Please submit written solutions in class on Friday, April 24 for feedback on Monday, April 27. Email solutions accepted until 9AM Saturday, April 25 for feedback on Monday, April 27 (emailers should probably skip question 2). Group work is encouraged (in which case you should only submit one copy of your answers).

1. Fill in the blanks:
   a. Aggregation models the ___________ relationship between a class and its attribute.
   b. Composition models the ___________ relationship between a class and its attribute.
   c. Unlike aggregation, composition implies _____________ of the attribute.

2. Draw the UML class diagram for the following classes:
   a. A class X that aggregates two String instances.

   b. A class Y that aggregates one String and one Date instance.

   c. A class Z that aggregates a List<String> instance where the instance holds zero or more String instances.

   d. A class C that is composed of one String, one Date, and one List<String> instance where the List instance always holds five String instances.

3. Review slides 6—9 from the lecture notes of March 27. With reference to the memory diagram on slide 8 briefly explain why the client code on slide 9 modifies the Date instance internal to the Person instance p.

4. Suppose you have the following class that claims to maintain one or more Date instances in chronological order (ie. the Date instances are sorted from earliest to latest):

```
public class X implement Comparable<Date> {
  public ArrayList<Date> theDates;

  // other attributes, constructors, methods, etc. that ensure
  // theDates is always sorted; theDates is the only attribute
  // holding Date instances

  @Override public Iterator<Date> iterator() {
    return this.theDates.iterator();
  }
}
```

   Is the claim true or false? Explain.

5. Review the Deck class from the March 30 lecture slides. Does Deck have any privacy leaks? Is Deck justified in claiming that it owns its' Cards?

6. Slide 17 from the March 30 lecture slides; second exercise question (add a method that sorts a Deck by rank).

7. Consider the following class:

```
public class X {
  private Y y;
  // constructors, methods, etc.
}
```

   a. Suppose `Y` is a primitive type (`int` for example). Do any of the methods on slide 10 from the April 8 slides cause a privacy leak? Why or why not?

   b. Suppose `Y` is an immutable type (`String` for example). Do any of the methods on slide 10 from the April 8 slides cause a privacy leak? Why or why not?

8.
   a. What is a shallow copy of a collection?

   b. Add one line of code to the following that prints some evidence that `sCopy` is a shallow copy of `dates`:

```
// assume there is an ArrayList<Date> dates
ArrayList<Date> sCopy = new ArrayList<Date>(dates);
```

9. See Chapter 5 notes, page 79—81. To deep copy a collection you need to create a new collection and you need to fill the new collection with new instances that are copies of the instances in the old collection. Show some code that performs a deep copy of the list `dates` from question 8.

10. Consider the Iterator interface.
    a. What does the method hasNext do?
    b. What does the method next do?
    c. What does the method remove do?

11. Inheritance models the is-a relationship between classes. What does is-a mean in Java?

12. Suppose you decide to create a subclass of some class. Consider the constructors of your new subclass.
    a. What must the first line of every constructor be?
    b. You decide that you want to delegate to another constructor. Where can you place the call to the other constructor?
    c. Show how to call the superclass constructor (from a subclass constructor).
    d. Show how to call a constructor of the subclass (from another subclass constructor).