I. General questions

1. (5 pts. each) Give brief definitions of the following (as they relate to Java and to object-oriented programming)

   a. Class

   b. Object

   c. Subclass

   e. Arithmetic Logic Unit

   f. Constructor (as a method)
II. Software Engineering Issues

1. (5 pts. each) Again, brief definitions related to the course

a. "divide and conquer" as a design strategy

b. coupling

c. cohesion

d. unit testing

e. information hiding
2. (10 pts.) One of the major features of classes and objects is this notion of providing a public interface and a private implementation. Explain how this is done in Java, and how it relates to the notion of information hiding.

3. (10 pts.) The idea of structured programming grew out of problems in Fortran and COBOL and suggested that we should write our programs out of blocks of code having (each block) only one entrance and one exit, tied together by a small collection of simple control structures. There are three such control structures in modern computer languages. Name them, and give an example in Java of each of them.
4. (15 pts.) Consider an alarm clock. Such devices can usually display the current time and allow the user to set the current time and the alarm time. The user can turn the alarm on, off, or to “snooze”. No code is to be written for this problem.

What internal fields might we have for an alarm clock class (simply describe them)?

What methods might we want to implement for an alarm clock class (again, list and describe them)?
III. Classes, objects, and code

1. (15 pts.) Here is a procedure for finding the greatest common divisor of two integers \(a\) and \(b\).

   So long as \(a\) is not equal to \(b\), do the following:
   - if \(b > a\), swap \(a\) and \(b\)
   - subtract \(b\) from \(a\), and set \(a\) to \(b\) and \(b\) to the difference \(a - b\).

   The following illustrates the process for \(a = 21\) and \(b = 15\):

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>15</td>
</tr>
<tr>
<td>15</td>
<td>6</td>
</tr>
<tr>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

   Write suitable code to do this for integer variables \(a\) and \(b\).
For the following series of exercise we will write a portion of the class definition for a Student class. Documentation / Comments are not required for exam questions.

2. (10 pts.) Write the basic Class definition for a class called Student with fields studentName (a String), exam1, exam2, examAverage (all double), and grade (a String). Write the basic class structure and fields, but do not write any methods (yet),

3. (15 pts.) Write two constructors for the class Student, one which accepts only the student name and sets all other fields to zero, and one which accepts the student name and two exam scores (as arguments). What feature of object oriented program does this illustrate by having two methods with the same name? How does the Java compiler tell them apart?
4. (10 pts.) Write a public method called calculate which computes the average of the two exam scores, sets the examAverage field to that value, and sets the grade to “PASS” if the exam average is 70 or better and to “FAIL” otherwise.

5. (15 pts.) Over-ride the toString method to produce a formatted listing of the fields in the Student object. What does it mean to over-ride a method?
IV Collections

Several of the following questions refer to the Part class you have written for various exercises this term, and to various implementations of the Inventory class you have written for course exercises.

Not all of the following questions ask you to write complete methods. Some of them ask simply for short snippets of code. Except as noted you need to write code, not simply refer to a method you have already written for your Inventory object.

1. (10 pts.) Write the code necessary to declare the variable inventory to be a new ArrayList of Part objects.

2. (10 pts.) Illustrate how a new part is added to the ArrayList by creating a new part (using suitable values for the fields) and adding that part to the inventory.
3. (10 pts.) Assuming that a toString method has been written for the Part class, write the code necessary to list all the parts in your ArrayList.
4. (20 pts.) Now using all of the methods you have written for your Part and Inventory classes (without using JOptionPane), write a main program to
   1. Create one of your inventory objects.
   2. Put two or three parts in it
   3. List the inventory (using your listAll method from your Inventory object)
   4. Remove a part (you choose)
   5. List the inventory again.