1 of Handout C

Consider the following Java code that also makes use of the Student class. As you can see, in order for this code to compile, two additional methods would have to be added to the Student class, namely the ones invoked on lines 16 and 17.

On p. 2 of Handout B, you will see the output of various invocations of this program (assuming the necessary change are made to the code, including the methods you added `Student.java` in questions 4 and 5, and the implementations of these two additional methods.

For this question, we focus ONLY on the method `sortByName` invoked on line 16.
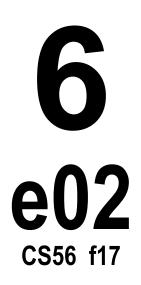
```
1  // StudentMain.java
2  import java.util.ArrayList;
3
4  class StudentMain {
5
6      public static void main(String [] args) {
7          ArrayList<Student> roster = new ArrayList<Student>();
8          roster.add(new Student(6234567,"Bob",3.25,"CMPSC"));
9          roster.add(new Student(7152353,"Charlie",3.96,"CMPSC"));
10         roster.add(new Student(2352353,"Alice",3.97,"CMPEN"));
11         roster.add(new Student(2888888,"Danielle",4.00,"CMPSC"));
12         roster.add(new Student(1152353,"Ed",2.90,"CMPEN"));
13
14         if (args.length > 0 ) {
15             switch (args[0]) {
16             case "name":     Student.sortByName(roster); break;
17             case "perm":     Student.sortByPerm(roster); break;
18             default: break;
19             }
20         }
21
22         for (Student s: roster) {
23             System.out.println(s);
24         }
25     }
26 }
```

In the space below, please write an implementation of `sortByName` that sorts the roster by name as indicated on p. 2 of Handout B.

For full credit: please write the body of this method as a single line of code.
Specifically: use a line of code that takes advantage of the fact that `Student` implements `Comparable<Student>`.

For half-credit, you may use any technique that sorts the array correctly.

8. (10 pts) Please continue to refer to:

- The `StudentMain.java` file on p. 5 of the exam.
- p. 1 and p. 2 of Handout B
- p. 1 of Handout C

Now, please write the code for the `sortByPerm` method invoked on line 17 of `StudentMain.java`.

For Full Credit you MUST use a **lambda expression**.

That is, the body of your method should use the `sort` method of `java.util.ArrayList`, passing in an object that implements `Comparator<ArrayList>` in the form of a **lambda expression**.

As a reminder, the general syntax of a Java lambda expression is as follows:

parameters	syntax
x	`(x) -> return_value`
x,y	`(x,y) -> return_value`
etc…	

For half credit: use any technique that will sort the array correctly (e.g. inner class, anonymous class). If you use an inner class, be sure to include the definition of that inner class in your answer.

End of Exam