Student Name: _____ Student ID # _____

UOSA Statement of Academic Integrity

On my honor I affirm that I have neither given nor received inappropriate aid in the completion of this exercise.

Signature: _____ Date: _____

Question 1: Inheritance (20 points)

Your text claims (Chapter 11, page 377) that:

> Not all is-a relationships should be modeled using inheritance. For example, a square is a rectangle, but you should not declare a **Square** class to extend a **Rectangle** class, because there is nothing to extend (or supplement) from a rectangle to a square.

Give a counter example to the author's specific example and explain *why* this is a counter example. That is, find some thing T that is true about squares that is not necessarily true about all rectangles, then describe a new variable or method that you could usefully add to **Square** class that makes use of this T. *Explain why this new variable or method is useful.*

Question 2: Abstract Classes & Interfaces (20 points)

In languages that allow for multiple inheritance (such as C++), classes that are derived from multiple parent classes inherit all the variables and methods of all of their parents. Java does not allow for multiple inheritance but it does allow a class to implement multiple interfaces. Your textbook claims (Chapter 10, page 342) that "[w]ith interfaces, you can obtain the effect of multiple inheritance."

A. Is "the effect" referred to by the author inheriting all the variables and methods of all of the interfaces the class implements? *Explain why or why not.*

B. Aside from inheriting variables and/or methods, describe one effect that implementing a method has that is similar to an effect of inheriting from a class.

Question 3: **Object**-Oriented Design (20 points)

Describe the classes you would create and how they are related to one another for the following simple problem:

> You wish to create a program that can show colorful box-plots of data (such as histograms) and can label the boxes. The things your program will be drawing are boxes, colored boxes, labels, labeled boxes, and labeled boxes with color.

You may draw this with a simplified UML class diagram (showing just the class names and the relations between them) or write this out in words.

Question 4: Generics (20 points)

In Chapter 9 of your text, **ArrayList** (page 316) is shown as a self-contained class (that is, a subclass only of **Object**), which deals with objects of class **Object**. In Chapter 22, we can see **ArrayList** (page 724) re-defined to be a subclass of **List** (page 723), which is in turn a subclass of **Collection** (page 715), which uses generics.

A. Why do some of the methods of **Collection** and **List** still take arguments of type **Object**, rather than <*E*> (the generic type designator for the class)?

B. What advantage is provided to a programmer who uses the newer definition of **ArrayList** with generics that wasn't there with the old version of **ArrayList** (which does not use generics)? *Why is this an advantage?*

Question 5: Java Collections Framework (20 points)

The **Collection** class does not contain methods for sorting, reversing, shuffling, or performing binary searches on collections. However, the **Collections** (note the 's') class does.

A. *Why* are these methods *not* found in the **Collection** class?

B. *Why are* these methods found in the **Collections** class?